

Uniform Powell–Sabin splines for the polygonal hole problem

Joris Windmolders
Paul Dierck

Report TW 328, September 2001



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Uniform Powell–Sabin splines for the polygonal hole problem

Joris Windmolders

Paul Dierck

Report TW 328, September 2001

Department of Computer Science, K.U.Leuven

Abstract

An algorithm for smoothly filling in a polygonal hole in a surface, with a parametric uniform Powell–Sabin spline surface, is described. It uses interpolation and subdivision techniques for iteratively determining an approximating solution. No assumptions are made about the surrounding surface. The user has to provide routines for calculating the curve points and the unit surface normal along the edge, as well as the unit tangent vector of the edge curves, parametrized on the unit interval. The method is illustrated by some examples.

Keywords : PS-spline, polygonal holes, n-sided patches.

Uniform Powell–Sabin splines for the polygonal hole problem

Joris Windmolders, Paul Dierckx

Abstract

An algorithm is described for smoothly filling in a polygonal hole in a surface, with a parametric uniform Powell–Sabin spline surface patch. It uses interpolation and subdivision techniques for iteratively determining an approximating solution. No assumptions are made about the surrounding surface. The user has to provide routines for calculating the curve points and the unit surface normal along the edge, as well as the unit tangent vector of the edge curves, parametrized on the unit interval.

1 Introduction

A classical problem in CAGD is to fill in a hole, bounded by a set of surfaces. This problem has already been addressed in the literature. A number of authors determine a patch by blending surfaces that meet the boundary conditions at certain curves: for example in [1], a pentagonal surface patch, joining adjacent rectangular patches with C^1 -continuity is presented; whereas in [4] an n -sided patch is found by extending a given C^k patch complex around the hole. For the case of cubic boundary curves, [2] calculates a C^1 cubic triangular spline patch, interpolating the curves exactly and approximating the boundary derivatives on the hole. By energy minimization, the bumpiness of the patch is reduced. A solution with a G^1 subdivision surface patch appeared in [5]. In this paper, we present an algorithm for filling in a 3, 4, 5 or 6-sided hole. Our algorithm differs from most of the solutions in the literature in that it makes no assumptions on the surrounding surfaces, and therefore it is generally applicable. On the other hand, the filling patch will meet the given boundary curves approximately. The input of our algorithm (see figure 1) consists of the boundary curves \mathbf{p} which join at their endpoints. Furthermore, the user should provide the unit tangent vector $\vec{\gamma}$ to the boundary curves at any point, and the unit normal vector \vec{n} to the surrounding surface at any curve point except the endpoints, where the tangent vectors of

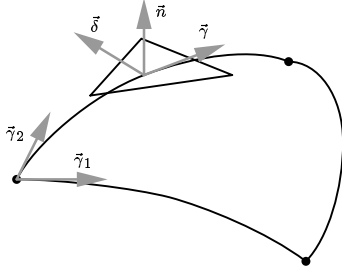


Figure 1: User supplied data

the joining curves are needed only (see figure 1 again). For other (inner) curve points, our algorithm will calculate a unit vector $\vec{\delta} = \vec{n} \times \vec{\gamma}$. It will be denoted the (unit) cross-boundary tangent vector. It shall be referred to as if it were provided by the user. We will calculate a filling surface patch that interpolates the user supplied boundary curves and has the same surface normal in a number of points. This will leave us some degrees of freedom, which we will use to fit the curve and the cross-boundary tangent vector in between each pair of interpolation points. In section 2 we briefly recall the basic properties of uniform Powell–Sabin splines. Section 3 explains how we can benefit from these properties to use UPS–splines for the polygonal hole problem. Section 4 explains our algorithm in detail. Finally we give some concluding remarks in section 5.

2 Uniform Powell-Sabin splines

This section recalls the main properties of Uniform Powell-Sabin splines. For details, we refer to the original papers [3, 6].

By $S_2^1(\Delta^*)$ we denote the linear space of uniform Powell–Sabin splines (in the sequel called UPS–splines), i.e., piecewise quadratic polynomials on a uniform triangulation Δ of a polygon Ω , where Δ^* is a PS–refinement of Δ . The boundary of Ω will be denoted $\delta\Omega$, whereas the boundary of the triangulation will be referred to as $\delta\Delta$. The vertices of Δ are denoted $V_i, i = 1, \dots, n$, and its triangles are $\rho_i, i = 1, \dots, m$. These splines have global C^1 –continuity on Δ^* . Any $\mathbf{s}(u, v)$ has a unique B–spline representation

$$\mathbf{s}(u, v) = \sum_{i=1}^n \sum_{j=1}^3 \mathbf{c}_{i,j} B_i^j(u, v), \quad (u, v) \in \Omega, \quad (1)$$

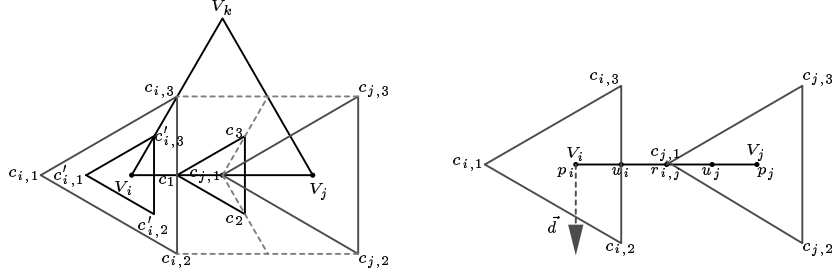


Figure 2: Subdivision and Bézier points

where the basis functions satisfy

$$\begin{cases} B_i^j(u, v) \geq 0 \\ \sum_{i=1}^n \sum_{j=1}^3 B_i^j(u, v) \equiv 1, \end{cases} \quad (2)$$

and $\mathbf{c}_{i,j} \in \mathbf{R}^3$ are the control points. It follows that $\mathbf{s}(u, v)$ belongs to the convex hull of $\{\mathbf{c}_{i,j}\}_{i,j}$. Furthermore, one can prove that the control triangles, being defined as $T_i(\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \mathbf{c}_{i,3}), i = 1, \dots, n$, are tangent to the surface at $\mathbf{s}(V_i)$. The basis functions have local support, so we have local control as well: a change to $\mathbf{c}_{i,j}$ will only affect $\mathbf{s}(u, v)|_{M_i}$, i.e., the restriction of $\mathbf{s}(u, v)$ to the molecule of V_i , being the set of triangles ρ_j that have V_i as a vertex. This indicates that we have a useful representation for representing C^1 -continuous surfaces, without being restricted to a rectangular domain, and still enjoying the interesting features of the classical B-spline representation for tensor product splines.

A note on the pictures. It is clear that while the vertices V_i are 2D-points in the parameter domain, the control points $\mathbf{c}_{i,j}$ live in 3D-space. However, we will sometimes use both notations in the same figure. The pictures therefore only reflect the situation schematically, denoting which control points are associated with each vertex or edge in the domain triangulation Δ . Further we will also introduce vectors γ_i and δ_i lying in plane of the control triangle T_i . These will also be drawn schematically in the domain picture.

2.1 Subdivision

In [6] we present a subdivision scheme for UPS-splines. Let Δ_r be a uniform refinement of Δ , obtained by midedge subdivision. For a given $\mathbf{s}(u, v)$ on Δ , the representation (1) on Δ_r can be calculated using convex barycentric combinations of the control points only. First, a

new control triangle along each edge $V_i V_j$ is calculated as illustrated in figure 2, left, for the bottom edge of a triangle $\rho_l(V_i, V_j, V_k) \in \Delta$:

$$\begin{cases} \mathbf{c}_1 &= \frac{1}{2}(\mathbf{c}_{i,2} + \mathbf{c}_{i,3}) \\ \mathbf{c}_2 &= \frac{1}{2}\mathbf{c}_{j,1} + \frac{1}{4}(\mathbf{c}_{1,2} + \mathbf{c}_{j,2}) \\ \mathbf{c}_3 &= \frac{1}{2}\mathbf{c}_{j,1} + \frac{1}{4}(\mathbf{c}_{i,3} + \mathbf{c}_{j,3}). \end{cases} \quad (3)$$

Next, the control triangles at the original vertices are rescaled: for example,

$$\begin{cases} \mathbf{c}'_{i,1} &= \frac{2}{3}\mathbf{c}_{i,1} + \frac{1}{6}(\mathbf{c}_{i,2} + \mathbf{c}_{i,3}) \\ \mathbf{c}'_{i,2} &= \frac{2}{3}\mathbf{c}_{i,2} + \frac{1}{6}(\mathbf{c}_{i,3} + \mathbf{c}_{i,1}) \\ \mathbf{c}'_{i,3} &= \frac{2}{3}\mathbf{c}_{i,3} + \frac{1}{6}(\mathbf{c}_{i,1} + \mathbf{c}_{i,2}). \end{cases} \quad (4)$$

They are still tangent to the surface at their barycenter, but their area is only a quarter that of the former control triangles. Therefore they connect tighter to the surface.

2.2 The piecewise Bézier representation

Another important property of the B-spline representation for UPS-splines, is that the piecewise Bézier representation can be calculated from (1) using simple convex barycentric combinations of the control points. In particular, focus an edge $V_i V_j$ of Δ (see figure 2, right). The Bézier points of the edge curve can be found from:

$$\mathbf{s}(V_i) = \mathbf{p}_i = \frac{1}{3}(\mathbf{c}_{i,1} + \mathbf{c}_{i,2} + \mathbf{c}_{i,3}) \quad (5)$$

$$\mathbf{s}(V_j) = \mathbf{p}_j = \frac{1}{3}(\mathbf{c}_{j,1} + \mathbf{c}_{j,2} + \mathbf{c}_{j,3}) \quad (6)$$

$$\mathbf{u}_i = \frac{1}{2}(\mathbf{c}_{i,2} + \mathbf{c}_{i,3}) \quad (7)$$

$$\mathbf{u}_j = \frac{2}{3}\mathbf{c}_{j,1} + \frac{1}{6}(\mathbf{c}_{j,2} + \mathbf{c}_{j,3}) \quad (8)$$

$$\mathbf{r}_{i,j} = \frac{1}{2}(\mathbf{u}_i + \mathbf{u}_j). \quad (9)$$

This is a piecewise quadratic Bézier curve, which means that \mathbf{p}_i , $\mathbf{r}_{i,j}$ and \mathbf{p}_j are surface points, and that $\mathbf{u}_i - \mathbf{p}_i$ and $\mathbf{p}_j - \mathbf{u}_j$ are tangent to the surface at \mathbf{p}_i , resp. \mathbf{p}_j . Assuming a (counterclockwise) ordering of the boundary vertices $V_i \in \delta\Delta$, the edge curve from $\mathbf{s}(V_i)$ to the next adjacent point $\mathbf{s}(V_j)$ will be denoted $\mathbf{e}_i(u, v)$.

3 Application to the polygonal hole problem

Recall that our goal is to calculate a UPS-spline filling a hole in a surface, given by a set of bounding curves (denoted \mathbf{p}), their derivatives

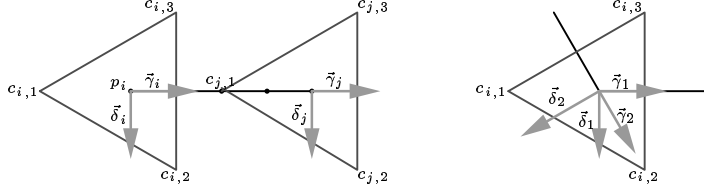


Figure 3: Tangent and binormal vectors

$\vec{\gamma}$ and the cross-boundary tangent vectors $\vec{\delta}$. The UPS-patch will fit these curves approximately along its boundary. In the first place, interpolation of the given data at the vertices $V_i \in \delta\Delta$ is achieved. This leaves some degrees of freedom allowing to fit the given curves. In the sequel we shall denote the user supplied data, evaluated at V_i , by $(\mathbf{p}_i, \vec{\gamma}_i, \vec{\delta}_i)$.

3.1 Interpolating UPS-splines and degrees of freedom

In order to obtain interpolation we determine a control triangle T_i in the tangent plane spanned by $\mathbf{p}_i + \epsilon \vec{\gamma}_i + \nu \vec{\delta}_i$, $\epsilon, \nu \in \mathbf{R}$, such that $\mathbf{s}(V_i) = \mathbf{p}_i$. Curve point interpolation is simply expressed by (5). Furthermore, we let the tangent to \mathbf{e}_i at V_i be parallel to $\vec{\gamma}_i$:

$$\mathbf{u}_i - \mathbf{p}_i = \frac{1}{6}(\mathbf{c}_{i,2} + \mathbf{c}_{i,3}) - \frac{1}{3}\mathbf{c}_{i,1} = \alpha_i \vec{\gamma}_i, \quad (10)$$

where α_i is a scaling factor. Next, we need the cross-boundary tangent vector of $\mathbf{s}(u, v)$ at V_i to be parallel to $\vec{\delta}_i$. Mapping the cross-boundary vector \vec{d} in the domain plane (see figure 2, right) onto the control triangle yields a vector parallel with $\mathbf{c}_{i,2} - \mathbf{c}_{i,3}$:

$$\mathbf{c}_{i,2} - \mathbf{c}_{i,3} = 2\beta_i \vec{\delta}_i, \quad (11)$$

where β_i is again a scaling factor.

Solving (5), (10) and (11) to $\mathbf{c}_{i,j}$ in terms of the unknown α_i and β_i (further called the α - and β -factors) yields

$$\begin{cases} \mathbf{c}_{i,1} &= \mathbf{p}_i - \alpha_i \vec{\gamma}_i \\ \mathbf{c}_{i,2} &= \mathbf{p}_i + \frac{\alpha_i}{2} \vec{\gamma}_i + \beta_i \vec{\delta}_i \\ \mathbf{c}_{i,3} &= \mathbf{p}_i + \frac{\alpha_i}{2} \vec{\gamma}_i - \beta_i \vec{\delta}_i. \end{cases} \quad (12)$$

These equations ensure that $\mathbf{s}(u, v)$ interpolates the given data at $V_i \in \delta\Delta$, and leaves us two degrees of freedom per vertex (α_i and β_i). These scaling factors are related to the size of the control triangle. For example, subdivision by (4) divides α_i and β_i by a factor of 2.

3.2 The fitting equations

We will now use these degrees of freedom to fit the user supplied data, in between each pair of adjacent interpolating vertices $V_i, V_j \in \delta\Delta$. First, the α -factors at V_i and V_j are determined by trying to interpolate the curve \mathbf{p} at the edge midpoint $V_{i,j} = \frac{1}{2}(V_i + V_j)$. From section 2.2, the interpolation condition reads $\mathbf{r}_{i,j} = \frac{1}{2}(\mathbf{u}_i + \mathbf{u}_j) = \mathbf{p}_{i,j}$, where $\mathbf{p}_{i,j}$ is the given curve point. Taking (7–8) and (12) into account, we have

$$\alpha_i \vec{\gamma}_i - \alpha_j \vec{\gamma}_j = 4\mathbf{p}_{i,j} - 2(\mathbf{p}_i + \mathbf{p}_j) = \mathbf{q}_{i,j}. \quad (13)$$

This is a system of 3 equations with (at most) 2 unknowns. It can be solved in the least squares sense.

Next, the β -factors at V_i and V_j are obtained by fitting the cross-boundary tangent vector at $V_{i,j}$. First, we derive a subdivision rule for the β -factors at the vertices of Δ from (3) and (11):

$$\beta'_{i,j} \vec{\delta}'_{i,j} = \frac{1}{4}(\beta_i \vec{\delta}_i + \beta_j \vec{\delta}_j), \quad (14)$$

where $\vec{\delta}'_{i,j}$ is the cross-boundary tangent vector to $\mathbf{s}(u, v)$ at $V_{i,j}$. This $\beta'_{i,j}$ -factor belongs to a finer subdivision level than β_i and β_j , so we have to scale it up by a factor of 2. The interpolation condition then is

$$\beta_{i,j} \vec{\delta}_{i,j} = \frac{1}{2}(\beta_i \vec{\delta}_i + \beta_j \vec{\delta}_j). \quad (15)$$

Note that $\vec{\delta}_{i,j}$ has been used instead of $\vec{\delta}'_{i,j}$. This is again an over-determined system which can be solved in the least squares sense.

4 The algorithm

We will restrict the figures illustrating the algorithm to the case of a triangular hole, although the algorithm is immediately applicable to cases with 4, 5 and 6 boundary curves as well (see section 4.5).

The idea is to calculate, during a pre-iteration step, an initial solution which is smooth, but in general not close enough, and to refine this approximation iteratively to obtain a better fit to the given curves. This refinement phase operates on the boundary of Ω only. At each refinement step, the control triangles from the previous steps are rescaled, and new ones are calculated in between them, using the interpolation equation. After the last iteration, we have a solution which is close enough on $\delta\Omega$. Finally, during a post-iteration step, the inner control triangles are calculated, actually filling the hole. Figure 4 illustrates this: imagine a pre-iteration step, two refinement steps and a post-iteration step. The control triangles added during a particular step have been shaded. This is a sketch of the algorithm:

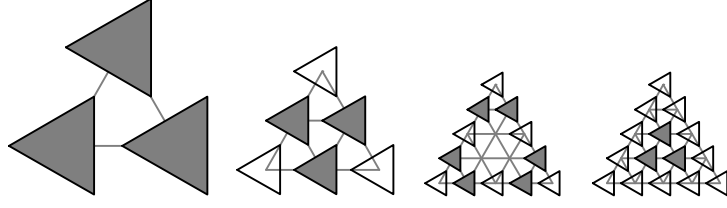


Figure 4: Consecutive iteration steps

Calculate initial solution
 While approximation is not good enough on boundary
 Refine approximation on boundary
 Calculate the inner of the patch

4.1 An initial solution

The initial solution (figure 4, leftmost) is easily obtained by solving (13) in the least squares sense for each edge V_iV_j . If we assume that $\vec{\gamma}_i \neq \vec{\gamma}_j$, then

$$\alpha_i = \frac{1}{D} ((\vec{\gamma}_i \cdot \mathbf{q}_{i,j}) - (\vec{\gamma}_j \cdot \mathbf{q}_{i,j})(\vec{\gamma}_i \cdot \vec{\gamma}_j)) \quad (16)$$

$$\alpha_j = \frac{1}{D} (-(\vec{\gamma}_j \cdot \mathbf{q}_{i,j}) + (\vec{\gamma}_i \cdot \mathbf{q}_{i,j})(\vec{\gamma}_i \cdot \vec{\gamma}_j)), \quad (17)$$

where $D = 1 - (\vec{\gamma}_i \cdot \vec{\gamma}_j)^2$. This yields two α -factors per vertex: one for each boundary edge being incident to that vertex. Therefore, no degrees of freedom are left. The β -factors are already determined completely by the α -factors. Recall from figure 1 that the tangent plane at the vertices of $\delta\Omega$ is already defined by $\vec{\gamma}_1$ and $\vec{\gamma}_2$. Figure 3, right, shows a more detailed example of this situation. The scaling factors of the unit tangent vectors $\vec{\gamma}_1$ and $\vec{\gamma}_2$ are known. The β -factors are found from writing (12) for both edges incident with the vertex and eliminating \mathbf{c}_2 , resp. \mathbf{c}_1 :

$$\beta_1 = \alpha_2(\vec{\gamma}_2 \cdot \vec{\delta}_1) \quad (18)$$

$$\beta_2 = -\alpha_1(\vec{\gamma}_1 \cdot \vec{\delta}_2). \quad (19)$$

There exist pathological cases where $\vec{\gamma}_2 \perp \vec{\delta}_1$ or $\vec{\gamma}_1 \perp \vec{\delta}_2$. Our algorithm then sets $\beta_1 = \alpha_1$, resp. $\beta_2 = \alpha_2$. For the case $\vec{\gamma}_i = \vec{\gamma}_j$, (13) has no solution in the least-squares sense. Assuming that \mathbf{s}_i is a straight line from $\mathbf{s}(V_i)$ to $\mathbf{s}(V_j)$, the α -factors can then be determined from the projection onto the domain plane, where the size of the so-called PS-triangles (the projections of the control triangles) is fixed. The reader can verify that this yields $\alpha_i = \alpha_j = \frac{1}{2}|V_iV_j|$.

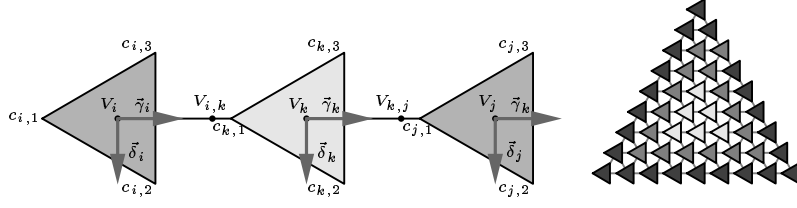


Figure 5: The refinement and post-iteration steps

4.2 The iteration step

First the control triangles from the previous steps are rescaled by subdivision. This is simply done by scaling down the α - and β -factors: $\alpha_i \leftarrow \frac{\alpha_i}{2}$ and $\beta_i \leftarrow \frac{\beta_i}{2}$, for each $V_i \in \delta\Delta$.

Next, a new control triangle is created in between any two adjacent vertices at the coarser level. This situation is illustrated in figure 5, left, where the darker triangles are known. We are looking for the α - and β -factors for the middle control polygon, which is tangent to the surface at $\mathbf{s}(V_k)$, $V_k = \frac{1}{2}(V_i + V_j)$. Consider the α -factor first. In order to obtain a better fit, we try to interpolate \mathbf{p} at $V_{i,k} = \frac{1}{2}(V_i + V_k)$ and $V_{k,j} = \frac{1}{2}(V_k + V_j)$. This yields a set of fitting equations

$$\begin{cases} \alpha_i \vec{\gamma}_i - \alpha_k \vec{\gamma}_k = \mathbf{q}_{i,k} \\ \alpha_k \vec{\gamma}_k - \alpha_j \vec{\gamma}_j = \mathbf{q}_{k,j} \end{cases} \quad (20)$$

where α_i and α_j are known. Thus, α_k can be obtained as the least-squares solution of (20):

$$\alpha_k = \frac{1}{2}(\vec{\gamma}_k \cdot (\alpha_i \vec{\gamma}_i - \mathbf{q}_{i,k} + \mathbf{q}_{k,j} - \alpha_j \vec{\gamma}_j)). \quad (21)$$

The β_k -factor is found by fitting the cross-boundary vectors at $V_{i,k}$ and $V_{k,j}$, i.e., by solving the following system in the least-squares sense:

$$\begin{cases} \beta_{i,k} \vec{\delta}_{i,k} = \frac{1}{2}(\beta_i \vec{\delta}_i + \beta_k \vec{\delta}_k) \\ \beta_{k,j} \vec{\delta}_{k,j} = \frac{1}{2}(\beta_k \vec{\delta}_k + \beta_j \vec{\delta}_j), \end{cases} \quad (22)$$

where β_i and β_j are known. If $\vec{\delta}_{i,k} = \vec{\delta}_k = \vec{\delta}_{k,j}$, as is always the case for a planar curve, this system has no solution in the least-squares sense. The β_k factor can then easily be obtained by equation (15), i.e., by subdivision and upscaling.

4.3 The stopping criterion

In our implementation, the user can choose between a fixed number of iterations, or he can iterate until the maximum residual is smaller

then a value ϵ of his choice. Therefore, each time we solve the fitting equations, the residual $r_{i,j} = \|\mathbf{q}_{i,j} - \alpha_i \vec{\gamma}_i + \alpha_j \vec{\gamma}_j\|$ is calculated. The maximum residual for each edge is stored in memory and updated each refinement step. If at the current refinement level, for a certain edge, $r_{i,j}^{\max} < \epsilon$, then the solution on this edge is subdivided during the next refinement steps, instead of solving the interpolation equation again and again.

4.4 The inner control points

Finally, as soon as the user supplied edge curves have been approximated well enough, the inner control points at the eventual refinement level have to be calculated. Our implementation allows the user to choose from three possibilities. These will be discussed with the help of an example. In [7] we derived a NURPS representation for patches on the sphere. Figure 6, left, shows two such patches, constituting a sphere sextant with a hole around the equator. This hole will be filled by two triangular patches, shown in figure 6, right. The common boundary curve is fitted to the great circle bisecting the hole. We now discuss three strategies to obtain the inner control triangles.

Copy From Initial. The inner control points are obtained directly from the initial solution by subdivision. This guarantees that the inner of the patch is smooth. A disadvantage is that the inner of the first approximation in general has no connection with the shape of the edge curves. This can cause unwanted artefacts near the boundary, after a few iterations (see figure 7, left). The next option will therefore take edge features into account.

Averaging. We will fill the hole gradually by calculating a ring of control triangles during each pass, going from the edge towards the inner of the patch. Figure 5, right shows an example where each ring has a different shade of grey. At each step, a control triangle of the current ring is obtained by averaging six surrounding control triangles. These come from the initial solution, or, if possible, from a previously calculated ring. Edge features are now smoothed out towards the inner of the patch. However, there is a main disadvantage to this approach, if averaging is applied after the last iteration step: the unwanted artefacts mentioned before are now repeated for every ring, smoothed out towards the inner of the surface, as shown on figure 7, right.

Instant Update. A good compromise would be to take edge features into account before we finish iterating. This can be accomplished by subdividing the initial solution at each refinement step, but, we always overwrite its edge with the most recent boundary approximation. The results of this strategy are depicted in figure 8.

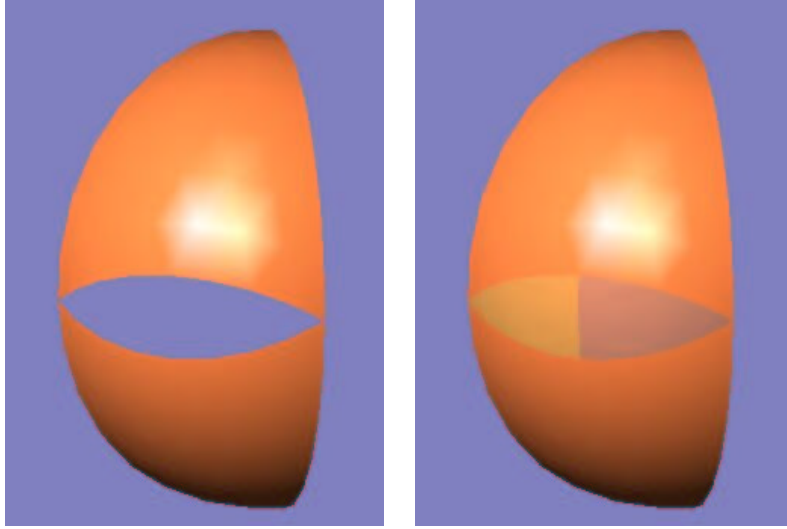


Figure 6: The hole and the triangular patches

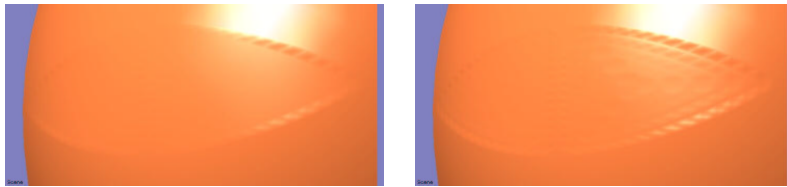


Figure 7: Copy from initial solution and averaging (4 iterations)

The following table shows how the maximum residual diminishes each step.

In any case can the user change the inner control triangles, and still he has a C^1 -continuous filling patch, fitting the specified edge curves with demanded precision.

Iteration	Maximum residual		
	edge 1	edge 2	edge 3
1	0.000372622	3.29594e-006	0.000372622
2	0.000124426	1.03392e-006	0.000124426
3	3.38407e-005	2.72169e-007	3.38407e-005
4	8.72902e-006	6.89716e-008	8.72902e-006



Figure 8: Instant update (3 iterations)

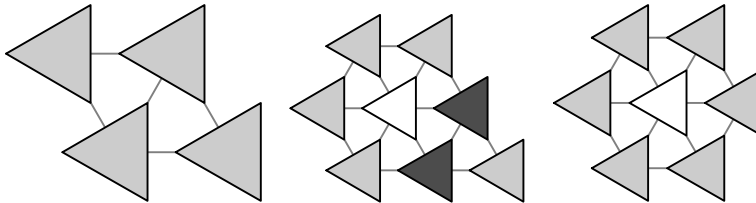


Figure 9: Cases with 4, 5 and 6 boundary curves

4.5 A note on the number of edges

The algorithm sketched in section 4 is immediately applicable to problems with 4, 5 and 6 boundary curves as well. Figure 9 shows the configuration of the initial solution for each of these cases. If we are working with 5 edges, there are 2 edges having a control triangle at its midpoint (shaded darker). This requires a tiny modification to the calculation of the initial solution for those edges. The α -factors are obtained by solving (20) to the unknown α_i, α_j and α_k . The β -factors of the outer control polygons are obtained as usual; for the middle polygon one can apply (15). Also, for the cases of 5 and 6 boundary curves, an inner control triangle (unshaded) has to be calculated for the initial solution. This can be done by averaging the six surrounding control polygons.

5 Conclusion

We gave an algorithm for smoothly filling in a polygonal hole in a surface, with a UPS-spline. The algorithm is generally applicable since it makes no assumptions on the surrounding surfaces. However, it only fits the boundary curves approximately.

The results are already promising, but there remain some questions and future challenges:

- Can we associate a design parameter with the β -factors?
- Are there alternative approaches for calculating the inner of the surface patch, enhancing the smoothness of the filling patch?
- Can we treat cases where not all boundary curves are known?
- Can we use the algorithm when the surrounding surfaces do not connect smoothly?
- Is it possible to have other cases than 3, 4, 5 or 6 edges?

References

- [1] Charrot, P. and A. Gregory, A pentagonal surface patch for computer aided geometric design, *Computer Aided Design* 1, pp 87–94.
- [2] Chui, C. K. and M.-J. Lai (2000), Filling polygonal holes using C^1 cubic triangular spline patches, *Computer Aided Geometric Design* 17, pp 297–307.
- [3] Dierckx, P. (1997), On calculating normalized Powell–Sabin B-splines, *Computer Aided Geometric Design* 15, pp 61–78.
- [4] Gregory, J.A., V. K. H. Lau, and J. M. Hahn (1993) , High order continuous polygonal patches, in *Geometric Modelling*, G. Farin, H. Hagen and H. Noltemeier (eds.), Springer–Verlag Wien.
- [5] Levin, A. (1999), Filling an n-sided hole using combined subdivision schemes, in *Curve and Surface Design: Saint–Malo 1999*, A. Cohen, C. Rabut and L.L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2000.
- [6] Windmolders, J., Dierckx, P. (1999), Subdivision of Uniform Powell–Sabin splines, *Computer Aided Geometric Design* 16, 301–315.
- [7] Windmolders, J. and P. Dierckx, NURPS for Special Effects and Quadrics: Oslo 2000, Tom Lyche and L. L. Schumaker (eds.), Vanderbilt Press, Nashville 2001, to appear.