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

Jordi Esteve Pere Brunet Alvar Vinacua

e-mail: [jesteve, brunet, alvar]@lsi.upc.es
Universitat Politcnica 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. Excluding the initial data points, any other previous information is not used (as, for example, the topological relations amongst the points or the normal surface at the data points). 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 *genus* ≥ 1 and closed surfaces with disconnected shells.

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".

Being impossible to detail all published papers dealing with this problem, the reader can consult some of the existing State-of-the-Art reports like [1]. Some papers are mentioned below. They are classified in four blocks according to the taxonomy used in [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 ([2] and [3]), others decompose the space in tetrahedrals ([4] and α-shapes[5]). Examples of volume oriented algorithms are [6], [7] and [8] (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 between 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 [3], [9] and [4].
- **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" [10] and the mass and spring model built from a triangular mesh [2]. Also the algorithms based on 3D snakes (energy-minimizing spline which is attracted toward the initial point set) can be classified in this category ([11] and [12]).
- **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 his neighborhood enlarging the surface. This is the idea of the surface oriented algorithm of Boissonnat [7].

This taxonomy has several drawbacks. Some algorithms can be included in more that one category. For example, some spatial subdivision algorithms use a distance function to find out which cells are stabbed by the surface ([3], [4]). Also 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 ([2], [3]), tri-variate implicit Bernstein-Bezier patches [4] or polytopes (sets of points, edges, triangles and tetrahedrals not necessarily defining a closed and regular surface) in the α-shapes algorithm [5].

Our algorithm can be included in the first category: 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.

The main goal is the construction of a closed surface that approximates a non uniform data point set in the 3D space which is known to approximate a closed solid (Figure 19(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 have points in its inside (hard voxels) or not (soft voxels) (Figure 19(b)).

2 Obtaining a discrete closed membrane: a set of 6-connected (face-connected) voxels set 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)). Later 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 can not 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 lightly to reduce the local curvature (Figure 18).

4 Construction of the final surface from the discrete membrane obtained in the previous step (see Figure 20).

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.

In some aspects our algorithm is similar to the algorithm of α-shapes [5]. 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. Polytopes are obtained (sets of points, edges, triangles and tetrahedrals not necessarily defining a closed and regular surface). 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 square plate vs. a sphere of radius α) and that our method performs a sequence of contractions with plates 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.

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 as maximum to a distance d from it, being d the diagonal of the used voxel. 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 voxel d. The surface approximating the data points with a tolerance d is named $d - error$ 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, it exists a margin of error in the acquired data. 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 an 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.

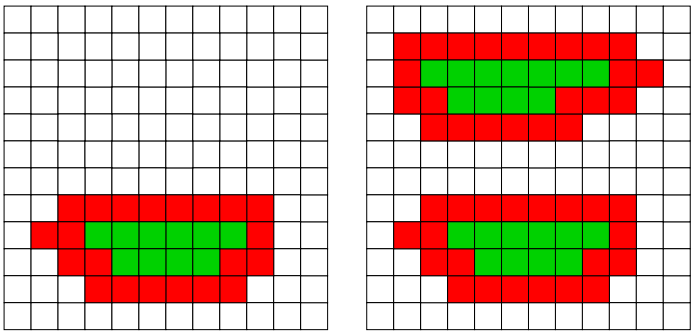
In Section 2 the main part of the algorithm is discussed: the voxelization of the space and the membrane shrinking. Section 3 describes the relaxation of the membrane and section 4 the construction of the final surface. The results are explained in section 5 and the final conclusions are pointed out in section 6.

2 The discrete membrane shrinking

2.1 Basic concepts involved in our algorithm

The algorithm works in a spatial division (**voxelization**) of a parallelepiped containing the cloud of points with cubes of the same size. The cubes of the voxelization are named voxels. 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 [1]. Figure 1(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 1: Examples of 2D Discrete Membranes

To deal with objects with disconnected shells is allowed to have several membranes inside the voxelization. Therefore we will work with a set of isolated DM (named Discrete Membrane Set or DMS). Any voxel of the voxelization can be classified according to:

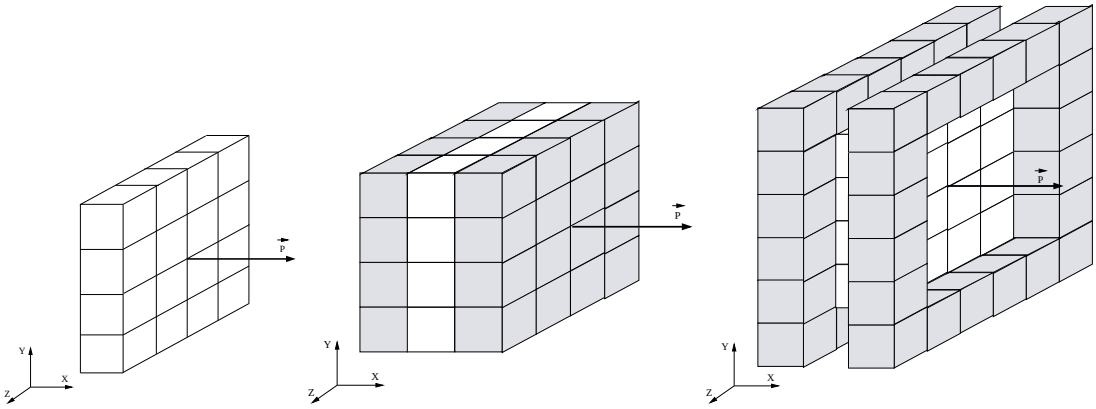
- HARDNESS: Hard, soft.
- POSITION: Inside (the voxel is inside one DM), Outside (the voxel is outside to all DM), Boundary (the voxel belongs to a DM).

[1] 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 1(b) illustrates a 2D voxelization with 2 DM: 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 understanding 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 2(a).



(a) Plate with its orientation

(b) Front and Back Side

(c) Lateral Front and Back Side

Figure 2: Plate of size $n = 4$

The **Front Side** (Back Side) of a plate of size n are the $n \times n$ voxels localized in front (behind) of the plate according to its orientation. See Figure 2(b).

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

The **Lateral Front Side** (Lateral Back Side) of a plate of size n are the $4 \times (n + 1)$ voxels localized around the front side (back side) of the plate. See Figure 2(c).

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

2.2 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:

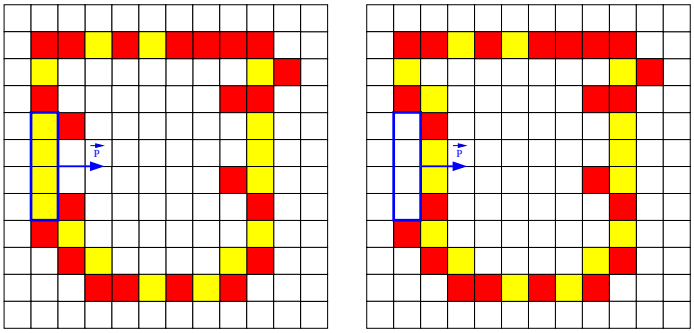
1 It is composed uniquely of one or more boundary soft voxels and outside voxels (i. e. does not contain hard voxels or inside voxels).

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 ($n \times n + 8 \times (n + 1)$ voxels) that were inside are converted to boundary voxels preserving their hardness.



(a) Before (b) After

Figure 3: CONTRACTION operation with a plate of size $n = 4$. The hard voxels are painted in red and the boundary soft voxels in yellow.

Figure 3 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 material 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 ($FinalDM = InitialDM - plate + front + lateral + lateralfront$).

3 The hard voxels never are converted to outside ones, since only the boundary soft voxels overlapped with the plate are converted to outside ones.

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 for several connected components. See Subsection 2.6 and Figure 12 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 material volume and yields an 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 material 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's 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's sizes n used by the algorithm allows to know the current generation. We define 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's size n caused the conversion from boundary voxel to outside voxel). See in Figure 4 how the outside voxels are labeled with the 1, 2, 3 and 4 generations that correspond to the plate's sizes $n = 12$, $n = 6$, $n = 3$ and $n = 2$ respectively.

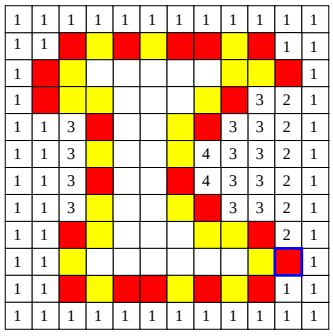


Figure 4: Labeling the outside voxels with its generation

The main differences between our algorithm and α-shapes algorithm[5] are the object used to perform the contraction/elimination (a square plate of size n vs. a sphere of radius α) and that our method performs a sequence of contractions with plates 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.

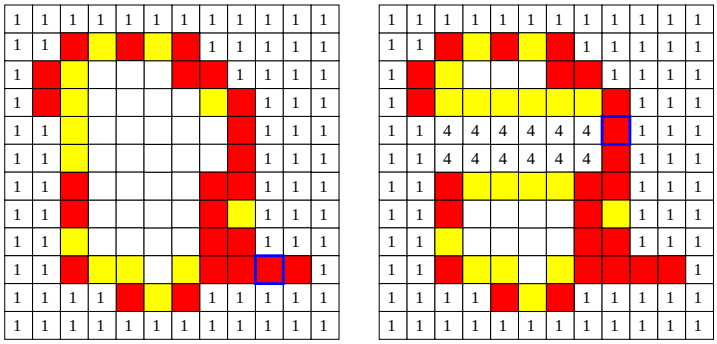
An incursion test must be defined to detect when the interior of a DM is invaded by a plate of size n. This incursion test detects when the plate of size n reaches the interior face of the DM boundary (see Figure 7). The incursion test definition is based on the Local-Arc Connectivity concept.

Local-Arc Connectivity: Two voxels in the 26-neighborhood[2] 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 bottom right of Figure 4. The outside voxels localized above and below of the highlighted voxel are local-arc connected.

Incursion Test: An incursion is detected when two outside voxels of different generations

[2]The 26-neighborhood of a voxel V is the set of 26 voxels that share a vertex, edge or face with V

at the opposed faces of a boundary hard voxel are not local-arc connected between them. Observe Figure 5(a): No incursion has happened here in the highlighted hard voxel since the voxels localized at the opposed faces, though not local-arc connected, have the same generation. Instead, in Figure 5(b), an incursion has been detected since the voxels localized at the opposed faces have different generation.



(a) Not (b) Yes

Figure 5: 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.

2.3 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 parallelepiped sides 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 19(b) the hard voxels of the voxelization of the data points of Figure 19(a) have been displayed.

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

$$l \approx \sqrt{\frac{2\left(l_x l_y + l_y l_z + l_z l_x\right)}{np}} \tag{1}$$

being np the number of the initial data points. This expression considers that the data point set defines a surface of similar extension as the 6 faces of the parallelepiped that contains it and its density is more or less uniform. In all examples shown in Section 5 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 $\frac{voxels\ containing\ 2\ or\ more\ points}{voxels\ containing\ 1\ point}$ after the voxelization and if its value is too high we can recalculate the voxelization decreasing the edge's length of the voxel l.

The second step constructs the initial discrete membrane with the voxels belonging to the exterior faces of the voxelization. Thereafter 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's 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.

Figure 6 illustrates the two previous points in a main algorithm.

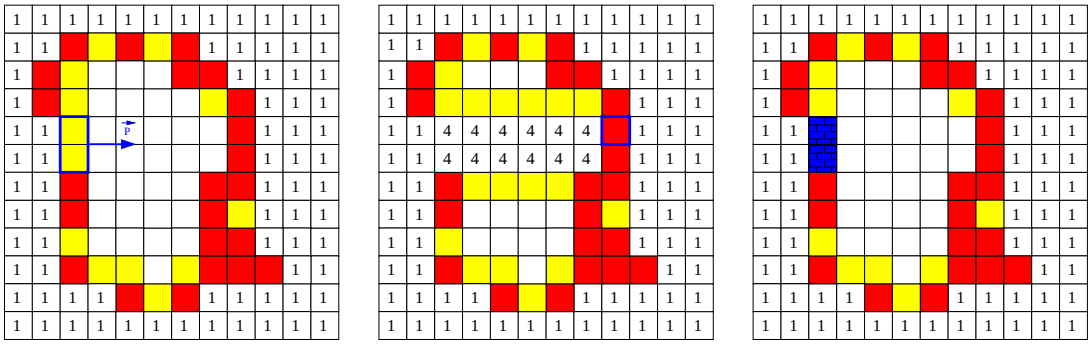
```
main() {
  vox = Voxelization( cloud_of_points );
  DMS = InitialDiscreteMembrane( vox );
  n   = max( vox.size.x, vox.size.y, vox.size.z );
  do {
    n = (n+1)/2;
    FindContractionPlace( DMS, n, placeOk, place );
    while (placeOk) {
      incursion = FALSE;
      NewStack( ContractionStack );
      RecursiveContraction( DMS, n, place, ContractionStack, incursion );
      FindContractionPlace( DMS, n, placeOk, place );
    }
  } while (n > 1);
}
```

Figure 6: Main algorithm scheme

The algorithm results depend on the choice of the initial plate's size and how it is reduced progressively. The best solution, though with more computational cost, is to start with a plate's size the biggest of the 3 voxelization sizes $n = max(nx, ny, nz)$ and decreasing the plate's 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 $n = (n + 1)/2$. This obviously reduces considerably the algorithm run-time. The results shown in Section 5 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 7(b)), the UNDO CONTRACTION operation is applied as many times as CONTRACTION operations were performed. Thereafter a FREEZING operation is applied (Figure 7(c)). This cancels an undesired expansion of the plate inside the solid.



(a) Before of initiate the shrinking

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

(c) Backtracking and freezing

Figure 7: Incursion detection

Figure 8 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` action.

2.4 Algorithm properties

In this section we shall establish the following four fundamental properties of our algorithm:

- The algorithm always finishes.
- The result is a DMS (the voxels are face-connected and they divide the remaining voxels in outside and inside).
- The boundary voxels that has an outside voxel in its 6-neighborhood[3] are necessarily hard or frozen soft voxels.
- All outside voxels are soft.

The algorithm always finishes because it decreases the material volume and the number of voxels of the initial voxelization is finite. The material volume is reduced because a recursive sequence of CONTRACTION operations is applied which result can be:

[3]The 6-neighborhood of a voxel V is the set of 6 voxels sharing a face with V

```
RecursiveContraction( DMS, n, place, ContractionStack, incursion ) {

  if (incursion) return;
  c = CONTRACTION( DMS, n, place );
  PushStack( ContractionStack, c );
  if (IncursionTest( c )) {
    while (!EmptyStack( ContractionStack ) {
      c = TopStack( ContractionStack );
      PopStack( ContractionStack );
      UNDO_CONTRACTION( DMS, n, c );
    }
    FREEZING( DMS, n, place );
    incursion = TRUE;
    return;
  }

  for (direction={front, up, down, right, left}) {
    place2 = place + direction;
    if ContractionOk( DMS, n, place2 ) {
      RecursiveContraction( DMS, n, place2, ContractionStack, incursion );
    }
  }
}
```

Figure 8: Scheme of the contraction recursive action

1 An incursion is never detected. It would suppose a reduction of the material volume since only CONTRACTION operations have been applied.

2 An incursion is detected. This means the maintenance of the material volume, since the sequence of UNDO CONTRACTION operations recover the material volume that the sequence of CONTRACTION operations had reduced and the FREEZING operation keeps the material volume. This sequence of CONTRACTION, UNDO CONTRACTION and FREEZING operations is never again repeated because of converting soft voxels to frozen soft voxels by the FREEZING operation.

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 2.6). Therefore each DM of the final set is face-connected and divides the remaining voxels in outside and inside.

The boundary voxels having an outside voxel in their 6-neighborhood are hard or frozen soft voxels, never regular soft voxels. The reason is the algorithm works with plates of diminishing size arriving to a size of $n = 1$. When the plate's 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. In other case a backtracking and a freezing would have happened

converting the soft voxel to a frozen soft voxel.

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.

2.5 Incursion detection

The surface reconstruction from a data point set is a complex problem. Also, 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 when 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 with $genus \geq 1$).

To observe how the algorithm works and see the consequences of the incursion detection we will simplify the complexity problem.

Property 1. If these two propositions are true:

1 The data point density is homogeneous obtaining in the voxelization step a set of hard voxels forming themselves a DM (the hard voxels are face-connected and divide the remaining voxels in inside and outside).

2 All hard voxels have at least one inside voxel in their 26-neighborhood.

Then no incursion arises in the execution of our algorithm.

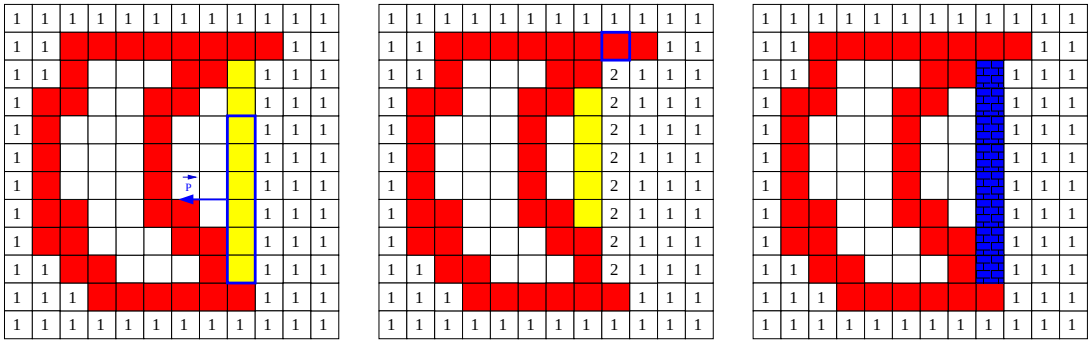
Proof: The case of having two outside voxels at the opposed faces of a hard voxel is not possible. In the 26-neighborhood of the hard voxel there must be at least an inside voxel and this inside voxel must be separated from the outside voxels by boundary voxels.

Since an incursion is never detected, the algorithm will contract the DM with plates of diminishing size arriving to a size of $n = 1$ without doing any UNDO CONTRACTION operation. Therefore the DM is shrunk until it coincides with the set of existing hard voxels. Under the precondition of the property 1 we can affirm that the algorithm always produces the expected result since, as the hard voxels form a face-connected closed set, it was not necessary to perform any UNDO CONTRACTION operation during the algorithm.

If the second precondition of the property 1 is eliminated there can be situations where incursions are detected and UNDO CONTRACTION operations are applied. All these operations are not necessary because of having a set of face-connected closed hard voxels. Figures 9 and 10 show some of these situations: Protuberances a voxel thick and DM pieces overlapped. In these cases the final result and the expected result can differ.

The properties of the incursion test are:

- **Property 2**. Acceptation of protuberances composed by a unique voxel (checking if there is local-arc connectivity between the two outside voxels localized at the opposed faces of a boundary hard voxel). Observe the highlighted hard voxel localized at left of Figure 11: An incursion is not detected because there is local-arc connectivity between the two outside voxels localized at the opposed faces of the highlighted hard voxel.
- **Property 3**. Acceptation of some isolated protuberances (checking if the two outside voxels localized at the opposed faces of a boundary hard voxel have different gener-

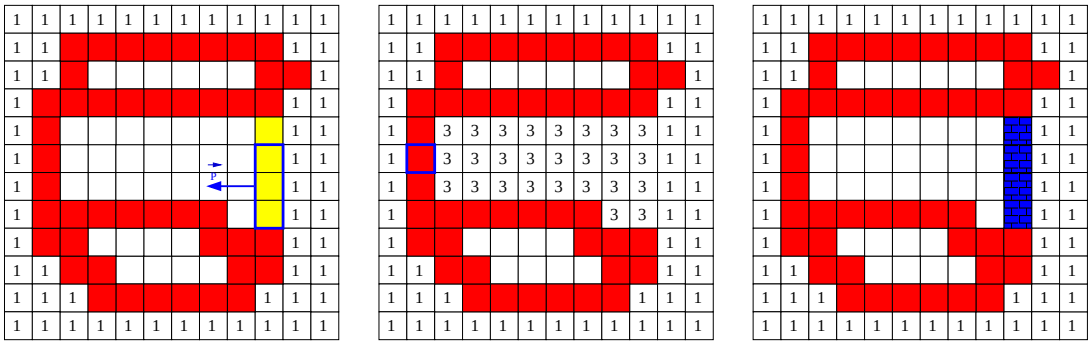


(a) Before of starting the shrinking

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

(c) Backtracking and freezing

Figure 9: Undesirable incursion: Protuberance a voxel thick



(a) Before of starting the shrinking

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

(c) Backtracking and freezing

Figure 10: Undesirable incursion: DM pieces overlapped

ations). See the highlighted hard voxel at bottom right of Figure 11: An incursion has not been detected because the two outside voxels sharing the opposed faces of the highlighted hard voxel have the same generation.

These two properties allow the incursion test to accept the more common protuberances. 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. Some of the reasons of detecting an incursion are the protuberances a voxel thick (Figure 9), the overlapping of DM pieces at some place (Figure 10) or the contraction at zones of low data point density where the plate can be introduced towards the interior of the surface (Figure 7). The question is the difficulty to distinguish if a protuberance a voxel thick is due to the own shape of the surface to reconstruct or to a low data point density nearby.

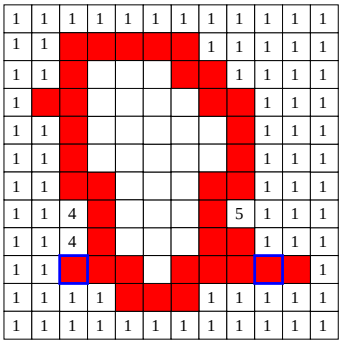
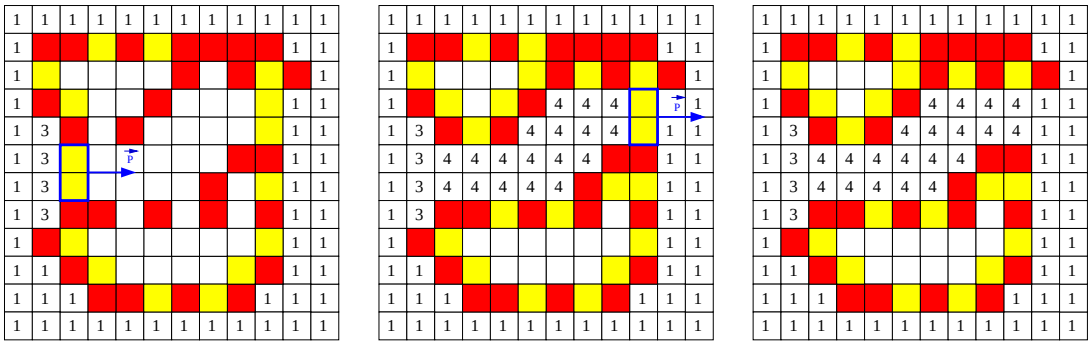


Figure 11: Discrete membrane with protuberances a voxel thick

2.6 Mitosis and increase of the genus

If the incursion test is applied in the boundary hard voxels but not in the soft ones, it will be possible the removal of DM pieces formed only by soft voxels. 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 12 illustrates a mitosis and Figure 21(c),(d) (at the handle) an increase of the genus.



(a) Before of starting the shrinking

(b) Shrinking with $n = 2$

(c) End of the shrinking

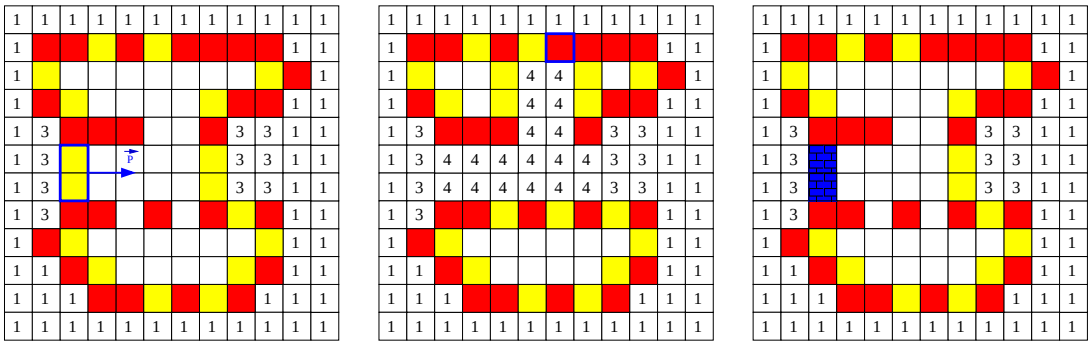
Figure 12: Shrinking in a tunnel of size 2 producing a mitosis

A **Tunnel** of size n exists in the DMS when, applying a sequence of CONTRACTION operations with a plate of size n from a particular location, two or more sets of boundary soft voxels are eliminated and an incursion is not detected. The two or more sets of boundary soft voxels must not be face-connected before of applying the sequence of CONTRACTION operations. In the example of Figure 12(a) there is a tunnel of size 2: The set of highlighted boundary soft voxels in 12(a) and the highlighted in 12(b) are not face-connected and a sequence of CONTRACTION operations with a plate of size 2 has eliminated them without detecting an incursion.

Property 4. The algorithm only is capable to produce mitosis or increases of the genus when the DMS has a tunnel of a certain size n.

We will study two examples that the DMS does not have tunnels due to the detection of an incursion.

- The data point density (hard voxels density) around of the possible tunnel is not sufficiently big in relation to the size of the tunnel. See Figure 13.
- At the ends of the tunnel there are one-voxel thick protuberances. Figure 14 shows one of these cases.

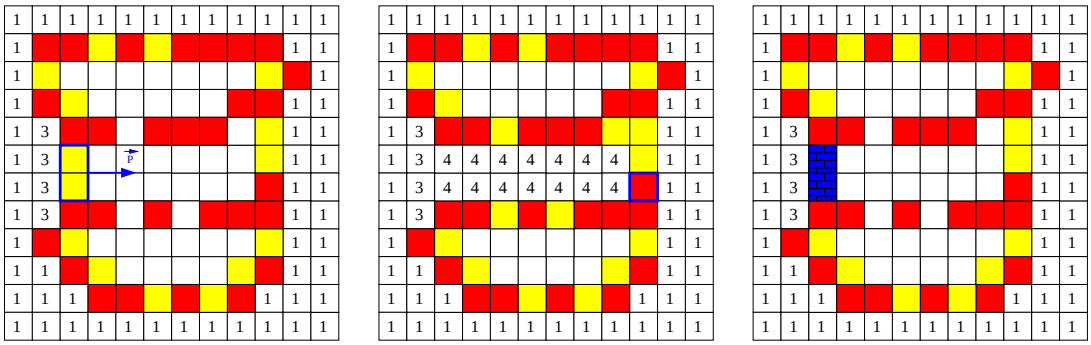


(a) Before of starting the shrinking

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

(c) Backtracking and freezing

Figure 13: Mitosis not performed due to a low data point density



(a) Before of starting the shrinking

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

(c) Backtracking and freezing

Figure 14: Mitosis not performed due to a one-voxel thick protuberance

The last CONTRACTION operation causing a mitosis/increase of the genus (see the evolution of Figure 12(b) to the 12(c)) also produces a reduction of the material volume: Though the number of inside voxels is the same, the number of boundary voxels is reduced.

3 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 15).

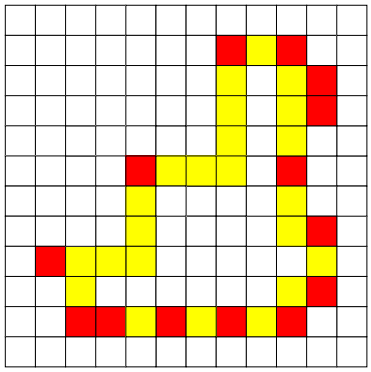


Figure 15: Planar regions in the final DMS

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 neighbors 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: The discrete local curvature of a boundary voxel is 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 16 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]).

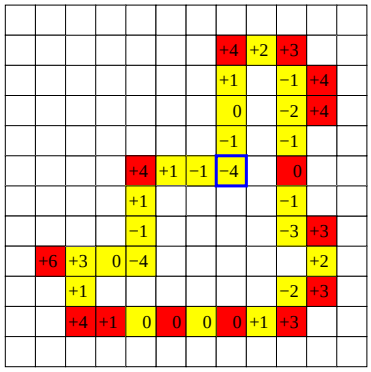


Figure 16: Discrete Local Curvature (DLC) of the boundary voxels

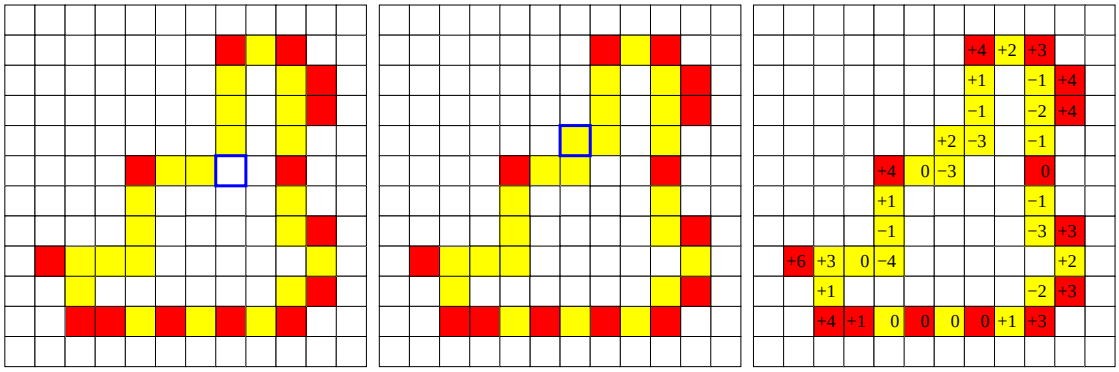
The relaxing process first selects the boundary soft voxels V with $DLC(V) \leq -13$. It starts to process the most negative ones and finishes with those soft voxels having $DLC(V) = -13$. To each selected voxel V the following steps are applied:

1 The voxel V is converted to inside voxel (Figure 17(a)).

2 The outside voxels of the 26-neighborhood of V are converted to boundary (Figure 17(b)).

3 Recalculate the DLC on the boundary voxels in the 124-neighborhood[4] of the voxel V (Figure 17(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 17 illustrates the steps described previously applied on the highlighted voxel of Figure 16.



(a) Elimination of the selected voxel

(b) Outside to boundary voxel conversion

(c) DLC recalculation

Figure 17: 2D Relaxing

Afterwards the simetrical above steps are applied 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 co-ordinates planes or at 45 degrees. Moreover it causes the relaxing process performed on voxels V with $DLC(V) > 0$ to destroy the previous relaxation done on voxels $V\prime$ with $DLC(V\prime) < 0$. Applying the relaxing steps described previously on some voxel configurations with $DLC(V) = 12$ generates other boundary voxel $V\prime$ with $DLC(V\prime) = -12$ and vice-versa.

[4]The 124-neighborhood of a voxel V is the set of the $5 \times 5 \times 5 - 1$ neighbors voxels of V localizing V at the center

Figure 18 shows the teapot of Figure 21(f) after applying the relaxing process on the boundary voxels.

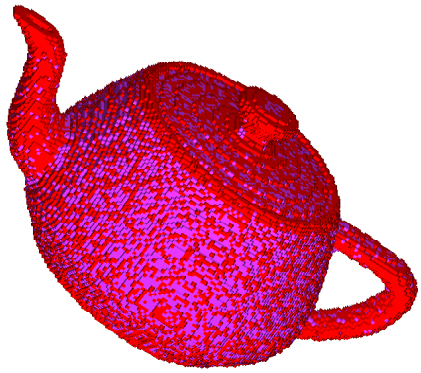


Figure 18: 3D Relaxing

4 Final surface construction

The last step must build a surface stabbing the DMS obtained in previous steps. For example, the discrete marching cubes algorithm [13] can be used to obtain a mesh of triangles, the algorithm described in [14] 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.

We have improved the conversion to a piecewise algebraic surface (see [15]). The construction of the piecewise algebraic surface stabbing the DMS uses anti-aliasing techniques to achieve smoother surfaces and takes advantage of the knowledge of the initial data points lie in the hard voxel to approximate better the final surface. First an initial isosurface is calculated setting positive or negative weights to the vertexes of the DMS depending on if the neighbor voxels are exterior or interior. Later a battery of filters ([15]) are applied to fair the surface and to fit it to 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 [16].

Figure 20 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 [17].

5 Results

The first example is the approximation of a data point set obtained from a 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 19). There are 19143 hard voxels in the voxelization since 16767 points have coincided in voxels where already there were 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's size (number of different locations in the DMS where a recursive sequence of CONTRACTION operations has been started to apply) and the number of performed backtrackings (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 DM	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 17202 hard voxels (painted in red) and 27491 soft 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 (1.9%). Most of the data points (98.1%) 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 locations more nearby. The run-time of the whole shrinking process is 5 minutes and 25 seconds in a PC with a 350Mhz AMD-K6-2 CPU and 128Mb of main memory. Figure 20 shows the final surface obtained from the DM with two different algorithms.

The second example uses a set of 100461 points modeling a 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 22(a)) is composed of 64130 hard voxels and 68869 soft voxels. Inside the DM 1228510 soft voxels and 1766 hard voxels (a 2.7% of the total of hard voxels) remain. The final surface, a cubic Bspline isosurface displayed with a raycasting algorithm, appears in Figure 22(b). The table 2 shows the evolution of the DM shrinking of the bird.

The third example uses a set of 56306 points modeling a dinosaur. The voxelization has a size

Table 2: Bird gradual shrinking

Size of the plate	Number soft voxels in the DM	Number of started contractions	Number of backtrackings made
76	103134	8326	0
38	123733	1162	0
19	71648	1423	1
10	46274	842	9
5	33818	445	19
3	27013	310	31
2	24567	597	47
1	22327	13181	12307

of $201 \times 170 \times 70$ with 31088 hard voxels (25218 points are within hard voxels that already contained other data points). The final DM (Figure 24(a)) is composed of 25269 hard voxels and 17052 soft voxels. Inside the DM 90855 soft voxels and 22 hard voxels (a 0.09% of the total of hard voxels) remain. The run-time is 9 minutes and 2 seconds. The final surface, a cubic Bspline isosurface displayed with a raycasting algorithm, appears in Figure 24(b). The table 3 shows the evolution of the DM shrinking of the dinosaur.

Table 3: Dinosaur gradual shrinking

Size of the plate	Number soft voxels in the DM	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

Finally Figure 23 shows the DM and the final surface obtained from a cloud of points modeling an oil pump. All these examples illustrate a correct and robust operation of the gradual DM shrinking algorithm.

6 Conclusions

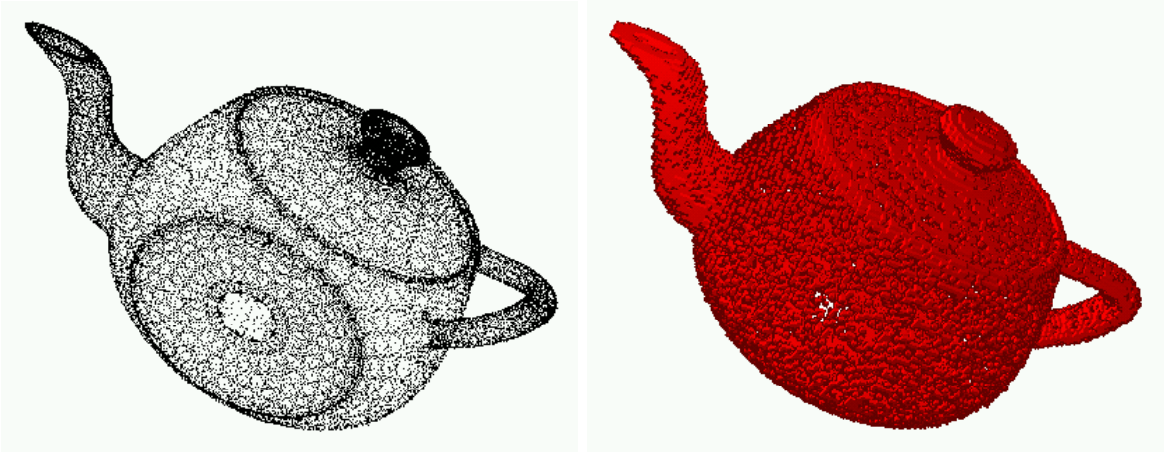
An algorithm that obtains one or more closed surfaces approximating a data point set in the 3D space has been presented. The algorithm does not require to know the topological relations among the points or other additional information. The obtained surface approximates and does not stab exactly the points: 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 $genus \geq 1$ 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 algorithm success 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.

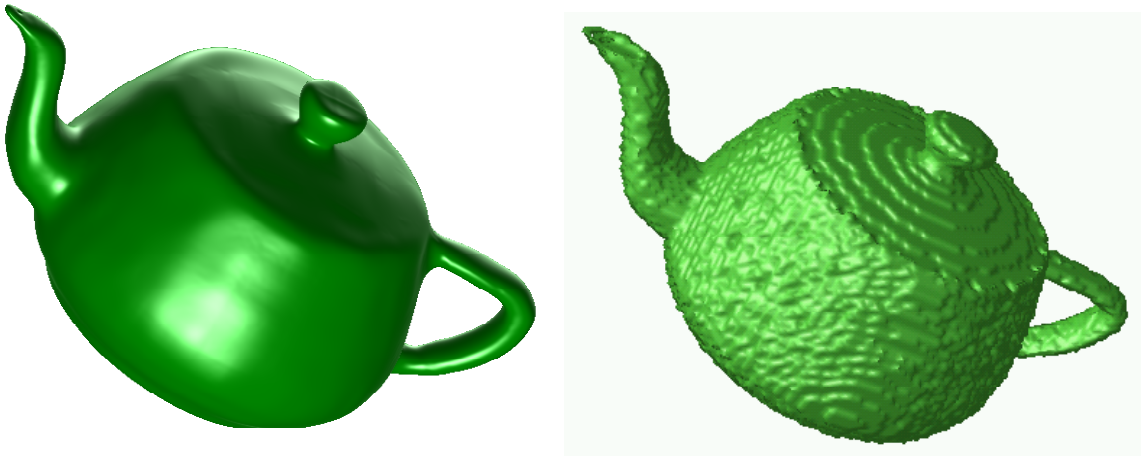
The construction of a piecewise algebraic surface stabbing the final discrete membrane includes a constrained fairing algorithm [15] to achieve smoother surfaces and to approximate better the final surface to the initial data points.



(a) Initial data point set

(b) Voxelization

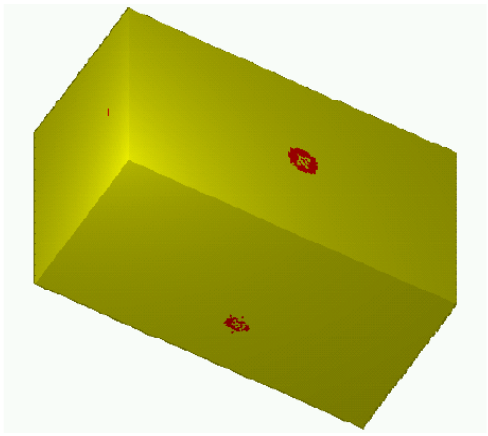
Figure 19: Voxelization of a cloud of points



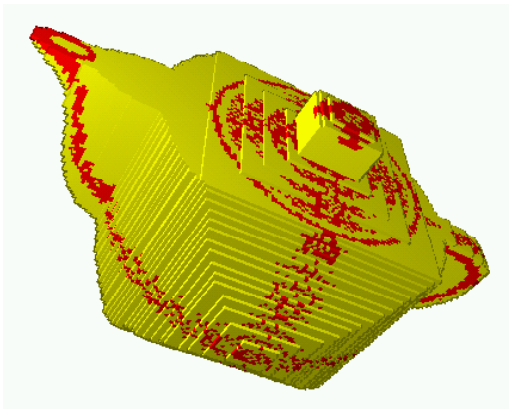
(a) Cubic Bspline isosurface obtained by the algorithm described in [14]

(b) Mesh of triangles obtained by marching-cubes [17] (displayed with a Gouraud smoothing)

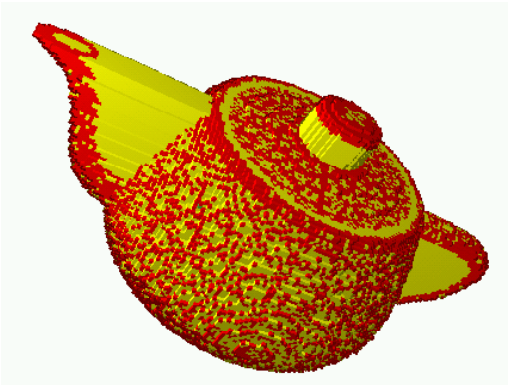
Figure 20: Construction of the surface stabbing the final DMS



(a) Initial discrete membrane

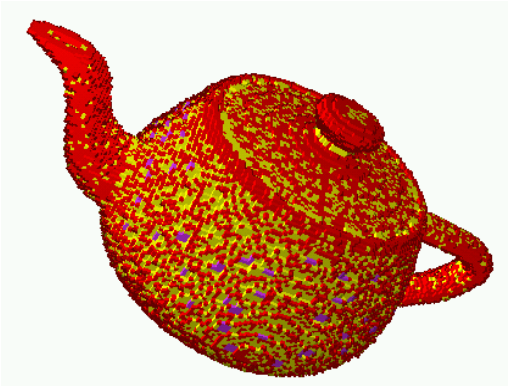


(b) Silhouette shrinking

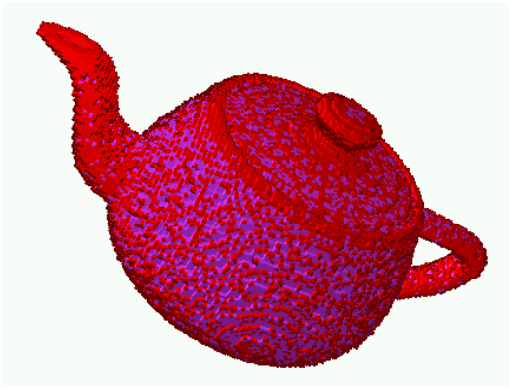


(c) Shrinking with $n = 41$

(d) Shrinking with $n = 6$



(e) Shrinking with $n = 3$



(f) $n = 1$: Final discrete membrane

Figure 21: Discrete membrane shrinking

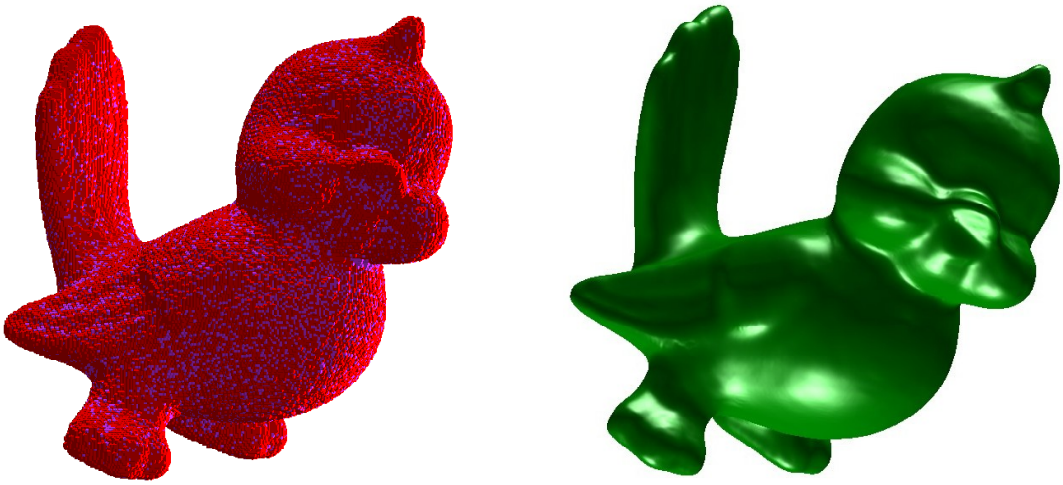


Figure 22: Bird: Discrete membrane and final surface

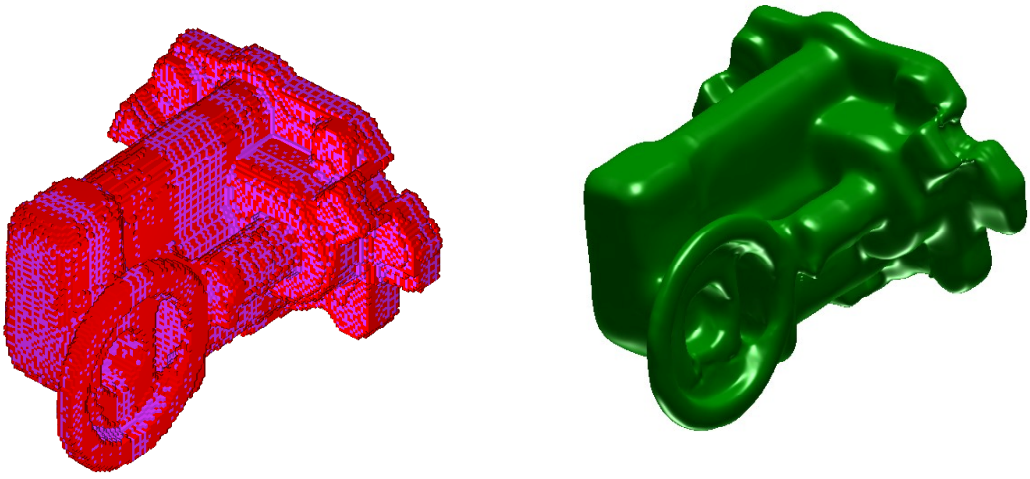


Figure 23: Oil pump: Discrete membrane and final surface

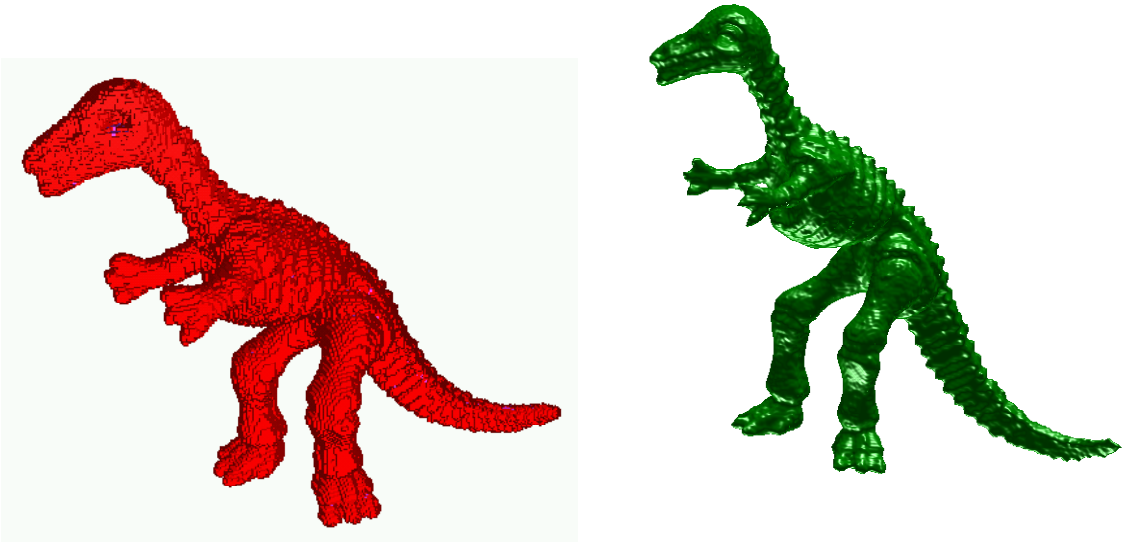


Figure 24: Dinosaur: Discrete membrane and final surface

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] M.E. Algorri and F. Schmitt. Surface reconstruction from unstructured 3d data. *Computer Graphics Forum*, 15(1):47–60, 1996.

[3] 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.

[4] 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.

[5] H. Edelsbrunner and E. Mücke. Three-dimensional alpha shapes. *ACM transactions on Graphics*, 13(1):43–72, 1994.

[6] R.C. Veltkamp. Boundaries through scattered points of unknown density. *Graphical Models and Image Processing*, 57(6):441–452, November 1995.

[7] J.D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM transactions on Graphics*, 3(4):266–286, October 1984.

[8] 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.

[9] 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.

[10] S. Muraki. Volumetric shape description of range data using "blobby model". *SIGGRAPH. Computer Graphics Proceedings, Annual Conference Series*, 25(4):227–235, July 1991.

[11] I. Takanashi, S. Muraki, A. Doi, and A. Kaufman. 3d active net for volume extraction. *Proc. SPIE*, (3298):184–193, 1998.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, (1):321–331, 1988.

[13] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. *IEEE Visualization'94*, pages 281–287, 1994.

[14] 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.

[15] J. Esteve, P. Brunet, and A. Vinacua. Fairing and fitting for algebraic curves and surfaces. Technical Report LSI-03, Dept. L.S.I., Universitat Politècnica de Catalunya, 2003.

[16] J. Esteve, P. Brunet, and A. Vinacua. Multiresolution for algebraic curves and surfaces using wavelets. *Computer Graphics Forum*, 20(1):47–58, 2001.

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