Acta Cybernetica — *online–first paper version* — pages 1–16.

Corner-Based Implicit Patches*

Ágoston Sipos[a]

Abstract

Free-form multi-sided surfaces are often defined by side interpolants (also called ribbons), requiring that the surface has to connect to them with a prescribed degree of smoothness. I-patches represent a family of implicit surfaces defined by an arbitrary number of ribbons. While in the case of parametric surfaces describing ribbons is a well-discussed problem, defining implicit ribbons is a different task.

In this paper, we introduce a new representation, *corner I-patches*, where implicit corner interpolants are blended together. Corner interpolants are usually simpler, lower-degree surfaces than ribbons. The shape of the patch depends on a handful of scalar parameters; constraining them ensures continuity between adjacent patches. Corner I-patches have several favorable properties that can be exploited for design, volume rendering, or cell-based approximation of complex shapes.

Keywords: implicit surfaces, multi-sided patches, volumetric data

1 Introduction

Computer Aided Geometric Design focuses on the mathematical representation of complex surface geometries. There is a wide variety of side interpolating multi-sided free-form surfaces in the literature, including both parametric [4, 11, 7, 17, 18] and implicit [1, 6, 16] patches. They are popular in curvenet-based design, as a patchwork of smoothly connected complex N-sided patches can be automatically created from simple ribbon surfaces.

The common concept behind these patches is that *ribbons* are introduced for each side, then *blending functions*, that satisfy prescribed continuity constraints at the boundaries, mix those together. In the case of parametric multi-sided patches, ribbons in most cases are tensor-product surfaces. The blend functions are usually (not always) defined on a polygonal domain and the surface points are calculated as a weighted sum of the well-parametrized points of the ribbons.

*This work was supported by the Hungarian Scientific Research Fund (OTKA, No. 124727).

[a]Department of Control Engineering and Information Technology, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Hungary, E-mail: asipos@edu.bme.hu, ORCID: 0000-0002-5562-2849

DOI: 10.14232/actacyb.299598

In the case of implicit surfaces, ribbons and blending functions are represented by implicit functions, defined on the whole 3D space. This, however, requires careful construction. An implicit surface, including the ribbons themselves, might interpolate the desired patch boundary, and have a very uneven shape inside the relevant space region at the same time. Sometimes it can have disconnected branches or self-intersections. However, many operations are computationally less expensive when using implicit surfaces, including ray tracing, intersections, point classification, or joining trimmed patches.

For this reason, the polynomial degree of ribbons and blend functions is preferred to be as low as possible. This poses limitations when used in design, as very detailed surfaces cannot be represented with a single patch. Locally defined surface elements are therefore important.

This paper explores the capabilities of a corner-based implicit surfacing scheme, where the patch is constructed by blending corner interpolants.

2 Previous work

The precursor of research on ribbon-based implicit surfaces is Liming's work on interpolating curves [10] and the functional splines [9]. Formally, in the surface equation, the product of the surfaces to be interpolated is used, which, in the case of implicit surfaces, means taking their union. Thus, the information we had while they were separate surfaces is lost. The improved version of the functional spline, the *symmetric functional spline* [6] collects the interpolated surfaces into two categories in its equation, but similarly can take the union of a higher number of them.

In the case of I-patches [16] the surface equation has an arbitrary number of sides appearing separately in the equation. This helps ensure that the surface is consistently oriented and the appropriate sides of the ribbons are joined together. For details, see Section 3. I-patches were applied for polyhedral design [14] and approximating triangular meshes [13] while ensuring geometric continuity.

A different way to approach the problem of interpolating implicit surfaces is to directly solve equations to get a minimal degree surface adhering to point, normal, curve, and normal fence constraints [2]. Doing this on the whole space may lead to high-degree, poor-quality surfaces. A scheme similar to the current work, A-patches [1], prevents this by constraining each surface inside a tetrahedron.

Another aspect of current work relates to the extraction of isosurfaces from discrete data on a regular grid. Generating a mesh is usually performed by derivative methods of Marching Cubes [12]. Direct rendering generally goes through interpolation methods, smooth surfaces can be acquired by tricubic interpolation [8]. There is also a list of enhanced approaches like using a modified (BCC or FCC) grid structure [15] or storing gradient values and approximating the isosurface by Taylor polynomials [3].

The potential representation of isosurfaces extracted on a grid by patches was investigated in [5]. First, a boundary curve network with Hermite interpolation is

created, then the surface is represented using multi-sided parametric patches.

3 Preliminaries

Implicit surfaces are constant isosurfaces of real-valued functions defined on the 3D space. Usually the zero-isosurface is used: for a function $f : \mathbb{R}^3 \to \mathbb{R}$ the surface is $\{(x, y, z) \in \mathbb{R}^3 \mid f(x, y, z) = 0\}$. In the following, capital letters in the formulae will mean implicit functions ($\mathbb{R}^3 \to \mathbb{R}$), but the function arguments (x, y, z) will be omitted for readability.

Ribbon-based implicit surfaces are usually described in the following way. For each side, there is a given surface R_i, that the surface should smoothly connect to, with a given order of continuity. Then, there is a fixed equation of the patch, combining the R_i-s and other defining surfaces.

In the case of I-patches, which are the basis of the current research, the equation is

$$\sum_{i=1}^{n} \left(w_i R_i \prod_{\substack{j=1 \\ j \neq i}}^{n} B_j^k \right) + w_0 \prod_{j=1}^{n} B_j^k = 0, \tag{1}$$

where

- n is the number of sides
- R_i are the ribbons, one for each side, to which the patch connects
- B_i are the *bounding surfaces*, whose intersection curves with the corresponding R_i define the boundaries of the patch
- $0 \neq w_i \in \mathbb{R}$ are scalar parameters and $2 \leq k \in \mathbb{N}$ is an integer parameter determining the degree of continuity

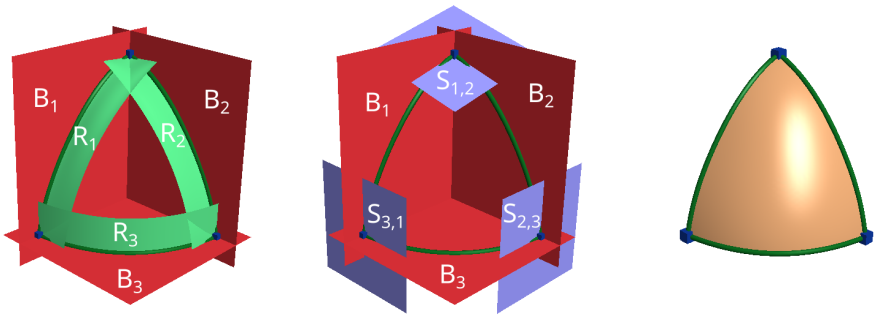
The patch connects with G^{k-1} continuity to the ribbons along the bounding surfaces, as shown in [16]. It has also been proven that the I-patch represents a consistent distance function with inside and outside, in case of well-chosen signs of w_i-s [14]. See Figures 1a and 1c for an example.

In the following, we will use $k = 2$ to keep the polynomial degree as low as possible, and in this paper, we discuss patches with G^1 continuity.

4 Corner I-patch

4.1 Basic equation

A corner I-patch is composed of corner interpolants $S_{1,2}, S_{2,3}, ..., S_{n,1}$ and bounding surfaces $B_1, B_2, ..., B_n$ (neither of them coincides with another one), such that $S_{i,i+1}$ denotes the corner interpolant between the ith and the $(i + 1)$th boundaries.



(a) Input (ribbons and bounding surfaces) for an **I-patch**

(b) Input (corners and bounding surfaces) for a **corner I-patch**

(c) Approximate shape of resulting patches

Figure 1: I-patch and corner I-patch

Then, the equation of the corner I-patch is

$$\sum_{i=1}^{n}\left(w_{i,i+1}\cdot S_{i,i+1}\cdot\prod_{\substack{j=1\\ j\neq i,j\neq i+1}}^{n}B_j^2\right)+\sum_{i=1}^{n}\left(w_i\cdot\prod_{\substack{j=1\\ j\neq i}}^{n}B_j^2\right)+w\prod_{i=1}^{n}B_i^2=0, \qquad (2)$$

where the $w_{i,i+1}$ scalars can be merged into $S_{i,i+1}$, as multiplying with a nonzero number does not change the implicit isosurface, only its distance metric. See an example of corner interpolants and bounding surfaces in Figure 1b.

Some important properties of this representation are:

- In each corner, the patch connects with G^1 continuity to the corner interpolants. (This means that the gradient vectors of the surface have the same direction as the gradients of the interpolants there.)
- Along the ith boundary, the shape of the surface does not depend on w and w_j for $j \neq i$.

The $w_{i,i+1}$ coefficients will be called corner coefficients, w_i-s are the side coefficients and w is the central coefficient.

4.2 Comparison to (side-based) I-patches

A disadvantage of I-patches is that their gradient is a zero vector in the corner points. This may lead to poor surface quality and generally should be avoided. However, the gradient of the corner I-patch can easily be proven to be the gradient of the corner interpolant times a nonzero number.

The corner I-patch along the ith boundary connects smoothly to the implicit surface

$$S_{i-1,i} \cdot B_{i+1}^2 + S_{i,i+1} \cdot B_{i-1}^2 + w_i \cdot B_{i-1}^2 \cdot B_{i+1}^2 = 0. \tag{3}$$

This itself is a 2-sided I-patch. Corner I-patches are thus similar (but not equivalent) to I-patches defined by ribbons that are themselves I-patches. (Such surfaces were described in [14].) This is because the I-patch defined by the ribbons in Equation 3 would be

$$\begin{aligned} \sum_{i=1}^{n} \left(S_{i,i+1}(B_{i+1}^2 B_{i-1}^2 + B_i^2 B_{i-2}^2) \prod_{\substack{j=1 \\ j\neq i, j\neq i+1}}^{n} B_j^2 \right) + \\ + \sum_{i=1}^{n} \left(w_i B_{i-1}^2 B_{i+1}^2 \prod_{\substack{j=1 \\ j\neq i}}^{n} B_j^2 \right) + w \prod_{i=1}^{n} B_i^2 = 0, \end{aligned} \tag{4}$$

which is not equivalent to Equation 2. Indeed, the factor $(B_{i+1}^2 B_{i-1}^2 + B_i^2 B_{i-2}^2)$ causes the I-patch's gradient to be zero at the corner points. Accordingly, corner I-patches have a lower degree of $2n$, as opposed to the $2n + 2$ for this kind of I-patches.

4.3 Setting coefficients

The w_i and w parameters can be set in a process similar to I-patches [16] forcing the patch to interpolate one point on each boundary and one point in the interior of the patch. As the shape of the surface on the ith boundary depends only on w_i, each of those can be set separately, and finally, w can be set to interpolate an interior point. I.e.:

$$w_i := -\frac{S_{i-1,i}(\mathbf{p}_i) \cdot B_{i+1}^2(\mathbf{p}_i) + S_{i,i+1}(\mathbf{p}_i) \cdot B_{i-1}^2(\mathbf{p}_i)}{B_{i-1}^2(\mathbf{p}_i) \cdot B_{i+1}^2(\mathbf{p}_i)} \tag{5}$$

$$w := -\frac{\displaystyle\sum_{i=1}^{n} \left(S_{i,i+1}(\mathbf{p}_0) \cdot \prod_{\substack{j=1 \\ j\neq i, j\neq i+1}}^{n} B_j^2(\mathbf{p}_0) \right) + \sum_{i=1}^{n} \left(w_i \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n} B_j^2(\mathbf{p}_0) \right)}{\displaystyle\prod_{i=1}^{n} B_i^2(\mathbf{p}_0)}, \tag{6}$$

where $\mathbf{p}_i$ is a point on the ith side, $\mathbf{p}_0$ is a point in the interior to interpolate. The parameters can be computed in this order, i.e. first all w_i, then w.

See examples later, in Figure 4.

4.4 Limitations

When connecting neighboring patches with geometric continuity, we need them to coincide at their common boundary in both a positional and a differential sense. As the corner I-patch along B_i connects to the surface defined by Equation 3, the patch on the other side of B_i also has to connect to it. This, however, only happens if B_{i-1} and B_{i+1} are identical to the corresponding bounding surfaces for the other patch.

This is not easily fulfilled when creating a general topology patchwork, but it is straightforward if the space is subdivided by planes, creating finite volume cells. Any such cell structure could theoretically work with corner I-patches, however, the most practical and useful is a regular grid of cubes.

5 Corner I-patch with multiple loops

5.1 Motivation and equation

In some cases, an isosurface must be represented by several disjoint surface elements. In Marching Cubes [12] for example, 7 of the 15 basic configurations result in a surface represented by more than one polygon. These surface elements could be represented separately, but in an implicit representation, it is advantageous to have the same implicit function on a well-defined 3D volume, otherwise, the piecewise implicit function would likely have discontinuities.

Fortunately, corner I-patches are capable of achieving this with a little modification. Consider m separate boundary loops where the lth of those is n_l-sided and for each of them the previously defined corner interpolants $S^l_{1,2}, S^l_{2,3}, ..., S^l_{n_l,1}$ and the bounding surfaces $B^l_1, B^l_2, ..., B^l_{n_l}$. Then the new equation is

$$\begin{aligned}\sum_{l=1}^{m}\sum_{i=1}^{n_l}\left(w_{l,i,i+1}\cdot S^l_{i,i+1}\cdot\prod_{k=1}^{m}\prod_{\substack{j=1\\k\neq l\vee(j\neq i\wedge j\neq i+1)}}^{n_k}(B^k_j)^2\right)+\\+\sum_{l=1}^{m}\sum_{i=1}^{n_l}\left(w_{l,i}\cdot\prod_{k=1}^{m}\prod_{\substack{j=1\\k\neq l\vee j\neq i}}^{n_k}(B^k_j)^2\right)+w\prod_{l=1}^{m}\prod_{i=1}^{n_l}(B^l_i)^2=0.\end{aligned}\tag{7}$$

The meaning of this is that for each corner interpolant, the bounding surfaces not multiplied to it are the two ones beside it, corresponding to the next and previous boundaries of the patch. A simple multiloop surface can be seen in Figure 2a, with two loops each composed of three corners.

5.2 Coinciding bounding surfaces

In a multiloop setting, especially if working in a grid of cubes, some bounding surfaces will likely coincide. Consider, for example, the configuration in Figure 2b

(a) Two 3-sided loops (b) A 3- and a 4-sided loop

Figure 2: Multiloop patches

(a configuration of Marching Cubes), where one of the boundings for the 3-sided loop coincides with one of those of the 4-sided loop. The problem with that is that when two bounding surfaces coincide, they will be in all the members of the weighted sum and thus can be factored out from the equation.

With regard to I-patches, a modified equation for this problematic case has been proposed in [14], however, it takes advantage of the 1-to-1 relation between ribbons and bounding surfaces which is not applicable to corner patches.

In this paper, the following solution is proposed. When computing the product of the bounding surfaces not neighboring the respective corner, omit those as well which coincide with one of the neighboring ones.

The formalized equation is:

$$\begin{aligned}\sum_{l=1}^{m}\sum_{i=1}^{n_l}\left(w_{l,i,i+1}\cdot S^l_{i,i+1}\cdot\prod_{k=1}^{m}\prod_{\substack{j=1\\B^k_j\neq B^l_i,B^k_j\neq B^l_{i-1}}}^{n_k}(B^k_j)^2\right)+\\ +\sum_{l=1}^{m}\sum_{i=1}^{n_l}\left(w_{l,i}\cdot\prod_{k=1}^{m}\prod_{\substack{j=1\\B^k_j\neq B^l_i}}^{n_k}(B^k_j)^2\right)+w\prod_{l=1}^{m}\prod_{i=1}^{n_l}(B^l_i)^2=0.\end{aligned} \tag{8}$$

What this means is that each corner is multiplied with the product of all bounding surfaces (regardless of which loop they are in) unless they coincide with one of its neighbors. Side components for a given side are the product of all bounding

surfaces, other than those coinciding with the bounding surface representing that side. This formulation keeps the properties highlighted about the original equation (G^1 continuity to corner interpolants, and sides being only affected by the corresponding coefficient).

A practical implementation for evaluating this is to store all bounding surfaces in an array, and only store indices for them in the loops. When computing the products, we only have to check for the non-equality of the indices.

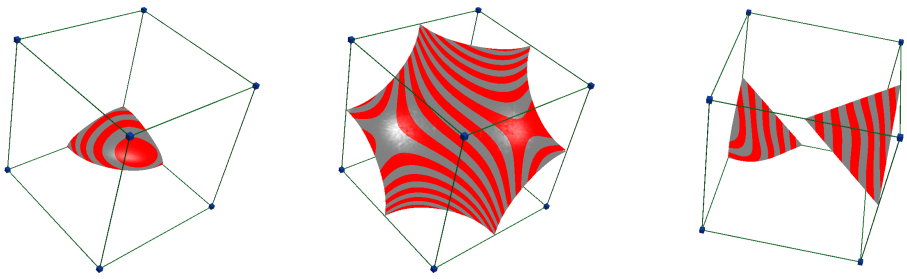
6 Use in cell structures

When used in regular cell structures, the $S_{i,j}$ and B_i surfaces are all planes. Thus, the patch itself is a polynomial surface, with a degree of twice the number of sides (see Equation 2).

In Figures 3 and 4 corner patches are defined inside the unit cube. Figure 3a is a 3-sided surface near a corner of the cube. Figure 3b is a 6-sided patch that intersects all faces of the cube.

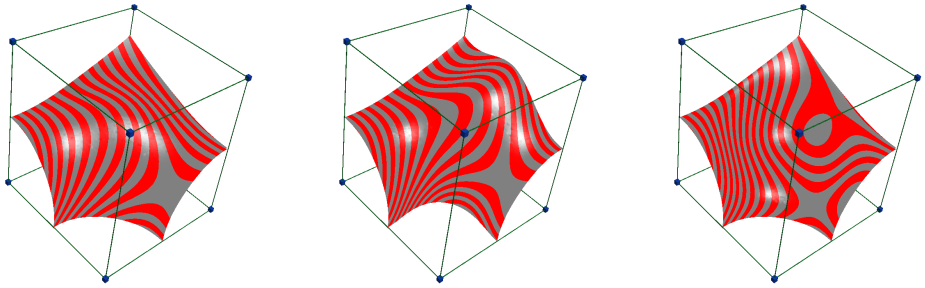
In Figure 4, three patches with the same corners but different coefficients can be seen. They are set using the algorithm presented in Section 4.3, i.e. on each side and in the interior one point is fixed and respective coefficients are calculated from them. Between Figures 4a and 4b, two side points; between Figure 4a and 4c, the interior point is changed. The numerical data for these patches can be found in Table 1.

The possible topological configurations are similar to those of Marching Cubes [12]. The multiloop scheme also works for topologically disjoint isosurfaces (Figure 3c).



(a) A 3-sided patch (b) A 6-sided patch (c) A patch consisting of two disjoint components

Figure 3: Corner I-patches inside the unit cube



(a) Base patch (b) Changing boundaries (c) Changing the interior

Figure 4: Corner I-patches with the same corner interpolants and different coefficients

Table 1: Points and coefficients for the patches in Figure 4. All patches are in the $[0;1]^3$ cube. Differences from Patch #1 are **bold**.

Corner points				
$[0, 0.5, 1]$	$[0, 0.8, 0]$	$[1, 0.5, 0]$	$[1, 0, 0.5]$	$[0.5, 0, 1]$
Patch #1				
Side points				
$[0, 0.6, 0.5]$	$[0.5, 0.6, 0]$,	$[1, 0.35, 0.35]$	$[0.65, 0, 0.65]$	$[0.35, 0.35, 1]$
Side coefficients				
0.6	0.6	-2.34	2.34	-2.34
Interior point:	$[0.5, 0.5, 0.5]$	Central coefficient:		-7.77
Patch #2				
Side points				
$[0, 0.6, 0.5]$	$[0.5, \mathbf{0.8}, 0]$	$[1, 0.35, 0.35]$	$[\mathbf{0.55}, 0, \mathbf{0.55}]$	$[0.35, 0.35, 1]$
Side coefficients				
0.6	**2.2**	-2.34	**0.45**	-2.34
Interior point:	$[0.5, 0.5, 0.5]$	Central coefficient:		$\mathbf{-8.94}$
Patch #3				
Side points				
$[0, 0.6, 0.5]$	$[0.5, 0.6, 0]$,	$[1, 0.35, 0.35]$	$[0.65, 0, 0.65]$	$[0.35, 0.35, 1]$
Side coefficients				
0.6	0.6	-2.34	2.34	-2.34
Interior point:	$[0.5, \mathbf{0.2}, 0.5]$	Central coefficient:		**55.83**

7 Discussion

7.1 Methodology

The examples were generated in the following way. The input was a voxel array of floating point isovalues. Then, for each cell, based on the eight isovalues in the corner, polyline loops were generated from the estimated edge intersections. Ambiguities (when on a face all four edges have an intersection) were resolved by minimizing the sum of distances between the two pairs. This resulted in one or more loops of closed polylines.

Then, in each of the isovertices, a plane was introduced, with its normal pointing towards the positive cell corner. From those, and the cube's faces, corner I-patches were generated. Coefficients for the corner components were set to 1, and side coefficients were calculated with the triangle rule or the tetragon rule (see Appendix A). The central coefficient was set to 0.

Rendering was done using raycasting, neighboring patches have an alternating texture color. Phong shading is used, and normal vectors are calculated from the exact gradients of the surface.

7.2 Single cell examples

The flexibility of the corner I-patch representation can be shown in Figure 5 where patches are generated from the corner sign settings corresponding to the 14 non-

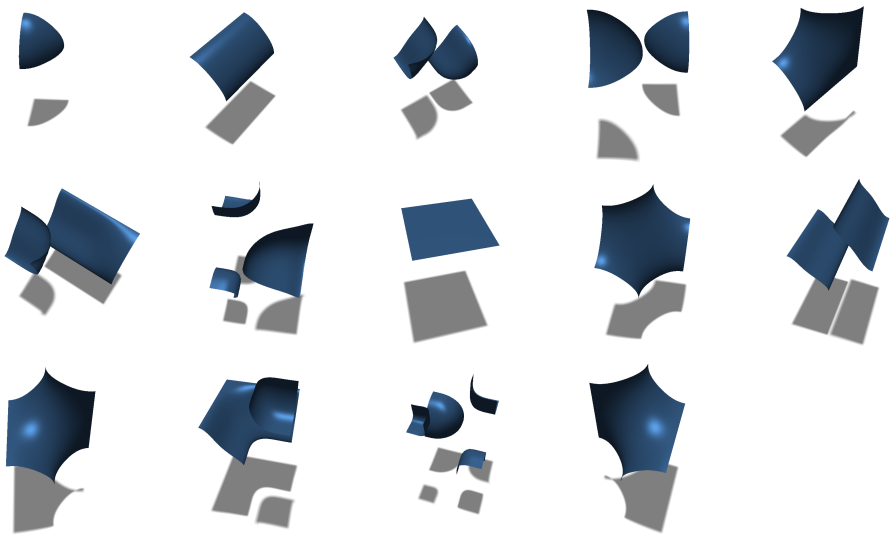


Figure 5: The 14 non-empty basic configurations of Marching Cubes represented with corner I-patches (in the same order as in [12])

empty basic configurations of Marching Cubes. The patches were created using the formulation in Equation 8, as in cases where there were two boundary curves on the same side of the cube, boundary surfaces coincided.

7.3 Multi-cell examples

In these examples, a sparse (9×9×9) voxel array was created and the patches were automatically generated from it. In the first example (Figure 6) two sphere-like objects can be seen which are close to each other. Notice that the brown surfaces at their closest points are represented with the same corner I-patch. In the second example (Figure 7) a voxel value was modified, extending the volume of the bigger object. In Figure 8 a hole was put into the object by modifying isovalues in a line.

7.4 Examples with exact vertices and gradients

Here, the scheme was modified so that when introducing the corner planes, an exact implicit function is used for both exactly calculating the isovertex and using an exact gradient. This can be useful in cases where evaluating the original functions would be very costly but approximating them with piecewise polynomial surfaces could bring a reduction in both storage and computation costs.

In Figures 9 and 10 an ellipsoid can be seen with a lower and a higher resolution cell structure. It can be observed that although the boundary curves approximate

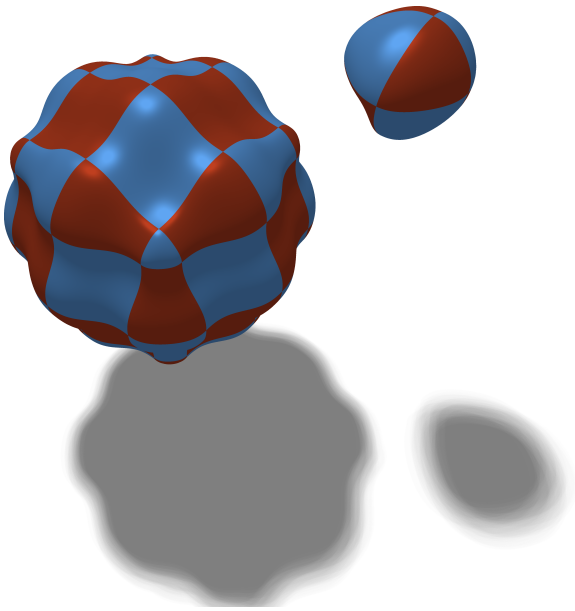


Figure 6: Disjoint surfaces generated from voxel data

the original surface well, the interior of some surfaces is less smooth. Optimization of the coefficients is therefore an important area of future research.

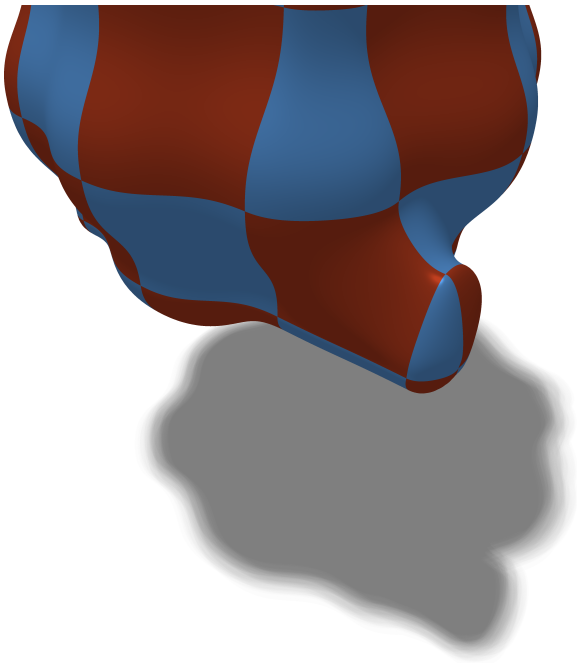


Figure 7: Enlarging the object by modifying a voxel value

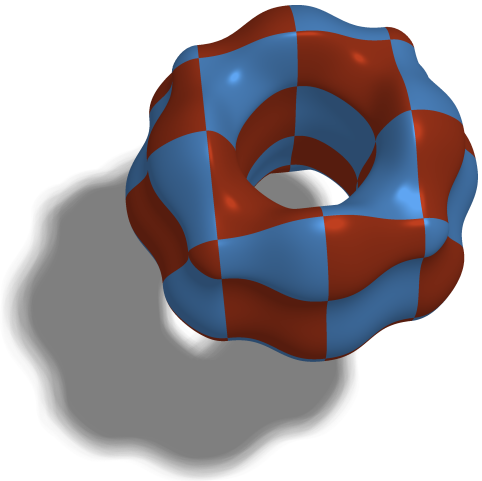


Figure 8: Putting a hole inside the object by modifying voxel values

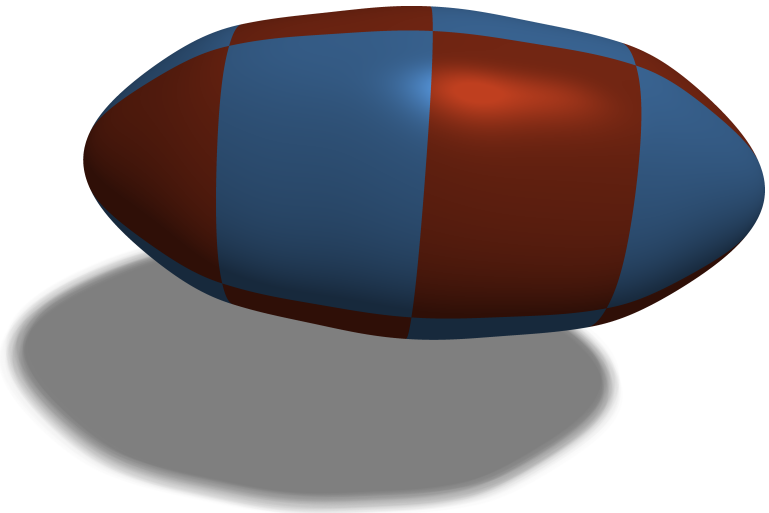


Figure 9: Ellipsoid with lower resolution

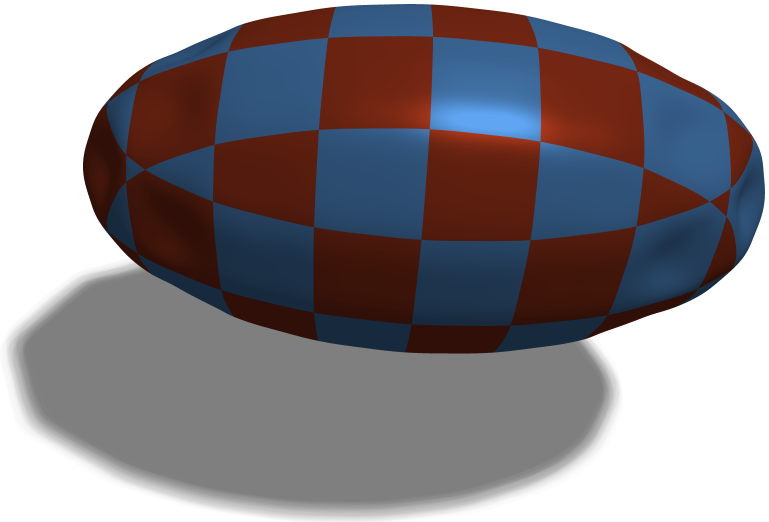


Figure 10: Ellipsoid with higher resolution

8 Conclusion

We have presented corner I-patches, a class of implicit surfaces with several advantages over existing representations. Patches can be defined by combining only planes, in contrast to the relatively more complicated ribbons needed to define I-patches. They can be used to create complex piecewise surfaces. Unlike I-patches,

corner I-patches have no singularities in the corners. Several scalar coefficients can be used to optimize a target function on the patch for approximation or surface fairing purposes.

The figures in the paper have been produced by raytracing, however, an effective implementation (possibly a GPU one) can be an important goal. Finding good target functions for optimizing the coefficients is a possible area of improvement. Detecting poor-quality patches and automatically adjusting the surface coefficients would also enhance the scheme.

Acknowledgements

This research would not have been possible without the help of Tamás Várady who guided the author for several years and raised many important issues. The author would like to also thank the interesting discussions with Alyn Rockwood, Péter Salvi, and Márton Vaitkus as well as the interesting questions raised by the audience of the CSCS 2022 conference.

References

[1] Bajaj, Chandrajit L., Chen, Jindon, and Xu, Guoliang. Modeling with cubic A-patches. *ACM Transactions on Graphics (TOG)*, 14(2):103–133, 1995. DOI: 10.1145/221659.221662.

[2] Bajaj, Chandrajit L. and Ihm, Insung. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics (TOG)*, 11(1):61–91, 1992. DOI: 10.1145/102377.120081.

[3] Bán, Róbert and Valasek, Gábor. First order signed distance fields. In *Eurographics (Short Papers)*, pages 33–36, 2020. DOI: 10.2312/egs.20201011.

[4] Charrot, Peter and Gregory, John A. A pentagonal surface patch for computer aided geometric design. *Computer Aided Geometric Design*, 1(1):87–94, 1984. DOI: 10.1016/0167-8396(84)90006-2.

[5] Chávez, Gustavo and Rockwood, Alyn. Marching surfaces: Isosurface approximation using G^1 multi-sided surfaces. *arXiv preprint arXiv:1502.02139*, 2015. DOI: 10.48550/arXiv.1502.02139.

[6] Hartmann, Erich. Implicit G^n-blending of vertices. *Computer Aided Geometric Design*, 18(3):267–285, 2001. DOI: 10.1016/S0167-8396(01)00030-9.

[7] Krasauskas, Rimvydas. Toric surface patches. *Advances in Computational Mathematics*, 17(1):89–113, 2002. DOI: 10.1023/A:1015289823859.

[8] Lekien, Francois and Marsden, Jerrold. Tricubic interpolation in three dimensions. *International Journal for Numerical Methods in Engineering*, 63(3):455–471, 2005. DOI: 10.1002/nme.1296.

[9] Li, Jinggong, Hoschek, Josef, and Hartmann, Erich. G^{n-1}-functional splines for interpolation and approximation of curves, surfaces and solids. *Computer Aided Geometric Design*, 7(1-4):209–220, 1990. DOI: 10.1016/0167-8396(90)90032-M.

[10] Liming, Roy A. Conic lofting of streamline bodies: The basic theory of a phase of analytic geometry applicable to aircraft. *Aircraft Engineering and Aerospace Technology*, 19(7):222–228, 1947. DOI: 10.1108/eb031528.

[11] Loop, Charles T. and DeRose, Tony D. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics*, 8(3):204–234, 1989. DOI: 10.1145/77055.77059.

[12] Lorensen, William E. and Cline, Harvey E. Marching Cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987. DOI: 10.1145/37402.37422.

[13] Sipos, Ágoston, Várady, Tamás, and Salvi, Péter. Approximating triangular meshes by implicit, multi-sided surfaces. *Computer-Aided Design and Applications*, 19(5):1015–1028, 2022. DOI: 10.14733/cadaps.2022.1015-1028.

[14] Sipos, Ágoston, Várady, Tamás, Salvi, Péter, and Vaitkus, Márton. Multi-sided implicit surfacing with I-patches. *Computers & Graphics*, 90:29–42, 2020. DOI: 10.1016/j.cag.2020.05.009.

[15] Vad, Viktor, Csébfalvi, Balázs, Rautek, Peter, and Gröller, Eduard. Towards an unbiased comparison of CC, BCC, and FCC lattices in terms of prealiasing. *Computer Graphics Forum*, 33(3):81–90, 2014. DOI: 10.1111/cgf.12364.

[16] Várady, Tamás, Benkő, Pál, Kós, Géza, and Rockwood, Alyn. Implicit surfaces revisited – I-patches. In *Geometric Modelling*, pages 323–335. Springer, 2001. DOI: 10.1007/978-3-7091-6270-5_19.

[17] Várady, Tamás, Rockwood, Alyn, and Salvi, Péter. Transfinite surface interpolation over irregular n-sided domains. *Computer Aided Design*, 43(11):1330–1340, 2011. DOI: 10.1016/j.cad.2011.08.028.

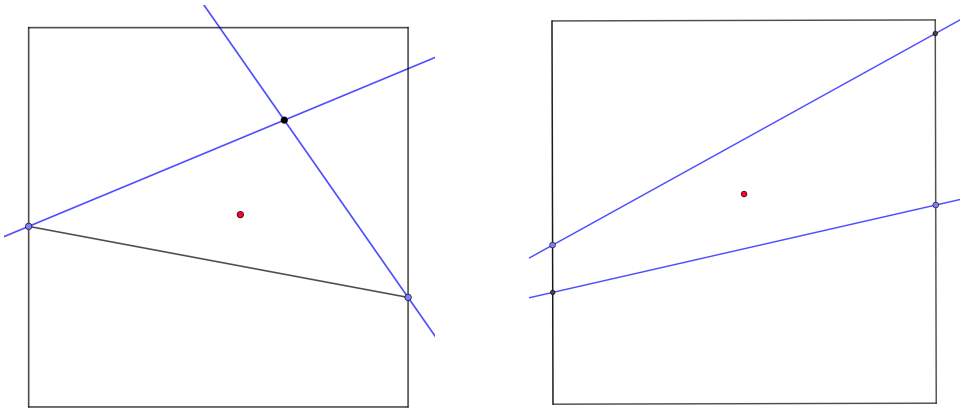
[18] Várady, Tamás, Salvi, Péter, and Karikó, György. A multi-sided Bézier patch with a simple control structure. *Computer Graphics Forum*, 35(2):307–317, 2016. DOI: 10.1111/cgf.12833.

A Auxiliary point calculation

Side coefficients in the examples are calculated such that the boundary curve interpolates a given point. That point is computed with either the triangle rule or the tetragon rule.

There are two cases: the two corner planes and the bounding face either have an intersection point inside the face, or it does not. If that point exists, we have a

triangle (Figure 11a) and we take its centroid as the point to interpolate. Otherwise, we have a tetragon (Figure 11b), by intersecting each corner plane with the opposite corner's cube edge. We then take the centroid of this polygon.



(a) Triangle rule (b) Tetragon rule

Figure 11: Rules for computing interpolated points. Blue points and lines represent corner points and planes.

The side coefficient can then easily be calculated by evaluating Equation 5 for each w_i, as the other side coefficients do not affect the current boundary.