Approximation of a variable density cloud of points by shrinking a discrete membrane

Jordi Esteve Pere Brunet Àlvar Vinacua

e-mail: [jesteve, brunet, alvar]@lsi.upc.es
Universitat Politècnica de Catalunya. Barcelona

Abstract

This paper describes a method to obtain a closed surface that approximates a general 3D data point set with non-uniform density. Aside from the positions of the initial data points, no other information is used. Particularly, neither the topological relations between the points nor the normal to the surface at the data points are needed. The reconstructed surface does not exactly interpolate the initial data points, but approximates them with a bounded maximum distance. The method allows to reconstruct closed surfaces with arbitrary genus and closed surfaces with disconnected shells.

ACM CSS: I.3.5 Computational Geometry and Object Modeling.
***Keywords**: Scattered data points, surface approximation, voxelization, discrete geometry.*

1. Introduction

Scattered data points obtained from real objects with optical, ultrasonic, tactile or other sensors are frequently used as data sources. Geometric modeling applications must process these scattered data points to obtain a surface that approximates the data point set. In this way,

- The generated surface will be more compact than the initial data point set.
- A more realistic visualization can be obtained from the surface.
- Standard geometric modeling operations (surface evaluation and editing, surface-surface intersection, etc.) will be feasible.

A large diversity of algorithms that approximate scattered data points have been published. There are many valid solutions approximating a cloud of points and each algorithm provides a solution with its own "aesthetic".

It is not possible to detail all published papers dealing with this problem here. The reader can consult some of the existing State-of-the-Art reports[1, 2]. Some papers are mentioned below. They are classified in four blocks according to the taxonomy used in the Mencl and Müller report[1]:

- **Spatial subdivision**. The space is decomposed in cells, then the cells that are stabbed by the final surface (surface oriented algorithms) or the cells that do not belong to the volume bounded by the surface (oriented to volume) are determined and, from them, the final surface is computed. Some surface oriented algorithms use a regular voxelization (Algorri et al.[3] and Hoppe et al.[4]), others decompose the space in tetrahedrals (Bajaj et al.[5], α-shapes of Edelsbrunner et al.[6], Amenta et al.[7] and the Cocone algorithm of Amenta et al.[8]). Examples of volume oriented algorithms are Veltkamp[9], Boissonnat[10] and Isselhard et al.[11] (their strategy is based on computing a Delaunay tetrahedrization of the convex-hull of the data point set and eliminating successively the tetrahedrals carrying out some properties).
- **Distance function**. The distance function calculates the minimum distance from any point of the space to the final surface. The distance function can give positive or negative values if the surface is closed. The final surface is implicitly determined by the distance function. Examples of algorithms using a distance function are Hoppe et al.[4],

Roth et al.[12], Bajaj et al.[5], Boissonnat et al.[13] and the level set method of Zhao et al.[14].

- **Deformation techniques** or **Warping**. An initial surface is deformed progressively to obtain a better approximation to the initial data points. In this kind of algorithm it is convenient to start with a surface that is a coarse approximation of the data point set. Examples are the "blobby model" of Muraki[15], the mass and spring model built from a triangular mesh of Algorri et al.[3] and the variational level set method applied to implicit surfaces[14]. Also the algorithms based on 3D snakes (energy-minimizing spline which is attracted toward the initial point set) can be classified in this category (Takanashi et al.[16] and Kass et al.[17]).
- **Incremental construction**. The surface is constructed in an incremental way from the properties of the initial data points. Normally, from an initial element (edge, triangle), the algorithm works by successively adding new elements (typically triangles) in its neighborhood enlarging the surface. This is the idea of the surface oriented algorithm of Boissonnat[10] and the ball-pivoting algorithm of Bernardini et al.[18].

This taxonomy has several drawbacks. Some algorithms can be included in more than one category. For example, some spatial subdivision algorithms use a distance function to find out which cells are stabbed by the surface[4, 5]. As we will see later, our algorithm can be classified in two different categories. Notice that this taxonomy classifies the algorithms by the method being used, not by the obtained result. For example, spatial subdivision algorithms can get a mesh of triangles[3, 4], tri-variate implicit Bernstein-Bezier patches[5] or polytopes (sets of points, edges, triangles and tetrahedrals not necessarily defining a closed and regular surface) in the α-shapes algorithm[6].

The problem framework is defined in next section. Section 3 introduces an overview of the algorithm. In Section 4 the main part of the algorithm is discussed: the voxelization of the space and the membrane shrinking. Section 5 describes the relaxation of the membrane and section 6 the construction of the final surface. The results are explained in section 7 and the final conclusions are pointed out in section 8.

2. Problem definition

The reconstruction of surfaces from a cloud of points has been intensely scrutinized in previous work when the sampling is sufficiently dense (characterized by their ε-sampling[2] or *r*-sampling[19] property or by other sampling criteria[20]). The ε-sampling criterion, however, cannot be verified in practice. Boissonnat and Oudot[21] discuss a variation of this condition of more practical application when the surface S is explicitely known.

Unfortunately practical models, obtained for instance through 3D scanners, are often not so regular. We are interested in finding reasonably efficient and robust algorithms to extract watertight models from non-uniformly and not densely sampled surfaces.

To this end, let Σ be a solid bounded by a 2-manifold $S = \partial\Sigma$, and let $M = \{p_i\}_{i\in I}$ be a set of point samples on S ($\forall i \in I, p_i \in S$). Similarly to α-shapes[6], we will probe this model, but the probe will be a cube rather than a sphere, for reasons that will become evident.

A valid movement of this cubic probe is defined as a rotation-free movement such that the center of the cube sweeps a 1 or 2-manifold without touching any p_i.

Definition 1. The model M is α-sparse if there is a valid movement of a cube of edge size α that takes it from the exterior of Σ to a position where it touches the inside of S (see Figures 1 and 2).

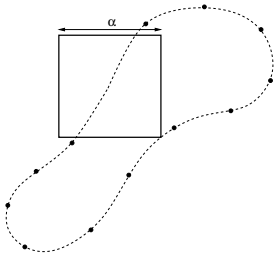


Figure 1: *The points of an α-sparse model M. S is shown as a dashed line.*

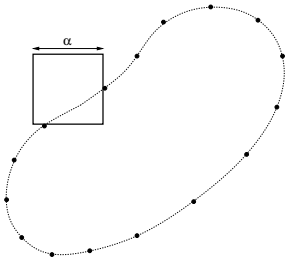


Figure 2: *The points of a non-α-sparse model M*

Definition 2. The model M is β-separable if it is not β-sparse, but Σ has two separate components, and there is a valid movement of a cube of edge size β along a 2-manifold such that the envelope of the cube's positions separates the two components of Σ (see Figure 3).

Definition 3. The model M has a γ-hole if it is not γ-sparse, and there is a valid movement along a 1-manifold of a cube of edge size γ through a hole in Σ (the object in Figure 3 has a β-hole if it is interpreted as the 2D section of a connected genus 1 3D object).

The algorithm we propose, outlined in the next section, is based on a discrete version of these concepts. Therefore we start by immersing the model M (where S is now unknown)

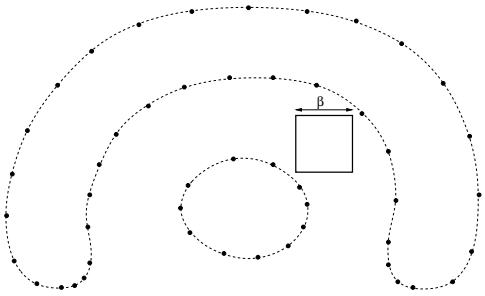


Figure 3: *The points of a* β*-separable model M*

in a voxelization of resolution l. Our probes are cubes made up of these voxels, and valid movements are made up of translations by a multiple of l in the direction of a coordinate axis.

In some aspects our algorithm is similar to the algorithm of α-shapes[6]. This method starts with a Delaunay tetrahedrization and a sphere of a given radius α. The algorithm moves the sphere around without going through data vertices, erasing the tetrahedrals, triangles and edges that it encounters, and obtaining polytopes. The success depends on the selection of the parameter α. The main differences between the two algorithms are the object used to perform the contraction/elimination (a cubic probe of size α vs. a sphere of radius α) and that our method performs a sequence of contractions with cubes of decreasing size to get a unique solution vs. α-shapes, which calculates a family of solutions each of them the result of eroding with a sphere of certain radius.

3. Algorithm overview

The main goal is the construction of a closed surface S that approximates a non uniform data point set M in the 3D space which is known to approximate the boundary of a closed solid Σ (Figure 20(a)). To achieve it, our algorithm uses a regular spatial subdivision and proceeds in the following way:

1. Voxelization of the space that contains the data point set. The voxels are labeled according to whether they have points in their inside (hard voxels) or not (soft voxels) (Figure 20(b)).
2. Obtaining a discrete closed membrane: a set of 6-connected (face-connected) voxels that contain the final surface. This voxel set is formed by 6-connected hard and soft voxels and divide the remaining voxels in inside and outside. It is like a discrete rubber band. Initially a discrete membrane composed by the voxels of the 6 exterior faces of the voxelization is constructed (Figure 21(a)). Then this discrete membrane is contracted at the locations where there are soft voxels (Figures 21(b), 21(c), 21(d) and 21(e)). The hard voxels stop the shrinking. When this membrane cannot be further contracted, the final 6-connected discrete membrane has been found (Figure 21(f)).
3. Relaxation of the discrete membrane to obtain a smoother surface. The soft voxels of the discrete membrane are displaced slightly to reduce the local curvature (Figure 19).
4. Construction of the final surface from the discrete membrane obtained in the previous step (see Figure 22).

Our algorithm can be included in the spatial subdivision category of section 1: It uses a regular voxelization as spatial subdivision and it is oriented to volume since it eliminates progressively those voxels not belonging to the volume bounded by the surface. It does not use any type of distance function, and does not have to calculate distances among points or find the neighbour points as it is usually done by most of the previous algorithms. This strategy provides robustness and efficiency to the algorithm.

Our algorithm could also be classified into the deformation techniques category, since it starts with an initial discrete membrane that it is deformed progressively by contractions. We will also describe an improved version of the algorithm that allows to deal with a set of isolated discrete membranes to obtain a set of closed surfaces with disconnected shells.

Most of the existing algorithms construct a surface stabbing exactly all or almost all initial points. Rather, our algorithm constructs a surface such that the initial points are located at most at a distance d from it, where $d = \sqrt{3}\, l$ is the length of the diagonal of the voxel used. Any surface stabbing the discrete membrane constructed in the second and third step will be a closed surface that approximates the initial data points with a tolerance equal to the diagonal of the voxel d. A surface approximating the data points with a tolerance d is called $d - error$ surface.

The fact that the final surface approximates the initial data points is usually not a major problem, since most of the data point sets are the result of 3D scanners and, therefore, contain error. Voxels whose size has the same order of magnitude as the maximum error produced in the data acquisition can be used. The approximating algorithm has the advantage of filtering the noise and obtaining a smoother final surface.

Some of the main advantages and contributions of the proposed algorithm are:

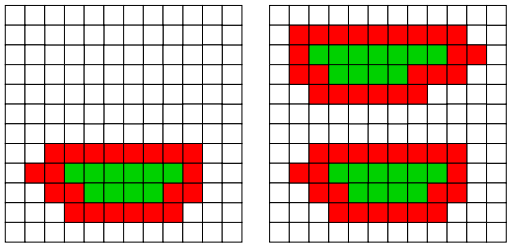
- Reconstruction of surfaces although the initial data point set does not have a uniform density.
- Reconstruction of objects with $genus \geq 1$ and/or objects with disconnected closed shells.
- A final closed object is guaranteed.
- Error-bounded approximation supporting a final fairing.
- Explicit conditions on the genus of the final object.
- Robust and efficient algorithm.

4. The discrete membrane shrinking

4.1. Basic concepts involved in our algorithm

The algorithm works in a spatial division (**voxelization**) of a rectangular box containing the cloud of points with cubes of the same size. In our problem the voxels are classified in hard (those that have one or more initial data points inside) and soft (those that do not have any data points).

Some voxels of the voxelization belong to a **discrete membrane** (DM). The DM is a set of face-connected voxels (6-connected) that divides the remaining voxels in inside and outside (there is no face-connected, edge-connected or vertex-connected path (26-connected path) that allows to go from an inside voxel to an outside voxel that does not include any voxel of the DM). Figure 4(a) shows a 2D voxelization with one DM (red voxels): The white voxels are outside and the green ones are inside in relation to the DM.



(a) Discrete Membrane (DM)

(b) Discrete Membrane Set (DMS)

Figure 4: *Examples of 2D Discrete Membranes*

The DM is a discretization of the surface S. The solid Σ is composed by the voxels belonging to the DM and the inside voxels.

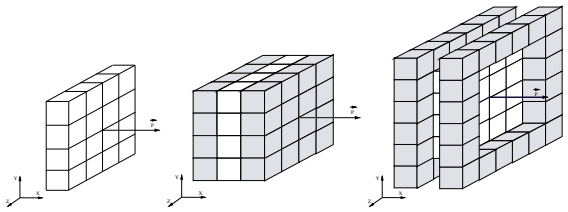
To deal with objects with disconnected shells we must allow for several membranes inside the voxelization (i.e. sets of discrete membranes, or DMS). Any voxel of the voxelization can be classified according to:

- HARDNESS: Hard, soft.
- POSITION: Inside (the voxel is inside one DM), Outside (the voxel is outside of all DM's), Boundary (the voxel belongs to a DM).

Figure 4(b) illustrates a 2D voxelization with 2 DM's: The red voxels are boundary, the white ones are outside and the green ones are inside in relation to the DMS.

Some definitions are introduced now to help discussing the ideas used in the algorithm.

Plate of size n: Set of $n \times n$ contiguous voxels that form a square parallel to a co-ordinate plane (all plate's voxels are face-connected). We will define the orientation of the plate as a perpendicular vector $\vec{p}$ to the plate that allows to distinguish its front and back side. See Figure 5(a).



(a) Plate with its orientation

(b) Front and Back Side

(c) Lateral Front and Back Side

Figure 5: *Plate of size $n = 4$*

The **Front Side** (Back Side) of a plate of size n are the $n \times n$ voxels located in front of (behind) the plate according to its orientation. See Figure 5(b).

The **Lateral Side** of a plate of size n are the $4 \times (n+1)$ voxels located around the plate.

The **Lateral Front Side** (Lateral Back Side) of a plate of size n are the $4 \times (n+1)$ voxels located around the front side (back side) of the plate. See Figure 5(c).

Model Volume: It is the sum of the number of inside voxels plus boundary voxels within the voxelization.

The **6-neighborhood** of a voxel V is the set of 6 voxels sharing a face with V.

The **26-neighborhood** of a voxel V is the set of $3 \times 3 \times 3 - 1$ voxels that share a vertex, edge or face with V.

The **124-neighborhood** of a voxel V is the set of $5 \times 5 \times 5 - 1$ neighbour voxels of V with V at the center.

In our algorithm the plate represents the frontal face of the cubic probes in definitions 1, 2 and 3. Only the frontal face of the cubic probes will interact with the DMS. The DMS evolves changing the position of the voxels belonging to a plate and also the voxels in its vicinity (front, lateral and lateral front side of the plate).

4.2. DMS operations

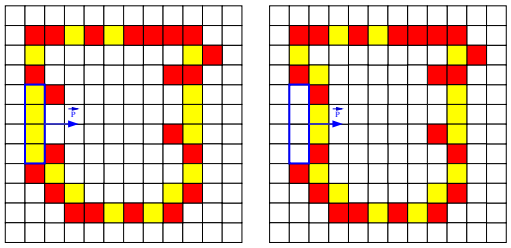
Three different internal operations will be used to modify locally the DMS: CONTRACTION (the interior volume is shrunk), UNDO CONTRACTION (a contraction is reversed) and FREEZING (conversion of boundary soft voxels to hard voxels).

The **CONTRACTION** operation modifies the DMS with a plate of size n that satisfies the following conditions:

1. It is composed uniquely of one or more boundary soft voxels and outside voxels (i. e. does not contain hard voxels or inside voxels).
2. The back side of the plate is composed only of outside voxels.

The operation modifies the DMS as follows:

1. The boundary soft voxels that belong to the plate are converted to outside voxels.
2. The front, lateral and lateral front voxels of the plate ($8 \times (n+1) + n \times n$ voxels) that were inside are converted to boundary voxels preserving their hardness.



(a) Before (b) After

Figure 6: *CONTRACTION operation with a plate of size $n = 4$. The hard voxels are painted in red and the boundary soft voxels in yellow.*

Figure 6 shows how the CONTRACTION operation is applied in a 2D voxelization. The plate of size $n = 4$ (blue color) in the 2D case is one-dimensional. Observe how the boundary voxels overlapped with the plate are converted to outside voxels and the interior front, lateral and front lateral voxels are converted to boundary voxels preserving their hardness.

The properties of the CONTRACTION operation are:

1. Reduction of the model volume, since the plate must be located on one or more boundary voxels and these are converted to outside voxels.
2. It is an internal operation: The DMS is transformed to another DMS. The result are sets of face-connected voxels that divide the rest of the voxels in inside and outside. This is because the voxels of the plate are replaced by the front, lateral and lateral front side of the plate. Notice that the front, lateral and lateral front side of the plate are face-connected voxels and completely separate any interior voxels from the voxels of the plate ($FinalDM = InitialDM - plate + front + lateral + lateral\,front$).
3. The hard voxels are never converted to outside ones. Only the boundary soft voxels overlapped with the plate are marked as outside.

Note: Not only a DM is contracted when the CONTRACTION operation is applied; also the cardinality of the DMS can be incremented: the topology of the DMS may change, which is crucial for the ability of our algorithm to deal with models with holes and/or several connected components. See Subsection 4.5 and Figure 13 for more details.

The **UNDO CONTRACTION** operation reverses the last CONTRACTION operation performed, recovering the previous state. From the properties of CONTRACTION it is obvious that UNDO CONTRACTION increases the model volume and yields a DMS.

The **FREEZING** operation converts the boundary soft voxels that are overlapped with a plate of size n in frozen soft voxels which behave as hard voxels in all further tests, except where indicated. Obviously this operation does not alter the model volume and yields an DMS (as no voxels have been added or removed from it).

The three previous operations depend on the size of the plate n used. As discussed in the next Subsection, the algorithm works with diminishing plate sizes to obtain successive DMS generations, each of them more contracted due to the smaller size of the plate. A counter of the different plate sizes n used by the algorithm allows to know the current generation. We store the **generation** as a property of the outside voxels. We therefore know in which generation each of these voxels was converted to outside (what plate size n caused the conversion from boundary voxel to outside voxel). See in Figure 7 how the outside voxels are labeled with the 1, 2, 3 and 4 generations that correspond to the plate sizes $n = 12$, $n = 6$, $n = 3$ and $n = 2$ respectively.

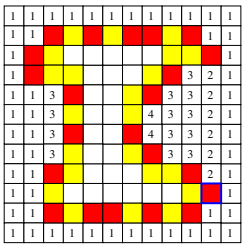


Figure 7: *Labeling the outside voxels with their generation*

To detect whether the data point set M is α-sparse a discrete Incursion Test is defined. The cubic probe of size α is replaced by a discrete version of its frontal face: a plate of size n where $\alpha = n \cdot l$. Recall that l is the length of the side of a voxel. This Incursion Test detects when the plate of size n reaches the interior face of the DM boundary (see Figure 9). The Incursion Test definition is based on the Local-Arc Connectivity concept.

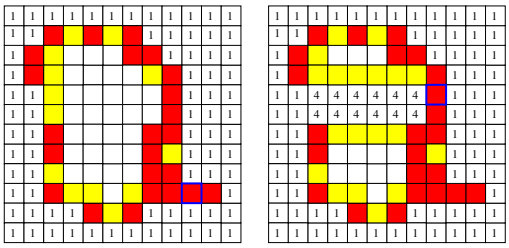
Local-Arc Connectivity: Two voxels in the 26-neighborhood of the voxel V are local-arc connected when there exists a face-connected path of outside voxels that connects them in this neighborhood of the voxel V.

Observe, for example, the highlighted hard voxel at bot-

tom right of Figure 7. The outside voxels located above and below of the highlighted voxel are local-arc connected.

Incursion Test: An incursion is detected when two outside voxels of different generations at the opposed faces of a boundary hard voxel are not local-arc connected between them. How this relates to the continuous definition is discussed in section 4.4.

Observe Figure 8(a): No incursion has been detected in the highlighted hard voxel since the voxels located at the opposed faces, though not local-arc connected, have the same generation. Instead, in Figure 8(b), an incursion has been detected since the voxels located at the opposed faces have different generation.



(a) No (b) Yes

Figure 8: *Incursion*

As is explained below, the algorithm, after applying the CONTRACTION operation, will check if an incursion has been produced in the hard voxels of the front, lateral and lateral front side of the plate to decide the convenience of applying the UNDO CONTRACTION and FREEZING operations.

4.3. Algorithm

Algorithm 1 illustrates the main reconstruction algorithm.

The first step builds the voxelization from the initial data point set. This step is immediate if the edge's length of the voxel l is known: We must divide the lengths (l_x, l_y, l_z) of the sides of the rectangular box that contains the data point set by the edge's length of the voxel to obtain the sizes of the voxelization $(n_x = l_x/l, n_y = l_y/l, n_z = l_z/l)$. When we build the voxelization, the voxels that contain one or more points of the initial set are labeled as hard voxels and the others as soft voxels. In Figure 20(b) the hard voxels of the voxelization of the data points of Figure 20(a) have been displayed.

If the length l is not known, it can be estimated as

$$l \approx \sqrt{\frac{2\,(l_x l_y + l_y l_z + l_z l_x)}{n_{points}}} \qquad (1)$$

where n_{points} is the number of the initial data points. This

Algorithm 1 Main algorithm scheme

```
vox := Voxelization( CloudOfPoints );
DMS := InitialDiscreteMembrane( vox );
n := max( vox.size.x, vox.size.y, vox.size.z );
repeat
  n := (n+1) div 2;
  FindContractionPlace( DMS, n, placeOk, place );
  while placeOk do
    incursion := FALSE;
    NewStack( ContractionStack );
    RecursiveContraction( DMS, n, place, Contraction-
    Stack, incursion );
    FindContractionPlace( DMS, n, placeOk, place );
  end while
until n = 1
```

heuristics considers that the data point set defines a surface of similar extension as the 6 faces of the rectangular box that contains it. In all examples shown in Section 7 this formula has been used, getting satisfactory results.

When the data point set has a very irregular sampling density, it is convenient to calculate the ratio number of voxels containing two or more points divided by the number of voxels containing one point. If this value is too high we can recalculate the voxelization decreasing the edge's length of the voxel l.

From here on we will use the chosen l as unit length.

The second step constructs the initial discrete membrane with the voxels belonging to the exterior faces of the voxelization. From then on the algorithm applies the 3 defined operations (CONTRACTION, UNDO CONTRACTION, FREEZING) to contract gradually the DMS until it is adapted to the hard voxels. Concretely:

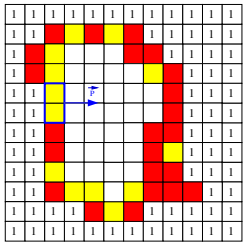
- It applies the CONTRACTION operation with a plate of size $n \simeq$ *Voxelization Size* and the plate size is reduced progressively until $n = 1$. Each reduction of n defines a new generation.
- From a fixed value of n, it searches in all boundary soft voxels the locations where a CONTRACTION operation with a plate of size n can be applied. The locations found are the starting points to apply recursively the CONTRACTION operation.

To find easily a place where to start a recursive contraction operation (function *FindContractionPlace()*), the boundary soft voxels are stored in a list. This list is searched for the first voxel that satisfies the required conditions to perform a contraction operation (see section 4.2).

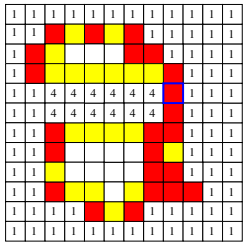
The results of the algorithm depend on the choice of the initial plate size and how it is reduced progressively. The best solution, though with more computational cost, is to start with a plate size equal to the maximum of the three voxelization sizes $n := max(nx, ny, nz)$ and decreasing the plate

size 1 by 1. However, with data point sets obtained from 3D scanners, normally it is sufficient to divide the size of the plate by 2 in each iteration $n := (n+1)/2$. In this way, the run-time of the algorithm is reduced considerably. The results shown in Section 7 show that this choice is reasonable under widely varying conditions. The algorithm can be sped up even more with a pre-process consisting in shrinking the initial DM to get the exterior silhouette of the hard voxels in the X, Y and Z directions (Figure 21(b)).

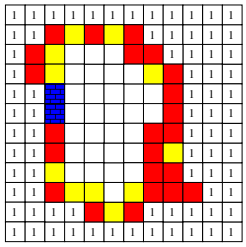
From a fixed value n and a particular location where the CONTRACTION operation can be applied, the CONTRACTION operation is applied recursively at the front, up, down, left and right directions in relation to the plate orientation (always with a plate of size n). Every time the CONTRACTION operation is applied, it is checked for incursion in the hard voxels of the front, lateral and lateral front side of the plate. If an incursion is detected (Figure 9(b)), the UNDO CONTRACTION operation is applied as many times as CONTRACTION operations were performed. Then a FREEZING operation is applied (Figure 9(c)). This cancels an undesired expansion of the plate inside the solid.



(a) Before starting the shrinking



(b) Shrinking with $n = 2$. Incursion detection



(c) Backtracking and freezing

Figure 9: *Incursion detection*

When the areas of undersampling are big, the algorithm is also capable to fill the holes. The initial size of the plate to perform the contractions must be bigger than the holes.

Function 1 shows the implementation of the recursive contraction. The stack *ContractionStack* saves the CONTRACTION operations performed to undo them in inverse order if an incursion is detected. A global Boolean variable *incursion* is used to indicate the fact that an incursion has been detected and avoid doing more CONTRACTION operations into the remaining calls to the *RecursiveContraction* function.

Function 1 Scheme of the contraction recursive function

```
function RecursiveContraction( DMS, n, place, ContractionStack, incursion )
if incursion then
   return;
end if
c := CONTRACTION( DMS, n, place );
PushStack( ContractionStack, c );
if IncursionTest( c ) then
   while NOT EmptyStack( ContractionStack ) do
      c := TopStack( ContractionStack );
      PopStack( ContractionStack );
      UNDOCONTRACTION( DMS, n, c );
   end while
   FREEZING( DMS, n, place );
   incursion := TRUE;
   return;
end if
for direction := {front, up, down, right, left} do
   place2 := place + direction;
   if ContractionOk( DMS, n, place2 ) then
      RecursiveContraction( DMS, n, place2, ContractionStack, incursion );
   end if
end for
end function
```

4.4. Algorithm properties

In this section we will establish some fundamental properties of our algorithm:

- **Property 1**: The algorithm always terminates.
  The algorithm always finishes because each iteration either decreases the model volume or it decreases the number of soft voxels in the DMS.
- **Property 2**: The result is a DMS (the voxels are face-connected and they divide the remaining voxels in outside and inside).
  The algorithm's result is a DMS because of starting with an initial DM and applying internal operations (the 3 operations transform the DMS to other DMS; the CONTRACTION operation can increase the cardinality of the DMS, see section 4.5). Therefore each DM of the final set is face-connected and divides the remaining voxels in outside and inside.

- **Property 3**: The final boundary voxels that have an outside voxel in their 6-neighborhood are necessarily hard or frozen soft voxels.
  The boundary voxels having an outside voxel in their 6-neighborhood are hard or frozen soft voxels, never regular soft voxels. The reason is that the algorithm works with plates of diminishing size arriving to a size of $n = 1$. When the plate size is $n = 1$, any boundary soft voxel having an outside voxel in its 6-neighborhood is a suitable location to initiate a sequence of CONTRACTION operations. If an incursion is not detected, this soft voxel has been converted to outside. Otherwise a backtracking and a freezing would have happened converting the soft voxel to a frozen voxel.
- **Property 4**: Outside voxels are never hard.
  The outside voxels are always soft, since the unique operation producing outside voxels is CONTRACTION and this operation only converts to outside voxels the soft ones.

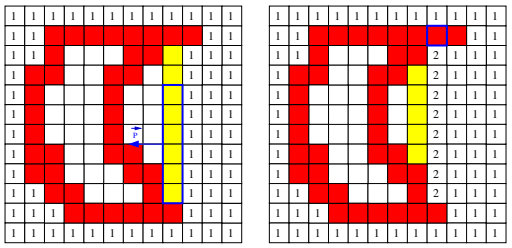
Next we focus on the properties of the Incursion Test.

The surface reconstruction from a data point set is a complex problem. Given an Algorithm A, one can always find a data point set that causes A to produce a surface different from "the expected" result. Furthermore, the problem is more complex the more flexible the initial conditions and the final results are, such as in our case (the initial data point set has no requirement about its density distribution and the final result can be a set of closed surfaces of any genus).

Consider a model M that is not α-sparse, where $\alpha \geq 1$ (the data point density is uniform and has the same magnitude as the voxelization resolution). In this case, for any size of the plate used in the shrinking process, an incursion detection should not occur. However protuberances one-voxel thick (Figure 10) and situations where the data, at the chosen resolution, correspond to a non-manifold discrete surface (Figure 11) may cause failure. The model M is not α-sparse ($\alpha \geq 1$) but in these cases incursions are detected and UNDO CONTRACTION operations are performed so the final result and the expected result differ.

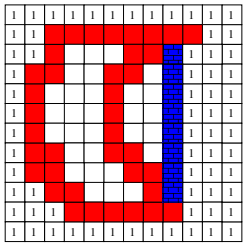
Cases like that in Figure 11 are really improbable and correspond to an ill-choice of the discretization scale. So, we have striven to deal correctly with protuberances one-voxel thick. The Local-Arc Connectivity condition in our definition of Incursion Test addresses part of the problem:

- **Property 5**: If the hard voxels present a protuberance of one voxel, the algorithm will not trigger an unwanted backtracking.
  Observe the highlighted hard voxel located at left of Figure 12: An incursion is not detected because there is local-arc connectivity between the two outside voxels located at the opposed faces of the highlighted hard voxel.



(a) Before starting the shrinking

(b) Shrinking with $n = 6$. Incursion detection



(c) Backtracking and freezing

Figure 10: *Undesirable incursion: Protuberance one-voxel thick*

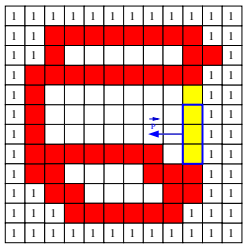
Similarly the labeling of the generation and its use in our definition of Incursion Test addresses protuberances one-voxel thick:

- **Property 6**: If the hard voxels present a protuberance of more than one voxel, the algorithm will not backtrack unnecessarily if both sides of the protuberance belong to the same generation.
  See the highlighted hard voxel at bottom right of Figure 12: An incursion has not been detected because the two outside voxels sharing the opposed faces of the highlighted hard voxel have the same generation.

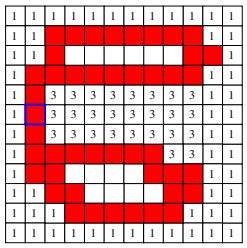
It is difficult to accept other types of protuberances without losing the capability to detect incursions when the data point density is no longer homogeneous. It is impossible to distinguish if a protuberance one-voxel thick is due to the shape of the surface to reconstruct or to a low data point density nearby.

4.5. Mitosis and increase of the genus

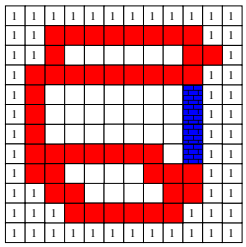
Our algorithm is also capable of recovering the correct surface of a data point set model M that is β-separable or has a γ-hole. To achieve this the algorithm can remove DM pieces



(a) Before starting the shrinking



(b) Shrinking with $n = 3$. Incursion detection



(c) Backtracking and freezing

Figure 11: *Undesirable incursion: Non-manifold discrete surface*

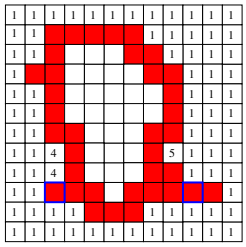
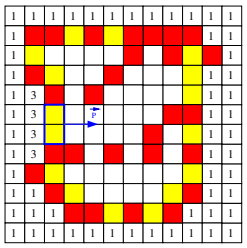


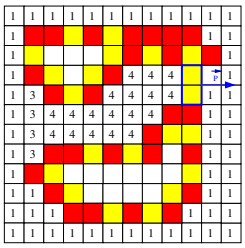
Figure 12: *Discrete membrane with one-voxel thick protuberances correctly processed*

formed only by soft voxels because the Incursion Test is applied in the boundary hard voxels but not in the soft ones. This produces a mitosis of the DM (increase of the cardinality of the DMS) or, in the 3D case, it may result in an increase of the genus of the DM instead. Figure 13 illustrates a mitosis and Figure 21(c),(d) (at the handle) an increase of the genus.

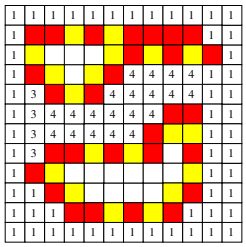
Unfortunately this cannot be achieved in all cases. A protuberance one-voxel thick at the rim of the gap between two components, or of a hole may cause an undue incursion detection. We call these protuberances **obstructions** (see figure



(a) Before starting the shrinking



(b) Shrinking with $n = 2$



(c) End of the shrinking

Figure 13: *Shrinking that produces a mitosis of size* 2

15). In the absence of these artifacts, however, the algorithm performs correctly.

Property 7. The algorithm produces a mitosis if for a certain plate size n:

- The model M is β-separable, $\beta \geq n+1$.
- The separation between the two components exhibits no obstructions.

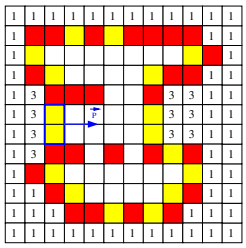
In the example of Figure 13 a mitosis for $n = 2$ is produced: The model M is β-separable and there are no obstructions.

Figure 14 shows an example where there is not a mitosis of size 2. The model is not β-separable because it is β-sparse. The data point density (hard voxels density) around the possible separation is not sufficiently big in relation to the size of the gap.

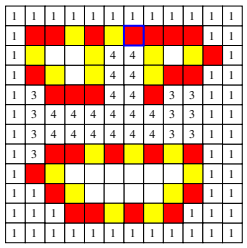
Property 8. The algorithm produces an increase of the genus if for a certain plate size n:

- The model M has a γ-hole, $\gamma \geq n+1$.
- The hole exhibits no obstructions.

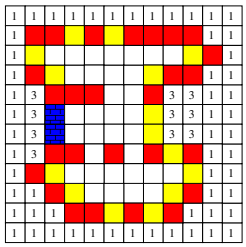
The last CONTRACTION operation causing a mitosis/increase of the genus (see the evolution of Figure 13(b) to the 13(c)) also produces a reduction of the model volume: Though the number of inside voxels is the same, the number of boundary voxels is reduced.



(a) Before starting the shrinking

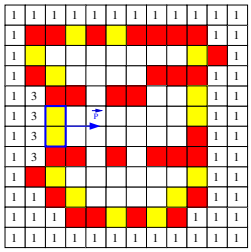


(b) Shrinking with $n = 2$. Incursion detection

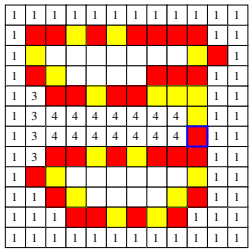


(c) Backtracking and freezing

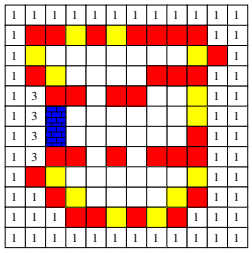
Figure 14: *Mitosis not performed due to a low data point density*



(a) Before starting the shrinking



(b) Shrinking with $n = 2$. Incursion detection



(c) Backtracking and freezing

Figure 15: *Mitosis not performed due to an obstruction*

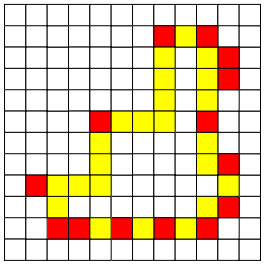


Figure 16: *Planar regions in the final DMS*

5. The discrete membrane relaxation

The results obtained in the previous Section are influenced by the shape of the object (the plate) used in the successive contractions. In our case, due to the flat shape of the plates, the final DMS shows flat regions or strongly stepping in those zones where the initial data point density was low (observe the upper left zone of Figure 16).

It is convenient to relax the DMS before obtaining the final surface. The goal is keep the hard voxels and possibly displace the soft voxels to smooth the local curvature on the boundary voxels. We define a measure of the discrete local curvature that will give a magnitude of the curvature on a boundary voxel from the configuration of the neighbours voxels. The relaxing process will attempt to decrease the discrete local curvature on the whole DMS by performing local moves of the boundary soft voxels.

Discrete Local Curvature: We define the discrete local curvature of a boundary voxel as the difference between the number of outside voxels less the number of inside voxels in its 26-neighborhood. The discrete local curvature of a voxel V will be named $DLC(V)$. The DLC values are in the interval $[-25, 25]$. Figure 17 shows the DLC values of the boundary voxels in the 2D case (the 2D DLC values are in this case in the interval $[-7, 7]$).

The relaxing process first selects the boundary soft voxels V with $DLC(V) \leq -13$. It starts to process the most negative ones and finishes with those soft voxels having $DLC(V) = -13$. To each selected voxel V the following steps are applied:

1. The voxel V is converted to inside voxel (Figure 18(a)).
2. The outside voxels of the 26-neighborhood of V are converted to boundary voxels (Figure 18(b)).
3. Recalculate the DLC on the boundary voxels in the 124-

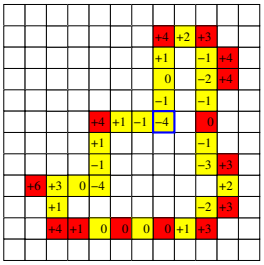
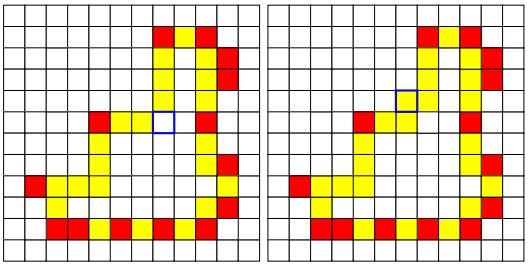


Figure 17: *Discrete Local Curvature (DLC) of the boundary voxels*

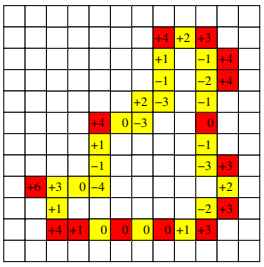
neighborhood of the voxel V (Figure 18(c)). Thus we assure that all boundary voxels that could be affected by the changes in the steps 1 and 2 have the DLC updated. The voxels with $DLC(V) \leq -13$ are added to the selected voxel list.

Figure 18 illustrates the steps described previously applied on the highlighted voxel of Figure 17.



(a) Elimination of the selected voxel

(b) Outside to boundary voxel conversion



(c) DLC recalculation

Figure 18: *2D Relaxing step*

Afterward the above steps are applied symmetrically to soft voxels with $DLC(V) > 0$ starting from the higher value to $DLC(V) = 13$.

When the DLC of the selected voxel V is sufficiently high, this method achieves a decrease of the curvature in the nearby zone. It has been observed in the performed 3D tests that the best results are obtained if the relaxing algorithm only selects the voxels V with $|DLC(V)| \geq 13$ (in the 2D case $|DLC(V)| \geq 4$). Threshold values of the DLC less than 13 causes the final DMS to present flat zones parallel to the planes of co-ordinates or at 45^o degrees. Moreover it causes the relaxing process performed on voxels V with $DLC(V) > 0$ to destroy the previous relaxation done on voxels V' with $DLC(V') < 0$. Applying the relaxing steps described previously on some voxel configurations with $DLC(V) = 12$ generates other boundary voxel V' with $DLC(V') = -12$ and vice versa.

Figure 19 shows the teapot of Figure 21(f) after applying the relaxing process on the boundary voxels.

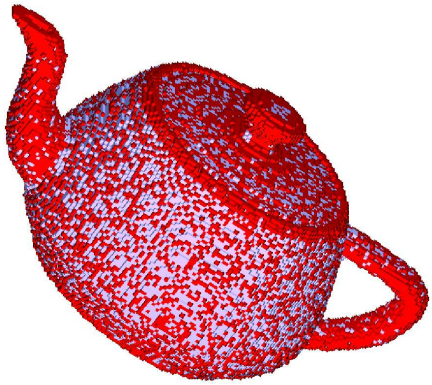


Figure 19: *3D Relaxing*

6. Final surface construction

The last step must build a surface stabbing the DMS obtained in previous steps. For example, the discrete marching cubes algorithm[22] can be used to obtain a mesh of triangles, the algorithm described in Vinacua et al.[23] to obtain a smooth surface (a piecewise algebraic surface defined as a cubic Bspline isosurface) or, simply, using the outside faces of the 6-connected boundary voxel sets to obtain a cuberille model.

In Esteve et al.[24], an improved conversion to a piecewise algebraic surface has been proposed. The construction of this piecewise algebraic surface stabbing the DMS uses fairing techniques to achieve smoother surfaces and takes advantage of the fact that the initial data points lie in the hard voxels, to better approximate the final surface. First an initial isosurface is calculated setting positive or negative weights to the vertices of the DMS depending on if the neighbour voxels are exterior or interior. Then a battery of filters[24] are applied to fair the surface and approximate the central point of the

hard voxels (in order to work with pre-calculated filters getting faster algorithms). From this surface, a multiresolution model can be constructed using the techniques that we describe in Esteve et al.[25].

Figure 22 shows the final surface obtained with a discrete marching cubes algorithm and with the piecewise algebraic surface fitting and fairing algorithm from the relaxed discrete membrane of the teapot.

A triangle mesh can also be obtained from the piecewise algebraic surface. The vertex values and normals of the voxelization are calculated from the piecewise algebraic surface and a triangle mesh is computed with a marching cubes algorithm[26].

7. Results

The first example is the approximation to the vertices of the Utah teapot. It is the same that has been used to illustrate the explanations in the previous sections.

A voxelization of size $164 \times 104 \times 82$ has been built from a set of 35910 points (Figure 20). There are 19144 hard voxels in the voxelization since 16766 points have coincided in voxels where there were other points. Then the initial DM has been built, a previous silhouette shrinking has been performed and the DM has been contracted with plates of sizes 41, 21, 11, 6, 3, 2 and 1 (Figure 21). The table 1 shows the evolution of the DM shrinking. In the table, for each size of the plate, the number of soft voxels not frozen in the DMS just before beginning the shrinking, the number of initiated contractions with this plate size (number of different locations in the DMS where a recursive sequence of CONTRACTION operations has been started to apply) and the number of backtrackings performed (number of times that a recursive sequence of CONTRACTION operations has been canceled and undone), are shown.

Table 1: *Teapot gradual shrinking*

Size of the plate	Number soft voxels in the DMS	Number of started contractions	Number of backtrackings made
41	26105	3480	0
21	21276	53	0
11	20353	105	0
6	19172	112	0
3	18432	336	108
2	16913	987	718
1	12405	8850	8699

The final DM is composed of 18817 hard voxels (painted in red) and 25822 frozen voxels (painted in blue). Inside the DM 356639 soft voxels and 327 hard voxels remain. Hence a little percentage of hard voxels stay inside the DM (1.7%). Most of the data points (98.3%) are within the bounded distance to the surface. In all cases a post-process can be performed to connect the remaining inside hard voxels with the DM at the closest locations. Figure 22 shows the final surface obtained from the DM with two different algorithms.

The second example uses a set of 100461 points modeling a toy bird. The voxelization has a size of $153 \times 209 \times 203$ with 73156 hard voxels (27305 points are within hard voxels that already contained other data points). The final DM (Figure 23(a)) is composed of 71386 hard voxels and 61618 soft voxels. Inside the DM 1228510 soft voxels and 1770 hard voxels (a 2.4% of the total of hard voxels) remain. The final surface, a cubic Bspline isosurface displayed with a raycasting algorithm, appears in Figure 23(b). The table 2 shows the evolution of the DM shrinking of the toy bird. A backtracking with a plate of size 19 has been done because the initial data point set has an area without any point (see the back of the toy bird in Figures 23(c) and 23(d)).

Table 2: *Toy Bird gradual shrinking*

Size of the plate	Number soft voxels in the DMS	Number of started contractions	Number of backtrackings made
38	103134	9152	0
19	63478	1397	1
10	45126	822	9
5	32808	450	19
3	26023	312	33
2	23605	596	46
1	21490	13184	12311

The third example uses a set of 56306 points modeling a dinosaur. The voxelization has a size of $201 \times 170 \times 70$ with 31089 hard voxels (25217 points are within hard voxels that already contained other data points). The final DM (Figure 24(a)) is composed of 31067 hard voxels and 11250 soft voxels. Inside the DM 90855 soft voxels and 22 hard voxels (a 0.07% of the total of hard voxels) remain. The final surface displayed with a raycasting algorithm appears in Figure 24(b). The table 3 shows the evolution of the DM shrinking of the dinosaur.

Finally Figure 25 shows the DM obtained from a cloud of 543652 points modeling a Buddha sculpture. Notice how the algorithm is capable to recover the holes of the model. The

Table 3: *Dinosaur gradual shrinking*

Size of the plate	Number soft voxels in the DMS	Number of started contractions	Number of backtrackings made
35	43025	2503	0
18	27934	600	0
9	14780	330	0
5	9060	250	0
3	5791	289	0
2	3185	316	2
1	1469	639	192

size of the voxelization is $309 \times 745 \times 309$ with 375725 hard voxels in its interior. The table 4 shows the evolution of the DM shrinking. Figure 26 shows two different views of the final surface.

Table 4: *Buddha gradual shrinking*

Size of the plate	Number soft voxels in the DMS	Number of started contractions	Number of backtrackings made
50	688471	33132	0
25	975176	9285	0
13	946651	10350	4
7	847419	7801	28
4	761650	10734	4012
2	556938	40922	33644
1	312134	199489	197343

All these examples illustrate a correct and robust operation of the gradual DM shrinking algorithm.

The run-times of the whole shrinking process are listed in Table 5. These times have been obtained on a personal computer with a 1.6GHz AMD-XP CPU and 256Mb of main memory. The reasons for the high run-time of the Buddha model are the big size of the voxelization, the big size the initial plate (50×50) and the use of swap memory on hard disk to store the model.

We have compared these results with those achieved by Cocone[8] on the same models. The Cocone algorithm enjoys theoretical guarantees and a faster runtime. However, on

Table 5: *Run-time of the shrinking process*

Model	Voxelization size	Time (h:mm:ss)
Teapot	$164 \times 104 \times 82$	0:33
Toy bird	$153 \times 209 \times 203$	1:59
Dinosaur	$201 \times 170 \times 70$	0:33
Buddha	$309 \times 745 \times 309$	6:35:00

models like these where the hypothesis required for the theoretical guarantees fail, the algorithm produces models with errors. Depending on the choice of the *flatness-ratio* parameter, the algorithm either introduces spurious triangles (for small values of the parameter) or generates spurious holes (for large values).

For instance, the large hole at the base of the bird model in figure 23 violates the ε-sampling hypothesis. Cocone fails to fill up that hole, but in its attempts to do so, generates several artifact triangles.

8. Conclusions

An algorithm that obtains one or more closed surfaces approximating a data point set in the 3D space has been presented. The algorithm does not require to know the topological relations among the points or other additional information. The obtained surface approximates and does not stab exactly the points: The approximation error has a certain tolerance d in relation to the initial data point set. By using a shrinking plate of diminishing size, the algorithm allows to reconstruct surfaces from initial data points not having a uniform density. Surfaces with *genus* ≥ 1 and/or surfaces with disconnected shells can be reconstructed due to the way of detecting the incursions to the interior of the surface. The algorithm is robust and efficient since it works only with discrete values (voxels) and does not need to calculate distances among points or find the neighbour points (as normally done by most of the existing algorithms).

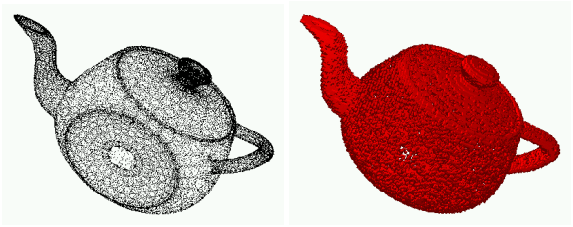
The algorithm could be improved with the employment of hierarchical structures like octrees. But it is important to remark that the success of the algorithm is based on displacing the plate one voxel every time.

A post process to relax the final membranes has been introduced to get better results in zones where the initial data point density was low. Different types of surfaces can be obtained from the final membranes like meshes of triangles or piecewise algebraic surfaces.

The construction of a piecewise algebraic surface stabbing the final discrete membrane includes a constrained fairing algorithm[24] to achieve smoother surfaces and to approximate better the final surface to the initial data points.

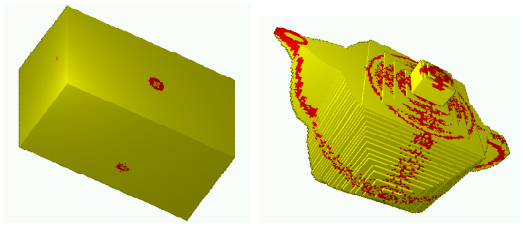
Acknowledgments:

We wish to acknowledge the helpful remarks of the anonymous reviewers, which have contributed to largely improve this paper. We also thank the Spanish Ministery of Science for support under grant TIN-2004-08065-C02-01, and the Stanford 3D Scaning Repository and the CNR Visual Computing Lab for suppling some of the test data.

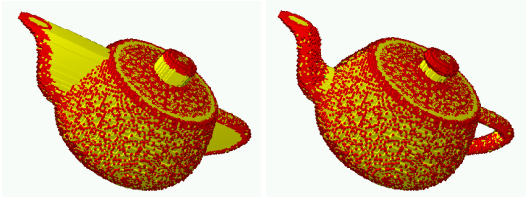


(a) Initial data point set (b) Voxelization

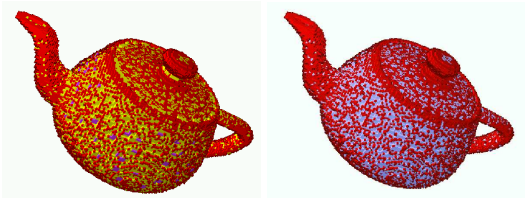
Figure 20: *Voxelization of a cloud of points*



(a) Initial discrete membrane (b) Silhouette shrinking

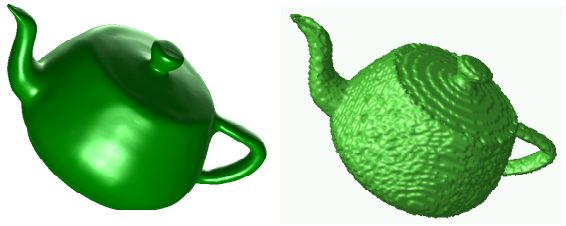


(c) Shrinking with $n = 41$ (d) Shrinking with $n = 6$



(e) Shrinking with $n = 3$ (f) $n = 1$: Final discrete membrane

Figure 21: *Discrete membrane shrinking*



(a) Cubic Bspline isosurface obtained by the algorithm described in Esteve et al.[24]

(b) Mesh of triangles obtained by discrete marching cubes[22] (displayed with a Gouraud smoothing)

Figure 22: *Construction of the surface stabbing the final DMS*

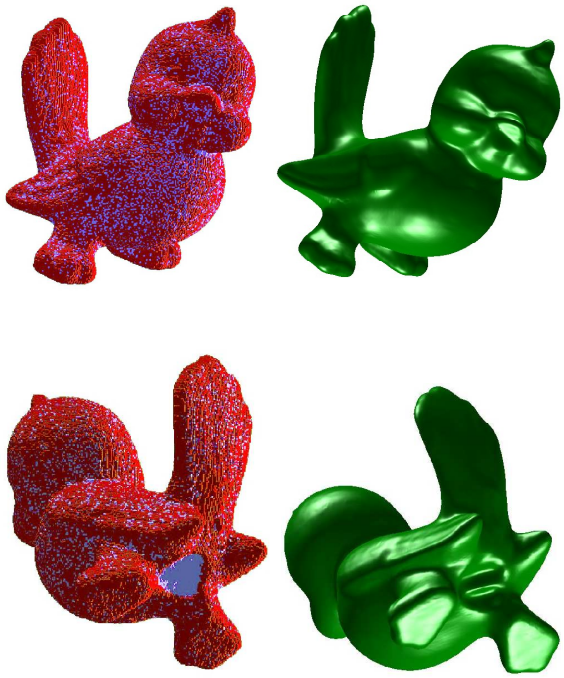


Figure 23: *Toy bird: Discrete membrane, final surface and detail of the big hole filled up*

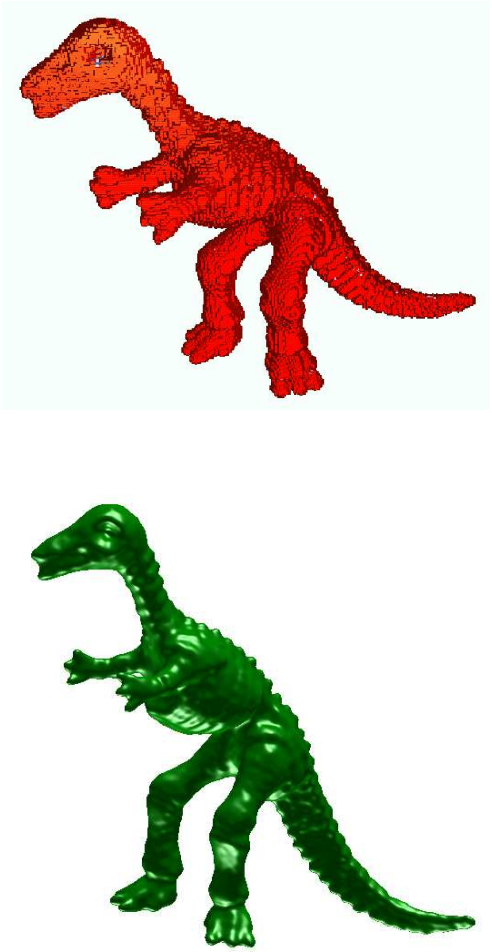


Figure 24: *Dinosaur: Discrete membrane and final surface*

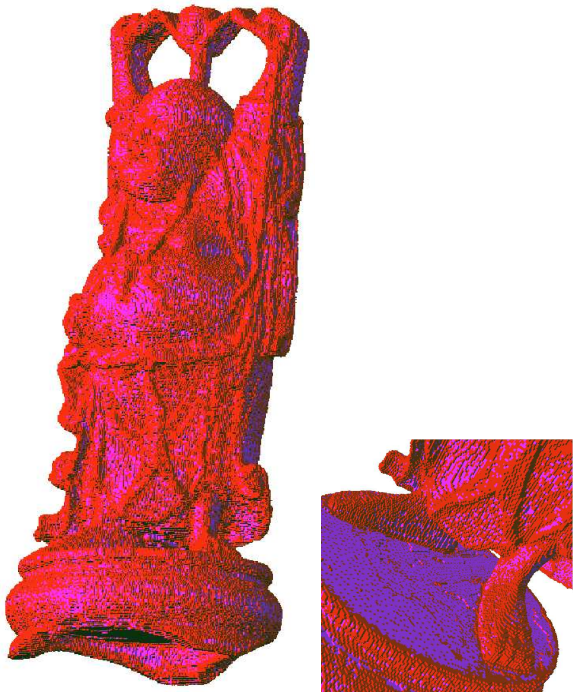


Figure 25: *Buddha discrete membrane and detail of the pedestal showing irregular sampling density*

Figure 26: *Buddha final surface: Front and back*

References

1. R. Mencl and H. Müller. Interpolation and approximation of surfaces from three-dimensional scattered data points. *Computer Graphics Forum (Proceedings of Eurographics)*, pages 51–67, July 1998.
2. T. K. Dey. Curve and surface reconstruction. In Goodman and O'Rourke, editors, *Handbook of Discrete and Computational Geometry*. CRC press, 2001.
3. M.E. Algorri and F. Schmitt. Surface reconstruction from unstructured 3d data. *Computer Graphics Forum*, 15(1):47–60, 1996.
4. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH. Computer Graphics Proceedings, Annual Conference Series*, 26(2):71–78, July 1992.
5. C. L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. *SIGGRAPH. Computer Graphics Proceedings, Annual Conference Series*, pages 109–118, 1995.
6. H. Edelsbrunner and E. Mücke. Three-dimensional alpha shapes. *ACM transactions on Graphics*, 13(1):43–72, 1994.
7. N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.
8. N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry & Applications*, 12(1):125–141, 2002.
9. R.C. Veltkamp. Boundaries through scattered points of unknown density. *Graphical Models and Image Processing*, 57(6):441–452, November 1995.
10. J.D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM transactions on Graphics*, 3(4):266–286, October 1984.
11. F. Isselhard, G. Brunnett, and T. Schreiber. Polyhedral reconstruction of 3d objects by tetrahedra removal. Technical Report 288/97, Fachbereich Informatik, University of Kaiserslautern, Germany, February 1997.
12. G. Roth and E. Wibowoo. An efficient volumetric method for building closed triangular meshes from 3d image and point data. *Graphics Interface*, pages 173–180, 1997.
13. J. D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbor interpolation of distance functions. *Proceedings of 16th Symposium on Computational Geometry*, pages 223–232, 2000.
14. H.K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. *Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision (VLSM 2001)*, page 194, July 2001.
15. S. Muraki. Volumetric shape description of range data using "blobby model". *SIGGRAPH. Computer Graphics Proceedings, Annual Conference Series*, 25(4):227–235, July 1991.
16. I. Takanashi, S. Muraki, A. Doi, and A. Kaufman. 3d active net for volume extraction. *Proc. SPIE*, (3298):184–193, 1998.
17. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, (1):321–331, 1988.
18. F. Bernardini, J. Mittleman, and H. Rushmeier. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999.
19. N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *SIGGRAPH. Computer Graphics Proceedings, Annual Conference Series*, pages 415–421, 1998.
20. M. Gopi, S. Krishnan, and C.T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Graphics Forum (Proceedings of Eurographics)*, 19(3), 2000.
21. J.-D. Boissonnat and S. Oudot. An effective condition for sampling surfaces with guarantees. *Proceedings 9th Annu. ACM Symposium on Solid Modeling and Applications*, pages 101–112, 2004.
22. C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. *IEEE Visualization'94*, pages 281–287, 1994.
23. A. Vinacua, I. Navazo, and P. Brunet. Octtrees meet splines. In G. Farin, H. Bieri, G. Brunett, and T. DeRose, editors, *Geometric Modelling, Computing [Suppl]*, volume 13, pages 225–233. Springer-Verlag, 1998.
24. J. Esteve, P. Brunet, and A. Vinacua. Piecewise algebraic surface computation and fairing from a discrete model. Technical Report LSI-04, Dept. L.S.I., Universitat Politècnica de Catalunya. A more extense version is pending publication and is available as a preprint from http://www.lsi.upc.edu/∼jesteve/algebraic-surface-fairing.pdf, 2004.
25. J. Esteve, P. Brunet, and A. Vinacua. Multiresolution for algebraic curves and surfaces using wavelets. *Computer Graphics Forum*, 20(1):47–58, 2001.
26. W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.