Direct Contouring of Implicit Closest Point Surfaces

S. Auer and R. Westermann

Technische Universität München

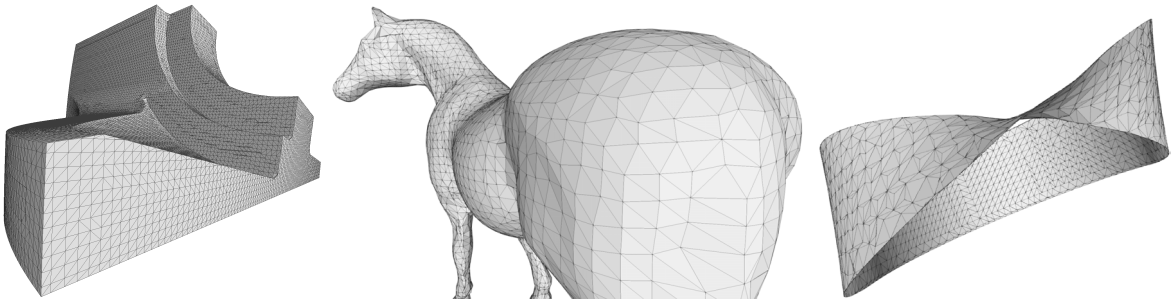

Figure 1: *Triangle meshes reconstructed from closest point fields using our novel closest point contouring (CPC)*

Abstract

Constructing a (signed) distance field and contouring its zero level set are two important steps in many surface reconstruction methods. While most high-quality distance transforms compute the distance to the surface as well as the closest point on it, the contouring step typically uses only the distance and omits the closest point information. Our novel closest point contouring algorithm (CPC) uses the full closest point field, and, thus, allows improving existing methods for high-quality triangle mesh reconstruction based on implicit function models: Since we select the vertex positions directly from the set of closest points, all triangle vertices are guaranteed to lie exactly on the zero-contour and no approximations are necessary. By employing recent findings in the context of so-called embedding techniques, we derive a formulation of the mean curvature vector on the closest point representation and use this formulation to properly select the vertices to be triangulated. In combination with a new table-based triangulation scheme this allows us to detect and preserve sharp features, and to avoid small degenerated triangles in smooth areas. CPC can handle open and non-orientable surfaces, and its data-parallel nature makes it well suited for GPUs.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction and Related Work

Implicit surfaces, defined as zero level sets in discrete volumetric distance fields, serve as intermediate representations in a number of computer graphics applications. A distance field can be computed in a number of ways. According to [JBS06] there are two categories of such algorithms: Exact or direct algorithms compute explicitly the Euclidean distance between a vertex of the discrete sampling grid and its corresponding closest point (CP) on the continuous surface. Approximate distance transforms perform a direction-dependent uneven propagation of distance measures from vertex to vertex, typically building up distances in increments to speed up computation time. The distance measure can either be scalar-valued, or the distance transform builds upon the propagation of vectors to the closest surface points to achieve superior accuracy. High-quality algorithms from both categories can thus be extended easily to also generate

DOI: 10.2312/conf/EG2013/short/001-004

a CP representation of a given surface, i.e., a 3D surface-embedding grid which stores at each grid vertex the CP on the surface.

For constructing a triangle mesh from a distance field, contouring algorithms like the Marching Cubes (MC) algorithm [LC87] can be employed. If the intersections of the surface with the edges of the embedding grid are known, the Extended Marching Cubes (EMC) [KBSS01] and the Dual Contouring (DC) [JLSW02] algorithms extract sharp features by approximating surface positions in the interior of grid cells via quadratic error function minimization.

1.1. Our contribution

We propose a new contouring approach that entirely avoids the approximation of interior-cell surface positions, and, instead, directly obtains a high-quality surface triangulation from a CP representation. The construction of a CP field is no more complicated than the construction of a high-quality distance field, and the CPs can be used directly as mesh vertices. Since the CPs lie exactly on the surface, there is no need for any approximation of triangle vertices in the interior of grid cells. We will subsequently call our proposed algorithm Closest Point Contouring (CPC).

It is not clear per se, however, which subset of the CPs in the 3D embedding grid should become the vertices of the triangle mesh, and how to connect the vertices in order to extract a high-quality triangulation. In this work we propose a vertex selection oracle which makes use of an embedding technique for the solution of partial differential equations (PDEs) on surfaces [RM08, MR08]. This oracle performs a Laplacian analysis on the embedding CP grid, and uses the mean curvature vectors at the CPs to guide the selection of mesh vertices according to the surface's shape.

In summary, the main contributions of our paper are:

- An efficient, high-quality contouring method that works solely on a surface embedding 3D closest point grid.
- An oracle for selecting a feature-preserving subset of all closest points, which is based on a Laplacian analysis of the embedding grid.
- A novel table-based algorithm for triangulating the voxels of a 6-separating surface voxelization.

1.2. The Closest Point Method

CPC builds upon the ideas of the Closest Point Method (CPM) [RM08, MR08], which is an embedding method for numerically solving PDEs on arbitrary surfaces, i.e., it replaces differential operators on a surface with their standard Cartesian counterparts on a discrete regular grid. All surface attributes are stored in a Cartesian multiblock grid, which embeds the surface and a narrow band around it. At every grid vertex the CPM requires the position of the surface point with the shortest Euclidean distance. For all but one of the examples in this work, the closest points were computed explicitly from triangle meshes, using the CUDA algorithm described in [AMT*12]. In Figure 5, however, we computed the CPs from an unorganized point cloud, by projecting the grid vertices onto a moving least squares surface representation using the APSS algorithm [GG07].

2. Algorithm Description

CPC is a two-pass approach, in which each CP corresponds to its defining vertex in the primal embedding grid and to the cell in the dual grid which is centered at the primal vertex.

The first pass iterates over the primal grid vertices and selects a subset of their CPs as mesh vertex candidates. We interpret the corresponding selection of cells in the dual grid as a surface voxelization. To obtain a closed triangulation in the second step, this voxelization must be 6-separating [KCY93]. A realization of the first pass for smooth surfaces is discussed in Section 2.1. The contour shown in Figure 5 bottom left was reconstructed with this technique. In Section 2.3 we present an advanced algorithm for surfaces with sharp features, which was used for all other CPC results shown in this work.

The second pass iterates over all primal grid cells with three or more selected corner vertices, and, comparable to the MC algorithm, it determines the connectivity of the respective candidate CPs with a lookup into a triangulation table. Contrary to MC, however, CPC does not restrict the triangle vertices to the edges of the grid, but instead uses the exact closest surface points of the grid vertices. The second pass is detailed in Section 2.2.

2.1. Triangulation-Aware CP Selection

A 6-separating CP selection can be computed by finding the voxels containing the surface. To do so, we consider only the distances between each voxel center $\mathbf{x}_v$ (i.e., the position of the primary grid vertex) and its CP on the surface $\mathbf{cp}(\mathbf{x}_v)$. 6-separation is guaranteed if we select CPs with a Euclidean distance less than half the voxel diameter. Such a selection, however, contains many false positives, which leads to over-tessellation and degenerated triangles.

To obtain a less conservative selection, we calculate the distances based on the L^∞-norm instead of the L^2-norm, and select all CPs with a Chebyshev distance $D_C(\mathbf{p},\mathbf{q}) := \max_i(|p_i - q_i|)$ less than half the grid spacing h. This intersection test is exact if the CP representation is also based on the L^∞-norm. For a Euclidean representation it can miss intersections if the Euclidean CP is outside the voxel and the Chebyshev CP is inside. Yet even in this case the resulting selection is 6-separating, which is why we always prefer the L^∞-norm for distance computations.

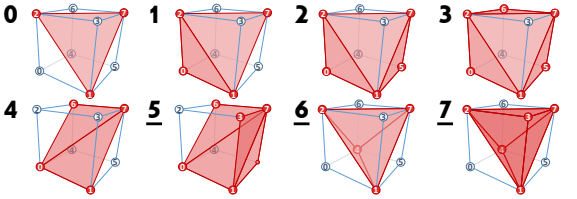

Figure 2: *Triangulated cubes*

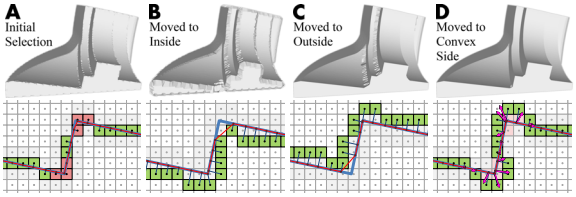

Figure 3: *Effect of moving the selection to the convex side*

2.2. Table-Based CP Triangulation

Like MC, CPC determines the triangle connectivity for a primal grid cell by building an 8-bit index from the local indices (depicted in Figure 2) of the selected cell vertices. This index identifies one out of 256 possible cube configurations in a precomputed triangulation table.

Each entry in the triangulation table is a triangle list containing the local indices of up to 6 triangles. To fill the table, we first consider the cube faces containing the corner with local index 0, to avoid redundant evaluation in adjacent cells. If the CPs of all 4 vertices of a face are selected, we triangulate this face by adding a quad of two triangles to the respective table entry. In a second step, we consider the 8 cases depicted in Figure 2 and all of their permutations arising from the 48 cube symmetries (24 rotations and reflection). If a cube configuration matches one of these cases, the respective table entry is filled with the triangulation depicted in the figure (rotated or reflected as required). For the underlined cases in the figure, which represent non-manifold triangulations, we set an additional non-manifold bit.

If this bit is detected after the first table lookup, we treat the most distant selected CP as unselected and compute a new cube configuration. A manifold triangulation is then obtained with a second lookup. The additional lookup can be omitted, if manifoldness is less important than performance.

2.3. Feature-Sensitive CP Selection

We adjust the initial selection from Section 2.1 so that CPs on sharp features are preferred. The idea underlying our approach is illustrated in Figure 3. Column A shows the initial selection of voxels and the resulting CPC reconstruction. In columns B and C this selection was moved to the 6-adjacent voxels on the inside, respectively outside of the object. We observe that features are preserved where the selection was moved to the locally convex side of the surface. The reason for this effect is that in a certain area on the convex side near an edge or corner, all voxel centers are closest to a point on the feature. Column D shows the result of the feature-sensitive CP selection. Here the selection was adaptively moved towards the locally convex side, guided by the negative mean curvature vectors (purple arrows) to identify areas of high curvature and their convex side.

The mean curvature vector at a point $\mathbf{x}_s$ on the continuous surface S is given by

$$\mathbf{H}(\mathbf{x}_s) = \nabla_S^2\, \mathbf{id}(\mathbf{x}_s), \tag{1}$$

where ∇_S^2 denotes the Laplace-Beltrami operator, and $\mathbf{id}(\mathbf{x}) = \mathbf{x}$ is the identity function. $\mathbf{H}(\mathbf{x}_s)$ is parallel to the surface normal, points to the locally concave side of S (i.e., on convex objects it points inside), and its length equals the absolute value of the mean curvature. The closest point method [RM08] allows us to define $\mathbf{H}$ on the discrete embedding grid. The CPM's equivalence of gradients principle states that the Laplace-Beltrami operator applied to a function f and the standard Cartesian Laplace operator applied to the *closest point extension* of f agree on the surface:

$$\nabla_S^2 f(\mathbf{x}_s) = \nabla^2 f(\mathbf{cp}(\mathbf{x}_s))$$

Since the CP function $\mathbf{cp}(\mathbf{x})$ is the CP extension of the identity function, we can rewrite Equation 1 to

$$\mathbf{H}(\mathbf{x}_s) = \nabla^2\, \mathbf{cp}(\mathbf{x}_s). \tag{2}$$

To discretize this PDE, a standard discrete Laplace operator with a $3\times3\times3$ stencil is applied to the CPs in the embedding grid. The result $\tilde{\mathbf{H}}$ on the discrete grid equals $\mathbf{H}$ at points on the surface. Since a grid vertex v does not coincide with a surface point in general, we evaluate $\tilde{\mathbf{H}}$ at the CP of v: $\mathbf{H}(\mathbf{x}_v) = \tilde{\mathbf{H}}(\mathbf{cp}(\mathbf{x}_v))$. This discrete CP extension is implemented as a WENO4 interpolation [MR08].

To guide the movement of the initial selection to the convex side, we classify the grid vertices based on their mean curvature vectors. The *classification* stage first computes $\mathbf{H}(\mathbf{x}_v)$ in the described way and then decides for each grid vertex v if its selection and deselection are allowed. A selection is allowed only if v was never selected before and if it does not lie on the locally concave side. Concavity is assumed if the inner product $\mathbf{H}(\mathbf{x}_v)\cdot(\mathbf{cp}(\mathbf{x}_v)-\mathbf{x}_v)$ is larger than a small ε. A deselection is allowed only if v is close to a feature and if the resulting selection is still 6-separating. We assume that a feature is close if $D_C(\mathbf{cp}(\mathbf{x}_v),\mathbf{x}_v)+h/2 < \lambda|\mathbf{H}(\mathbf{x}_v)|$, with λ being a user-defined value to control the sensitivity of the feature detection. 6-separation is preserved if $\mathbf{H}(\mathbf{x}_v)\cdot\mathbf{H}(\mathbf{x}_{\tilde{v}}) > \varepsilon$ for all $\tilde{v}$ in the 26 neighborhood, i.e. if the concave side does not change at the adjacent voxels.

Based on the classification we then apply two iterations of the *move* stage to the initial selection. In each iteration, we check for each currently selected vertex if its deselection is allowed. If allowed, we deselect the vertex and instead select all of its 6-adjacent neighbors which are selectable. After two iterations all concave deselectable grid vertices are effectively removed from the selection.

Table 1: *Contouring Timings (in milliseconds)*

Res.	Triangles	Classific.	Move	Triangul.	Total
64^3	25k	6	6	6	18
128^3	108k	21	15	17	53
256^3	401k	82	68	58	208
512^3	1.5M	270	213	221	704

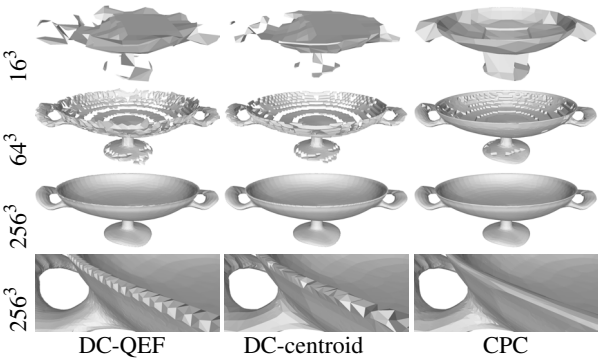

Figure 4: *CPC compared to Dual Contouring*

3. Results and Conclusion

Note that all triangle meshes in this work were rendered with per-pixel lighting, using flat triangle normals. Figure 1 shows that CPC effectively preserves sharp creases and corners (left), that it generates reasonable triangulations for smooth areas as well as for smaller details (middle), and that it supports open and non-orientable surfaces (right). Figure 5 shows that an existing technique for the generation of a distance-based implicit surface can easily be modified to generate a CP-based representation. In such cases, CPC with the initial selection algorithm only is a viable substitute for MC. It is just as simple to implement, yet it uses exact vertex positions which are not restricted to the grid edges and it avoids the generation of degenerated triangles, resulting in 18% less triangles in the example. In Figure 4 we compare the reconstruction quality of CPC and DC for a noisy input model and varying grid resolutions. Besides the standard DC which approximates vertex positions by minimizing a quadratic error function, we also include a more noise-tolerant variant which uses the centroid of the edge-surface intersections. Especially at the silhouettes it is apparent that the contour quality is further improved by using exact CPs.

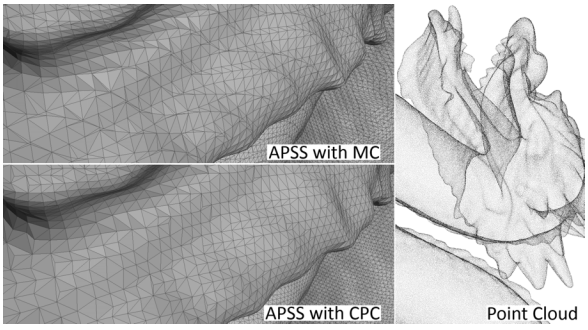

Figure 5: *Contouring a moving least squares surface*

In Table 1, we present reconstruction times for the results shown in Figure 4. The resolution of the defining Cartesian grid and the resulting number of generated triangles are listed in the first two columns. The third and forth columns show the times spent in the classification and move stages of the feature-sensitive CP selection, which are implemented in CUDA. The last two columns show the timings for the triangulation stage, which is implemented in a Direct3D geometry shader, and the total reconstruction time.

In summary, we have presented a very efficient contouring method which rigorously exploits a closest point representation to extract high-quality surfaces. The proposed contouring method opens a number of future research directions, especially in the context of the surface PDEs. When the CPM should be used to dynamically animate a surface, an explicit representation helps to accurately track the movement of the surface. After a few time-steps a re-initialization of the triangle mesh is required, however, since otherwise the simulation cannot generate new surface details. Further research is necessary to determine if a weighting scheme based on the Laplacian analysis in Section 2.3 could reduce the problems at sharp edges mentioned in [AMT*12].

Acknowledgment

This work was funded by the Munich Centre of Advanced Computing at the Technische Universität München.

References

[AMT*12] AUER S., MACDONALD C. B., TREIB M., SCHNEIDER J., WESTERMANN R.: Real-time fluid effects on surfaces using the closest point method. *Computer Graphics Forum 31*, 6 (2012). 2, 4

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. In *Proceedings of SIGGRAPH* (2007), vol. 26 of *ACM Transactions on Graphics*. 2

[JBS06] JONES M. W., BAERENTZEN J. A., SRAMEK M.: 3D distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006). 1

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. In *Proceedings of SIGGRAPH* (2002), vol. 21 of *ACM Transactions on Graphics*. 2

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature sensitive surface extraction from volume data. In *Proceedings of SIGGRAPH* (2001), vol. 20 of *ACM Transactions on Graphics*. 2

[KCY93] KAUFMAN A., COHEN D., YAGEL R.: Volume graphics. *Computer 26*, 7 (1993). 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of SIGGRAPH* (1987), vol. 21 of *ACM Computer Graphics*. 2

[MR08] MACDONALD C. B., RUUTH S. J.: Level set equations on surfaces via the Closest Point Method. *Journal of Scientific Computing 35* (2008). 2, 3

[RM08] RUUTH S. J., MERRIMAN B.: A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics 227* (2008). 2, 3