Reducing Artifacts in Surface Meshes Extracted from Binary Volumes

Ragnar Bade
Otto-von-Guericke-University
Universitaetsplatz 2
D-39106 Magdeburg, Germany
bade@isg.cs.uni-magdeburg.de

Olaf Konrad
MeVis Research
Universitaetsallee 29
D-28359 Bremen, Germany
okonrad@mevis.de

Bernhard Preim
Otto-von-Guericke-University
Universitaetsplatz 2
D-39106 Magdeburg, Germany
preim@isg.cs.uni-magdeburg.de

ABSTRACT

We present a mesh filtering method for surfaces extracted from binary volume data which guarantees a smooth and correct representation of the original binary sampled surface, even if the original volume data is inaccessible or unknown. This method reduces the typical block and staircase artifacts but adheres to the underlying binary volume data yielding an accurate and smooth representation. The proposed method is closest to the technique of *Constrained Elastic Surface Nets (CESN)*. CESN is a specialized surface extraction method with a subsequent iterative smoothing process, which uses the binary input data as a set of constraints. In contrast to CESN, our method processes surface meshes extracted by means of *Marching Cubes* and does not require the binary volume. It acts directly and solely on the surface mesh and is thus feasible even for surface meshes of inaccessible or unknown volume data. This is possible by reconstructing information concerning the binary volume from artifacts in the extracted mesh and applying a relaxation method constrained to the reconstructed information.

Keywords

Surface reconstruction, mesh smoothing, staircase artifacts, artifact reduction, segmented data, Marching Cubes.

1. INTRODUCTION

The extraction and visualization of surface models from medical volume data (e.g. CT, MRI) is supported by any clinical workstation and medical visualization software. For efficient and clear visualization, surface models are often extracted from medical volume data. Iso-surfaces, extracted from binary segmented volume data, suffer from aliasing and staircase artifacts. The human visual system is very sensitive to such discontinuities, since they normally represent salient features for object detection and classification. To improve the quality of extracted iso-surfaces, three strategies exist: (1) filtering of the binary volume (at the voxel level), (2) applying an extended extraction mechanism (combining voxel and mesh level), and (3) filtering of the extracted surface mesh (see Figure 1). While the first two strategies enable artifact reduction constrained to the volume data, methods following the third strategy did not yet address such volume data constraints. Consequently, mesh filtering approaches can not guarantee that a resulting mesh is a correct representation of the volume data. But if the underlying volume data is inaccessible or unknown, mesh filtering is the only possible approach.

This paper presents a surface mesh filtering approach that enables artifact reduction in iso-surface meshes extracted from binary volumes. The filtering process is constrained by information about the volume data. For this purpose, we reconstruct this information about the volume data from the iso-surface mesh. This strategy enables artifact reduction in iso-surface meshes constrained to the underling volume data, independent of the presence of the original volume data. The presented work splits up into two major parts. The first part deals with the extraction of information about the original volume data from a given iso-surface mesh. The second part adopts the technique of *Constrained Elastic Surface Nets (CESN)* [Gib98] for constrained smoothing of iso-surface meshes extracted by *Marching Cubes (MC)*.

In Section 2, we discuss relevant previous work in the area of artifact reduction in iso-surfaces. Then we present our method for reconstructing volume data information from iso-surface meshes in Section 3 and smoothing of these meshes constrained to the volume

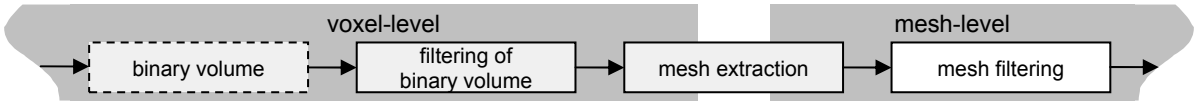


Figure 1: Surface extraction pipeline: Reduction of artifacts in iso-surfaces extracted from binary volume data is possible on the voxel-level (filtering of binary image), a combination of voxel- and mesh-level (as part of the mesh extraction process) as well as on the mesh-level (filtering of the mesh) which is the focus of this paper.

data in Section 4. In Section 5, we present applications and results of the proposed method and compare them with the results of previous work. Section 6 concludes and finally discusses the work presented.

2. PREVIOUS WORK

The most prevalent high-quality artifact reduction method at the voxel level is smoothing of the binary volume by means of a level-set method as described in [Whi00]. This method has several advantages: the volume is directly processed, no explicit surface representation is needed, and no tuning of parameters is required. Complex implementation and relatively long calculation times are drawbacks of the level-set method. Since this method is confined to the voxel-level, it is appropriate for volume rendering, but if surface rendering is the goal, extracted surfaces can still suffer from artifacts due to the subsequent extraction process.

Relevant mesh-extraction methods that include artifact reduction are *Precise Marching Cubes* [All98], *Constrained Elastic Surface Nets (CESN)* [Gib98], and *Dual Marching Cubes (DualMC)* [Nie04].

Precise Marching Cubes [All98] extended the original *Marching Cubes* (MC) [Lor87] by trilinear interpolation and adaptive error-controlled refinement of surface patches inside surface-containing cells. As a result, the precision as well as the smoothness of the extracted iso-surfaces could be improved. Unfortunately, this method creates a lot more triangles, requires much longer calculation time than MC and is not well suited for binary volumes.

Constrained Elastic Surface Nets (CESN) [Gib98] is the mesh extraction method closest to our work. It is dedicated to visualize binary volume data smoothly and precisely. In contrast to MC, [Gib98] uses an extraction scheme that builds a surface by connecting the centers of all cells that contain the surface to quadrilateral patches. In a second step, this initial surface is iteratively relaxed while all vertices are constrained to remain in their original surface cell. This method creates a well smoothed surface representation of the original binary volume. A comparable method is proposed by [Nie04]. His *Dual Marching Cubes (DualMC)* approach also connects adjacent surface cells to quadrilateral surface patches and then iteratively relaxes the extracted surface constrained to the binary volume. The major differences are a smoother initial surface extracted by means of an adapted extraction method and another relaxation scheme compared to CESN. While the methods by [Whi00], [Nie04], and [Gib98] enable appropriate artifact reduction, all methods discussed so far require the original binary volume data.

On the mesh-level, numerous mesh filtering and smoothing approaches exist, ranging from simple *Laplacian* filters to more complex *Mean Curvature Flow* and further advanced anisotropic filtering approaches (for example see [Baj03], [Des99], [Tau95]). Despite the diversity, all methods aim at noise reduction. Advanced methods additionally attempt to preserve salient features and edges. Reduction of staircase artifacts is usually not a goal of surface mesh smoothing approaches. Nevertheless, to smooth surfaces suffering from such artifacts all edge-preserving filtering methods are not appropriate. Staircase and block artifacts would be interpreted as salient features and preserved by those methods. Furthermore, smoothing methods that yield shrinkage and deformation of the mesh are not appropriate for anatomical and pathological structures.

[Tau95] introduced a signal processing driven two-stage *Laplacian* mesh filter *(λ/μ-Filter*) that first smoothes the mesh with a positive smoothing factor and then with a negative one. This strategy avoids shrinkage and [Tau95] showed that it behaves like a low-pass filter, if a large number of iterations is applied. Applying this filter can significantly reduce aliasing artifacts [Tau95]. A similar method has been presented by [VMM99], where in the second stage all vertices are moved back towards a linear combination of their original location and the inverse displacement of their neighbors. Unfortunately, in practice, finding the right parameters to smooth a specific object is tedious. While well chosen parameters can also yield shrinkage (see Figure 2b), wrong parameters will degenerate the mesh. In an empirical study, [Bad06] showed that even these non-shrinkage approaches are not appropriate to reduce artifacts in all iso-surface meshes without significant shrinkage and distortion.

In essence, none of the mesh-filtering techniques can ensure a correct representation of the original binary volume. Furthermore, considerable parameter tuning is required to avoid strongly distorted results. In contrast, filtering of the binary volume or filtering of the

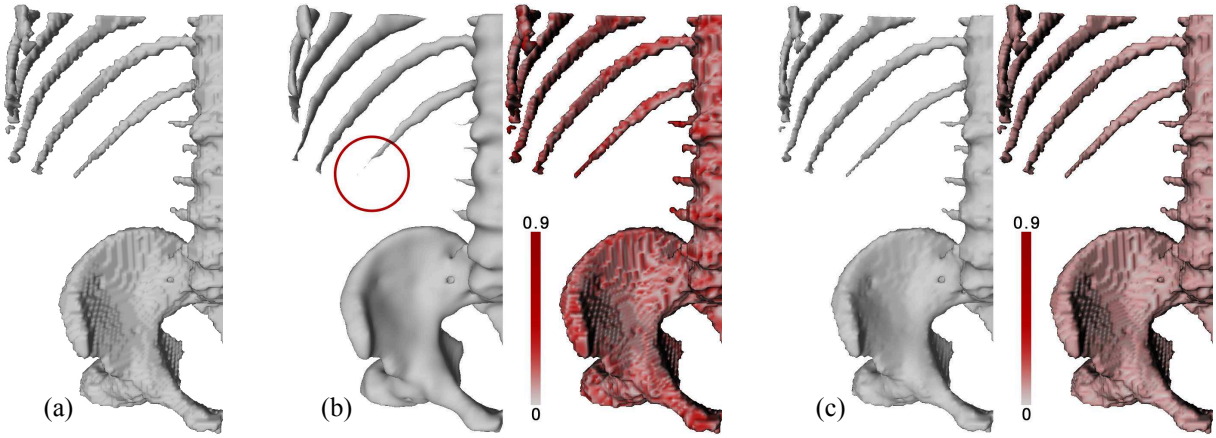


Figure 2: Extracted surface model of human bones from a binary segmented CT data set. (a) *Marching Cubes* result (104K triangles, 52k vertices); (b) surface *(a)* filtered by means of the *λ/μ-Filter* [Tau95] (with $\lambda \approx 0.7143$, $\mu \approx -0.7692$ and 110 iterations in 9.9 sec.) and corresponding distance map to *(a)* (max distance = 0.91); (c) surface *(a)* filtered by the proposed *diamond-constrained* method (stopping threshold of 0.002 achieved after 110 iterations in 7.4 sec.) and corresponding distance map to *(a)* (max distance = 0.29). (Distance measure is the symmetric Hausdorff-distance given as fraction of the cell diagonal)

mesh constrained to the volume data as part of the mesh extraction process can yield smooth and correct results. Furthermore, a constrained smoothing process converges against a surface of minimal area within the given constraints and requires no tuning of parameters. Unfortunately, these approaches require the original binary volume data.

Since extracted surfaces still bear information about the original volume data, we first address the problem of smoothing iso-surface meshes by detecting properties of the underlying volume in the extracted surface meshes. This information can then be used to constrain the mesh smoothing.

3. RECONSTRUCTION OF VOLUME DATA INFORMATION

In this section, we discuss how to reconstruct volume data information from iso-surface meshes. First, we discuss the basics of iso-surface extraction from binary volumes and derive legal assumptions about extracted surfaces. According to these assumptions, we present methods to reconstruct information about the underlying volume data from iso-surface meshes.

3.1 Iso-Surface Extraction from Binary Volumes

A binary volume from medical volume data (e.g. a segmentation result from CT or MRI data) is a three-dimensional, axis-aligned, regular grid with constant distances in each dimension (∂x, ∂y, ∂z) and with only two possible values (e.g. background and foreground) at each grid point. An iso-surface representing the border between foreground and background is defined as the surface located at the center between adjacent grid points with different values. The MC extraction method [Lor87] iterates cells defined by 8 grid points (voxels) over the whole volume and searches for cells containing the surface. Then, for each of these surface cells (with at least one foreground and one background labeled voxel) surface vertices are created. *MC* creates vertices exactly located at the midpoint of the edges of these surface cells (see Figure 3). Other methods act similar but may also create vertices inside the cells. Explaining each extraction algorithm in detail is beyond the scope of this paper. Thus, we will further refer to the *Marching Cubes* extraction scheme and its case table as illustrated in Figure 3.

In general, all common surface extraction methods create vertices that are located either inside or at the edge of the cells containing the surface. Therefore, we assume that each mesh vertex can be moved inside its cell or on its cell edge respectively without creating incorrect iso-surface representations of the binary volume. Exactly this effect is used by CESN to smooth the surface constrained to the cells in which each surface vertex is located.

For mesh smoothing without the original binary volume, cell size and cell centers have to be recon-

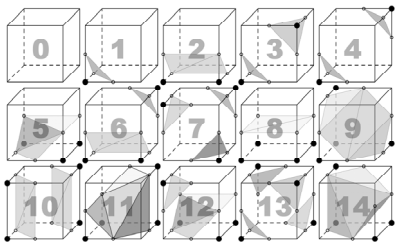


Figure 3: Marching Cubes case table. Concerning binary volumes: iso-surface vertices are only created at the midpoint of cube edges.

structed from the surface mesh. In the next sections, we present methods to reconstruct information about the volume data from iso-surface meshes. Since this is related to the extraction process used to extract a surface, we will explain cell size, cell center, and cell edge detection for *Marching-Cubes (MC)*-extracted meshes. We also give hints on easy adoption of this method for other extraction methods.

3.2 Cell Size Determination

The cell size of the original binary volume has an effect on the distance of extracted surface vertices. Here we use this relation to derive cell size from vertex distances.

Using MC, vertices are only created at cell edges. With the assumption of axis-aligned grid lines, the distance between adjacent mesh vertices in each dimension can only take three different values: 0, ½∂h, ∂h (see Figure 4a). Here, ∂h represents the extent of the cell in the current dimension. Thus, we determine the cell properties ∂x, ∂y, and ∂z by finding two different non-zero distances in each dimension.

```
FOR all vertices v ∈ Mesh DO
  FOR all n ∈ Neighbors(v) DO
    d = |v – n|
    IF d.x > 0 THEN
      add d.x to {sorted x-distances}
      (Note: The distance is only added
             if it is not in list yet)
    END IF
    IF d.y > 0 THEN
      add d.y to {sorted y-distances}
    END IF
    IF d.z > 0 THEN
      add d.z to {sorted z-distances}
    END IF
  END FOR
  IF each sorted distance list has two
     entries THEN terminate processing
END FOR
```

3.3 Cell Center Determination

To determine surface cell centers from a MC-extracted surface, one defined MC cell case has to be identified within the mesh. For simplicity, we decided to search for a cell of case 1 (see Figure 3). This is realized by searching for the vertex with the lowest x-, y-, and z-position.

```
FOR all vertices v ∈ Mesh DO
  IF v.x < minV.x THEN
    minV = v
  ELSE IF v.x = minV.x THEN
    IF v.y < minV.y THEN
      minV = v
    ELSE IF v.y = minV.y THEN
      IF v.z < minV.z THEN
        minV = v
      END IF
    END IF
  END IF
END FOR
```

With this strategy, we find vertex *minV* and its corresponding cell of case 1 as illustrated in Figure 4b. Since we know the position of vertex *minV* and the case of the cell, the coordinates of the cell center *c* are given by equation (1) as illustrated in Figure 4b.

$$c[x,y,z] = [\text{minV.x}, \text{minV.y} - ½\partial y, \text{minV.z} - ½\partial z] \quad (1)$$

Independent of the extraction method from one known cell center *c*, it is now possible to determine a cell center *c(v)* for each mesh vertex *v*. The position of the cell center for vertex *v* can be determined according to equation (2) for the x-dimension. The other dimensions are treated similarly.

$$c(v).x = c.x + \{ \text{round}[(c.x – v.x) / \partial x] \times \partial x \} \quad (2)$$

As a special property of *MC*-extracted iso-surfaces, their vertices can not be clearly associated with solely one cell center. Since vertices are positioned at the cell edges, each vertex can be associated with four neighboring cell centers. At this stage, it is sufficient to find one associated cell center. In Section 4.2, we will return to this problem.

3.4 Cell Edge Determination

Since MC creates vertices at cell edges, we have to determine the cell edge where each vertex is located. As illustrated in Figure 4a, there are 12 possible vertex locations. The distance in each dimension between a vertex and its associated cell center can be easily used to determine the cell edge where the vertex is located (see Figure 4b). The following pseudocode encodes each cell edge with a number as illustrated in Figure 4a:

```
FOR all vertices v ∈ Mesh DO
  d = c(v) – v
  IF d.x = 0 THEN      //x-direction
    case(v) = 1
    IF d.y < 0 THEN case(v)  = 3
    IF d.z < 0 THEN case(v) += 1
  ELSE IF d.y = 0 THEN //y-direction
    case(v) = 5
    IF d.x < 0 THEN case(v)  = 7
    IF d.z < 0 THEN case(v) += 1
  ELSE IF d.z = 0 THEN //z-direction
    case(v)  =  9
    IF d.x < 0 THEN case(v)  = 11
    IF d.y < 0 THEN case(v) += 1
  END IF
END FOR
```

The presented cell size, cell center, and cell edge determination methods yield sufficient information about the original binary volume. It must be noted that the presented determination methods have to be extended for arbitrarily rotated, skewed or otherwise manipulated surface meshes.

The gathered information can now be used to constrain vertex displacement during smoothing and it could even be used to reconstruct the binary volume

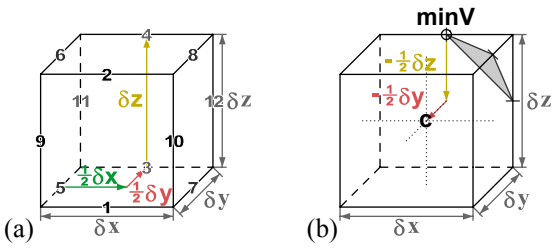


Figure 4: Cell size and cell center determination: (a) 12 possible vertex locations per cell with illustrated distance between vertex 5 and 4. (b) Vertex *minV* with minimal x-, y-, and z-position and illustrated distance to cell center.

itself. Since we focus on mesh smoothing, we further investigate the reduction of aliasing artifacts in iso-surfaces with this additional information.

4. CONSTRAINED ARTIFACT REDUCTION IN SURFACE MESHES

In this section, we discuss different schemes to constrain vertex displacement during smoothing. We derive these schemes from different levels of information about the volume data. We start by relying only on the cell size (e.g. voxel spacing) and proceed by adding more and more information about the volume data (see Section 3 for determination of this information). Furthermore, we derive a scheme to reduce artifacts in iso-surface meshes extracted from binary volume data by means of *Marching Cubes*.

4.1 Cell-Size-Constrained Smoothing

Assuming that all surface vertices are located inside a cell of the original discrete volume data, we state that the location of each vertex exhibits a discretization error of plus/minus one half of the cell size ±½(∂x, ∂y, ∂z). Consequently, we assume that the original object surface as well as a smooth surface representation of it is located within the given range of ±½(∂x, ∂y, ∂z) around each surface vertex. Now we can constrain the position of each vertex to that range around its original location.

For simplicity of the smoothing procedure, we iteratively move each vertex *v* towards a position *sv* equidistant to its neighbors. If *sv* is outside the given range around the original vertex position *vo*, the displacement vector from *vo* to *sv* is clipped at the border of the allowed range. This position is then used as the new vertex position *v*. Relaxation is stopped when the maximum occurring vertex displacement *maxDispl* in one iteration is lower than a given stopping threshold.

```
WHILE maxDispl > stoppingThreshold DO
  maxDispl = 0
  FOR all vertices v ∈ Mesh DO
    sv = equi-distant location between
         neighbors of v
    dv = sv – vo //displacement vector
    IF |dv.x| > ½δx THEN clip(dv.x, ½δx)
    IF |dv.y| > ½δy THEN clip(dv.y, ½δy)
    IF |dv.z| > ½δz THEN clip(dv.z, ½δz)
    v = vo + dv
    maxDispl = max(maxDispl, ||dv||)
  END FOR
WHEND
```

This strategy yields smooth results, but the underlying assumption is only fulfilled if all vertices are located in the center of the surface cells. This explains why CESN need to extract a mesh with vertices at cell centers only. For all other extraction processes, this method can yield incorrect representations of the binary volume as illustrated in Figure 5a.

With a maximum error to the original surface mesh of one half of the cell diagonal, this method can also be applied to MC-extracted surfaces if the precision is still adequate for the desired application. Considering the very low computation times (cell size determination included) and its error bound, this smoothing scheme can be considered as superior to most of the traditional mesh smoothing approaches (for results and comparison see Section 5).

4.2 Cell-Center-Constrained Smoothing

To guarantee a correct representation of the original volume data for surface meshes with vertices that are not located at cell centers, the vertices of those meshes have to remain inside their original surface cells. Thus, the algorithm from Section 4.1 has to be changed by replacing the original vertex location *vo* by the cell center *c(v)* (recall Section 3.3) for each vertex *v*. This method works well for all iso-surface meshes extracted by methods that create surface vertices inside surface cells (for example *DualMC* [Nie04]). This condition is not fulfilled for MC-extracted surfaces. Here, vertices are located at cell

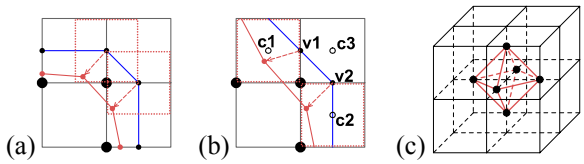


Figure 5: Incorrect Results: (a) vertices are allowed to move ±½(∂x, ∂y, ∂z); (b) vertex *v1* is constrained to remain inside its associated cell *c1* and *v2* to remain inside *c2* – To get a correct result, at least one vertex (*v1* or *v2*) has to be constrained to *c3*. However, whether *v1* or *v2* should be selected, depends on their neighbors. (c) 3d case where 6 vertices have to be constrained to 8 cell centers.

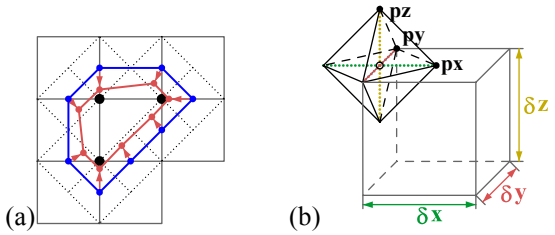
edges and are associated with four neighboring cell centers. As illustrated in Figure 5b, it is crucial which of those four cell centers is used for constraining a vertex. If it is not the correct one, constrained smoothing as described above can yield an incorrect representation of the original volume data.

Solving this assignment problem is not trivial. If one vertex has been assigned to one of its associated cell centers, the determination of the appropriate cell center for its neighboring vertices depends on the previous decision and on the decisions for all their neighbors. In 3d space, this assignment problem may also be insolvable in some cases. Figure 5c illustrates an example case with eight cells and only six vertices. Here all possible solutions leave two cell centers unassigned which may lead to incorrect representations. Thus, cell-center-constrained smoothing that guarantees a correct representation of the underlying volume data is not possible for surfaces extracted by means of MC.

4.3 Cell-Edge-Constrained Smoothing

To keep as close as possible to the original MC-extraction process and to guarantee correct representations of the underlying volume data, we constrain vertices to their cell edges where they are located. In Section 3.4, we presented a method to determine the exact cell edge where a vertex is located. Here we can simplify this method to distinguish only between cell edges in x-, y-, and z-direction. With that information for each vertex and the determined cell size, we can constrain vertices to move along their cell edge by a maximum of one half of the cell size in edge direction. This also speeds up the smoothing procedure since only the x-, y-, or z-component of a vertex according to the cell edge has to be calculated in each smoothing step.

```
//determine cell edge type simplified
FOR all vertices v ∈ Mesh DO
  d = c(v) - v
  IF d.x = 0 THEN      case(v) = x_edge
  ELSE IF d.y = 0 THEN case(v) = y_edge
  ELSE IF d.z = 0 THEN case(v) = z_edge
  END IF
END FOR
```



(a) (b)

Figure 6: Vertices created at cell edges are constrained to a rhomboid in 2d (a) and to a diamond-shaped region in 3d (b) to ensure correct representations of the binary volume.

```
//cell edge constrained mesh smoothing
WHILE maxDispl > stoppingThreshold DO
  maxDispl = 0
  FOR all vertices v ∈ Mesh DO
    IF case(v) = x_edge THEN
      sv.x= equi-distant position between
            neighbors in x-direction only
      dv.x= sv.x - vo.x //distance vector
      IF |dv.x|> ½δx THEN clip(dv.x, ½δx)
      v.x = vo.x + dv.x
      maxDispl = max(maxDispl, |dv.x|)
    ELSE IF case(v) = y_edge THEN
    ...
    ELSE IF case(v) = z_edge THEN
    ...
    END IF
  END FOR
WHEND
```

With this approach, each surface vertex remains at the cell edge where it was created. This strongly favors correctness over smoothness and forces the surface to retain small details. As a consequence, this method does not yield surfaces as smooth as possible with the other approaches (see Section 5). However, results are much smoother than an original MC-extracted surface and a correct representation of the original volume data is guaranteed in contrast to standard mesh smoothing approaches.

4.4 Diamond-Constrained Smoothing

Since *cell-edge-constrained* smoothing of MC-extracted surfaces does not yield well smoothed results, we derive a new constrained method that allows significant artifact reduction in MC-extracted iso-surface meshes while maintaining a correct representation of the original binary volume.

As Figure 6a illustrates in 2d space: Vertices created at the edge of a cell can be moved arbitrarily inside a rotated square or rhomboid centered at the cell edges while the resulting surface remains a correct representation of the binary data. We rely on this property and constrain vertices of MC-extracted surfaces to remain inside a diamond-shaped region as illustrated in Figure 6b. The diamond is centered at the vertex v with extents in x-, y-, and z-direction equal to the cell size in these directions.

To define these diamonds, we only need to reconstruct the cell size as described in Section 3.2. Then, the plane equations of the eight faces of the diamond can be pre-computed and re-used for each vertex. We use the standard equation $ax + by + cz + D = 0$ to represent the faces as planes, where $n=(a,b,c)$ represents the normal of the plane and D its distance from the center of the diamond. Since D is equal for all faces, we only store a single D and the normal of each of the eight faces.

During each smoothing step we can determine the vertex displacement vector dv from v to its relaxed position sv and clip dv with the appropriate face of

the diamond to find the new constrained position of *v*. To determine the appropriate face for clipping we check the sign of the displacement vector components. Then we calculate the intersection point of the displacement vector and the determined diamond face. If there is an intersection, we use this point as the new location of the current vertex.

With this approach, fast, converging, and volume-data-constrained artifact reduction in surface meshes extracted from binary volumes can be performed at the mesh-level. In contrast to previous work, it reconstructs information about the underlying volume data and constrains the smoothing process to yield correct representations of the binary volume independent of the presence of the original volume data.

```
//diamond generation
//face normals:
n[7] = (py - px).cross(pz - px)
n[7] = n[7] / ||n[7]||
n[0] = -n[7]
...
// face-center distance
// D = | t × Pn.dot(Rd) |
// t = 1;  Pn = n[7];  Rd = (px, 0, 0);
D  = | 1 × n[7].x × px |

//diamond clipping
diamondClipping(dv) {
  //Determine diamond face vector dv is
  //pointing at.
  IF dv.x > 0 THEN face = 4 ELSE face = 0
  IF dv.y > 0 THEN face = face + 2
  IF dv.z > 0 THEN face = face + 1
  //determine if dv has to be clipped
  denom = |n[face].dot(dv)|
  IF denom > D THEN //clip dv
    t = D / denom
    dv = dv × t
  ENDIF
  RETURN dv
}

//diamond constrained mesh smoothing
WHILE maxDispl > stoppingThreshold DO
  maxDispl = 0
  FOR all vertices v ∈ Mesh DO
    sv = equi-distant position between
         neighbors of v
    dv = sv - vo //distance vector
    dv = diamondClipping(dv)
    v  = vo + dv
    maxDispl = max(maxDispl, ||dv||)
  END FOR
WHEND
```

5. RESULTS

We used the *MeVisLab SDK* [MeV06] to implement the proposed methods: *cell-size-constrained*, *cell-edge-constrained* and *diamond-constrained* smoothing as well as CESN for comparison. Each method facilitates the same extended *Winged-Edge-Mesh* data structure for fast triangle mesh processing. Thus, computation times of the smoothing step can be compared between the different methods.

Figure 7 compares the results of the different smoothing approaches by means of a MC-extracted surface from a synthetic binary volume representing a binary sampled sphere with a diameter of 60 units. Furthermore, computation time (t in sec.), number of iterations (i), remaining percentage of original volume (V in %) as well as the maximum symmetric Hausdorff-distance to the MC-extracted surface (maxD as fraction of the cell diagonal) are given in the figure caption. For all smoothing examples we used a stopping threshold of 0.002 units.

As can be seen in Figure 7b, *cell-size-constrained smoothing* yields the best smoothing results but no correct representation of the underlying volume data (recall Section 4.1). Nevertheless, this method limits the maximum possible deviation to the initial mesh to one half of the cell diagonal which is superior to previous mesh smoothing approaches (see Figure 2b).

Correct representations are guaranteed by *cell-edge-constrained* (Figure 7c) and *diamond-constrained* smoothing (Figure 7d), while the diamond-constraint approach yields much better smoothing. In contrast to other mesh-smoothing approaches, the error is limited to the cell size in each dimension, artifacts are significantly reduced, and a correct representation is guaranteed. Similar results can only be achieved by CESN (Figure 7e), but that requires the binary volume.

Figure 8 shows smoothing results for a clinical dataset containing a segmented aneurysm (vessel pathology) achieved by the proposed *diamond-constrained smoothing* on the mesh level (Figure 8c) and by CESN on the mesh extraction level (Figure 8b). Since quantitative and visual results are very similar to each other, MC-extraction and subsequent *diamond-constrained* smoothing may also be used as an alternative to CESN on the mesh-extraction level.

6. CONCLUSION

We presented a strategy for artifact reduction in surface meshes extracted from binary volume data that acts (independently from the volume data) directly and solely on the surface mesh. In contrast to previous mesh filtering approaches, our method uses the volume data properties inherent in an extracted surface mesh to constrain the filtering process which requires no parameter tuning and yields smooth, converging, and correct representations. In detail, we presented the *diamond-constrained* mesh filtering method for surfaces extracted from binary volumes by means of *Marching Cubes*. Results are comparable to CESN, while in contrast to our method, CESN require the volume data and a specialized surface extraction scheme.

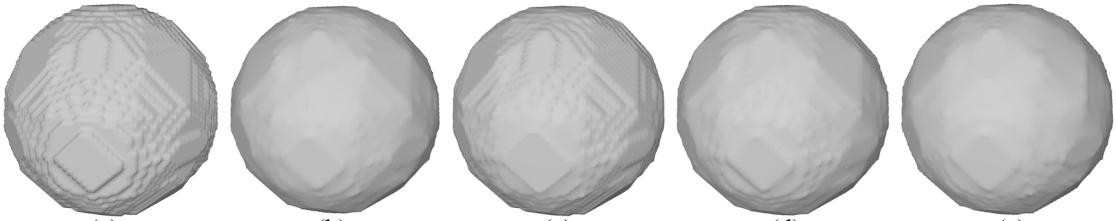


Figure 7: Surface of a binary sampled sphere (diameter: 60 units): (a) original MC result (35k triangles, 17k vertices); (b) *cell-size-constrained* smoothing ([incorrect representation], t=2.0, i=147, V=95.5%, maxD=0.5); (c) *cell-edge-constrained* smoothing (t=1.3, i=76, V=99.6%, maxD=0.29), (d) *diamond-constrained* smoothing (t=1.3, i=76, V=99.2%, maxD=0.24), (e) CESN (t=1.4, i=102, V=97.6%, maxD=0.39). (stopping threshold = 0.002)

In spite of the correctness and visual quality of artifact reduction in surface meshes by our method, it would be much better to reduce or avoid artifacts at an earlier stage of the surface extraction pipeline (recall Figure 1).

A still open surface mesh filtering problem is artifact reduction in elongated surface parts with a diameter of only one voxel, since such structures may collapse to a single point or line during smoothing. Thus, future work may focus on an appropriate treatment of such fine structures.

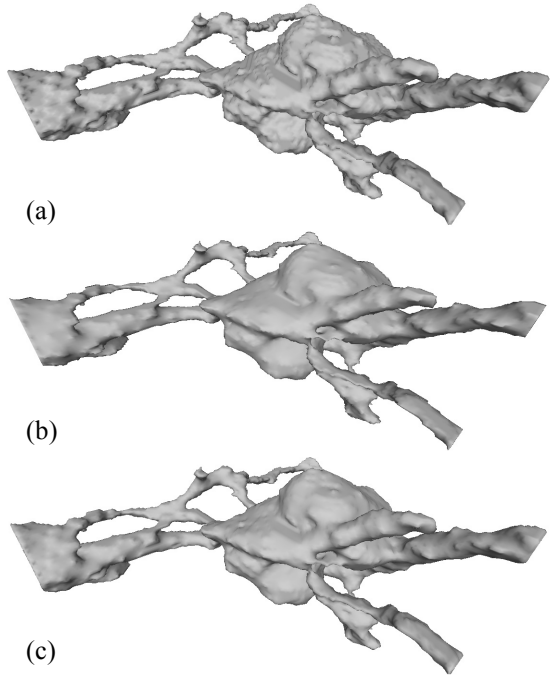


Figure 8: Surface representation of an aneurysm data set: (a) original MC-extracted surface (53k triangles, 26k vertices), (b) CESN result (t = 2.5 sec., i = 93, V = 89.9%, mD = 0.41), (c) *diamond-constrained* result (t = 2.6 sec., i = 59, V = 97.1%, mD = 0.28). (stopping threshold = 0.002).

7. REFERENCES

[All98] Allamandri F., Cignoni P., et al. Adaptively adjusting Marching Cubes output to fit a trilinear reconstruction filter. EG Workshop on Scientific Visualization '98, pp. 25-34, 1998.

[Bad06] Bade R., Haase J., Preim B. Comparison of fundamental mesh smoothing algorithms for medical surface models. SimVis'06, pp. 289-304, 2006.

[Baj03] Bajaj C. and Xu G. Anisotropic Difusion on Surfaces and Functions on Surfaces. ACM Trans. on Graphics, Vol. 22(1), pp. 4–32, 2003.

[Des99] Desbrun M., Meyer M., et al. Implicit fairing of irregular meshes using diffusion and curvature flow. SIGGRAPH '99, pp. 317–324, 1999.

[Gib98] Gibson S. F. F. Constrained Elastic Surface Nets: Generating Smooth Surfaces from Binary Segmented Data. MICCAI'98, pp. 888-898, 1998.

[Lor87] Lorensen W. E., Cline H. E. Marching Cubes: A high resolution 3D surface construction algorithm. SIGGRAPH'87, pp. 163-169, 1987.

[MeV06] MeVis Research. MeVisLab: Medical Image Processing and Visualization. www.mevislab.de. last viewed: Sep. 29th 2006.

[Nie04] Nielson, Gregory M. Dual Marching Cubes. IEEE Visualization'04, pp. 489-496, 2004.

[Tau95] Taubin, G. A signal processing approach to fair surface design. SIGGRAPH'95, pp. 351–358, 1995.

[VMM99] Vollmer, J.; Mencel, R. and Mueller, H. Improved laplacian smoothing of noisy surface meshes. EuroGraphics'99, pp. 131–138, 1999.

[Whi00] Whitaker, Ross T. Reducing aliasing artifacts in iso-surfaces of binary volumes. IEEE VolVis'00, pp. 23-32, 2000.