Uniform Powell–Sabin splines for the polygonal hole problem

Joris Windmolders and Paul Dierckx

Department of Computer Sciences, Kath. University Leuven, Belgium.
Joris.Windmolders@cs.kuleuven.ac.be, Paul.Dierckx@cs.kuleuven.ac.be

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 (e.g. [1, 2, 4]). In most cases, assumptions are made on the bounding surfaces. In this paper, we present an algorithm for filling in a 3, 4, 5 or 6–sided hole that 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 the joining curves are needed only (see Figure 1 again). For other (interior) curve points, our algorithm will calculate a unit vector $\vec{\delta} = \vec{n} \times \vec{\gamma}$, which will be called 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 remark that on the pictures, we will denote 2D and 3D entities interchangebly; therefore most pictures reflect the situation only schematically.

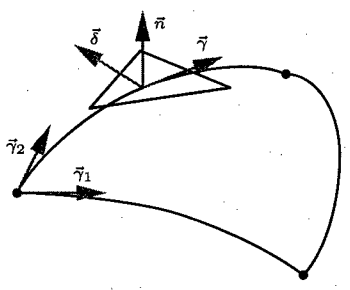

FIG. 1. User supplied data.

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, 5].

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 (which means that all triangles are equilateral and have the same size) of a polygon Ω, where Δ^* is a PS–refinement of Δ. The boundary of Ω will be called $\delta\Omega$, whereas the boundary of the triangulation will be referred to as $\delta\Delta$. The vertices of Δ are denoted $V_i, i = 1, \ldots, n$, and its triangles are $\rho_i, i = 1, \ldots, m$. These splines have global C^1–continuity on Δ^*. Any $\mathbf{s}(\mathbf{u}, \mathbf{v})$ has a unique B–spline representation

$$\mathbf{s}(\mathbf{u}, \mathbf{v}) = \sum_{\mathbf{i}=1}^{\mathbf{n}} \sum_{\mathbf{j}=1}^{\mathbf{3}} \mathbf{c_{i,j}} \mathbf{B_i^j}(\mathbf{u}, \mathbf{v}), \quad (\mathbf{u}, \mathbf{v}) \in \mathbf{\Omega}, \tag{2.1}$$

where the locally supported basis functions form a convex partition of unity and $\mathbf{c_{i,j}} \in \mathbf{R^3}$ are the control points. It follows that $\mathbf{s}(\mathbf{u}, \mathbf{v})$ belongs to the convex hull of $\{\mathbf{c_{i,j}}\}_{i,j}$. Furthermore, one can prove that the control triangles, being defined as $T_i(\mathbf{c_{i,1}}, \mathbf{c_{i,2}}, \mathbf{c_{i,3}})$, $i = 1, \ldots, n$, are tangent to the surface at $\mathbf{s}(\mathbf{V_i})$. Due to the local support of B_i^j, a change to $\mathbf{c_{i,j}}$ will only affect $\mathbf{s}(\mathbf{u}, \mathbf{v})|_{\mathbf{M_i}}$, i.e., the restriction of $\mathbf{s}(\mathbf{u}, \mathbf{v})$ 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 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.

2.1 Subdivision

In [5] we present a subdivision scheme for UPS–splines. Let Δ_r be a uniform refinement of Δ, obtained by midedge subdivision. For a given $\mathbf{s}(\mathbf{u}, \mathbf{v})$ on Δ, the representation (2.1) on Δ_r can be calculated using convex barycentric combinations of the control points only. First, a new control triangle along each edge V_iV_j is calculated as illustrated in

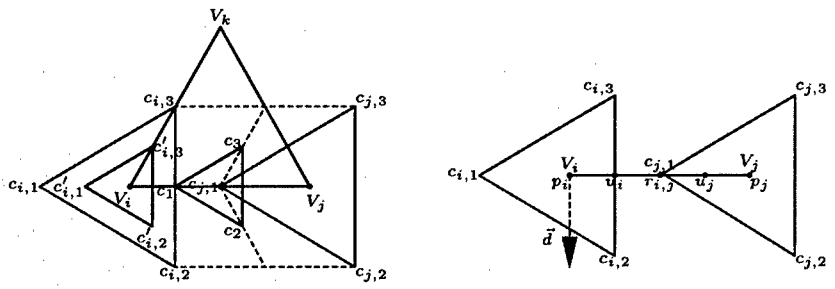

FIG. 2. Subdivision and Bézier points.

Figure 2, left, for the bottom edge of a triangle $\rho_l(V_i, V_j, V_k) \in \Delta$:

$$\begin{cases} \mathbf{c_1} &= \frac{1}{2}(\mathbf{c_{i,2}} + \mathbf{c_{i,3}}) \\ \mathbf{c_2} &= \frac{1}{2}\mathbf{c_{j,1}} + \frac{1}{4}(\mathbf{c_{1,2}} + \mathbf{c_{j,2}}) \\ \mathbf{c_3} &= \frac{1}{2}\mathbf{c_{j,1}} + \frac{1}{4}(\mathbf{c_{i,3}} + \mathbf{c_{j,3}}). \end{cases} \tag{2.2}$$

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

$$\begin{cases} \mathbf{c'_{i,1}} &= \frac{2}{3}\mathbf{c_{i,1}} + \frac{1}{6}(\mathbf{c_{i,2}} + \mathbf{c_{i,3}}) \\ \mathbf{c'_{i,2}} &= \frac{2}{3}\mathbf{c_{i,2}} + \frac{1}{6}(\mathbf{c_{i,3}} + \mathbf{c_{i,1}}) \\ \mathbf{c'_{i,3}} &= \frac{2}{3}\mathbf{c_{i,3}} + \frac{1}{6}(\mathbf{c_{i,1}} + \mathbf{c_{i,2}}). \end{cases} \tag{2.3}$$

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 (2.1) using simple convex barycentric combinations of the control points. In particular, focus an edge V_iV_j of Δ (see Figure 2, right). The Bézier points of the edge curve can be found from:

$$\mathbf{s(V_i)} = \mathbf{p_i} = \frac{1}{3}(\mathbf{c_{i,1}} + \mathbf{c_{i,2}} + \mathbf{c_{i,3}}), \quad \mathbf{s(V_j)} = \mathbf{p_j} = \frac{1}{3}(\mathbf{c_{j,1}} + \mathbf{c_{j,2}} + \mathbf{c_{j,3}}), \tag{2.4}$$

$$\mathbf{u_i} = \frac{1}{2}(\mathbf{c_{i,2}} + \mathbf{c_{i,3}}), \quad \mathbf{u_j} = \frac{2}{3}\mathbf{c_{j,1}} + \frac{1}{6}(\mathbf{c_{j,2}} + \mathbf{c_{j,3}}), \quad \mathbf{r_{i,j}} = \frac{1}{2}(\mathbf{u_i} + \mathbf{u_j}). \tag{2.5}$$

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 surace 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 $\vec{\gamma}$ 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

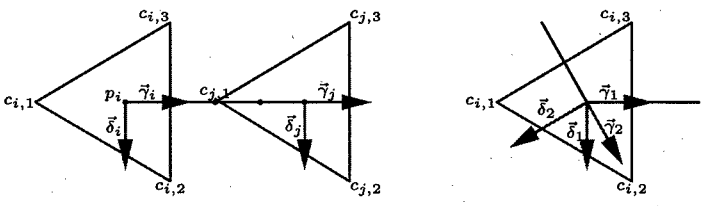

FIG. 3. Tangent and cross-boundary tangent vectors.

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 $(\mathbf{p_i}, \vec{\gamma}_\mathbf{i}, \vec{\delta}_\mathbf{i})$.

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 $\mathbf{p_i} + \epsilon\vec{\gamma}_\mathbf{i} + \nu\vec{\delta}_\mathbf{i}$, $\epsilon, \nu \in \mathbf{R}$, such that $\mathbf{s(V_i)} = \mathbf{p_i}$. Curve point interpolation is simply expressed by (2.4). Furthermore, we let the tangent to $\mathbf{e_i}$ at V_i be parallel to $\vec{\gamma}_i$:

$$\mathbf{u_i} - \mathbf{p_i} = \frac{\mathbf{1}}{\mathbf{6}}(\mathbf{c_{i,2}} + \mathbf{c_{i,3}}) - \frac{\mathbf{1}}{\mathbf{3}}\mathbf{c_{i,1}} = \alpha_\mathbf{i}\vec{\gamma}_\mathbf{i}, \tag{3.1}$$

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 $\mathbf{c_{i,2}} - \mathbf{c_{i,3}}$:

$$\mathbf{c_{i,2}} - \mathbf{c_{i,3}} = \mathbf{2}\beta_\mathbf{i}\vec{\delta}_\mathbf{i}, \tag{3.2}$$

where β_i is again a scaling factor.

Solving (2.4), (3.1) and (3.2) to $\mathbf{c_{i,j}}$ in terms of the unknown α_i and β_i (further called the α– and β–factors) yields

$$\begin{cases} \mathbf{c_{i,1}} &= \mathbf{p_i} - \alpha_\mathbf{i}\vec{\gamma}_\mathbf{i} \\ \mathbf{c_{i,2}} &= \mathbf{p_i} + \frac{\alpha_\mathbf{i}}{\mathbf{2}}\vec{\gamma}_\mathbf{i} + \beta_\mathbf{i}\vec{\delta}_\mathbf{i} \\ \mathbf{c_{i,3}} &= \mathbf{p_i} + \frac{\alpha_\mathbf{i}}{\mathbf{2}}\vec{\gamma}_\mathbf{i} - \beta_\mathbf{i}\vec{\delta}_\mathbf{i}. \end{cases} \tag{3.3}$$

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 (2.3) 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 $\mathbf{r_{i,j}} = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{u_i} + \mathbf{u_j}) = \mathbf{p_{i,j}}$, where $\mathbf{p_{i,j}}$ is the given curve point. Taking (2.5) and (3.3) into account, we have

$$\alpha_i\vec{\gamma}_i - \alpha_j\vec{\gamma}_j = 4\mathbf{p_{i,j}} - \mathbf{2}(\mathbf{p_i} + \mathbf{p_j}) = \mathbf{q_{i,j}}. \tag{3.4}$$

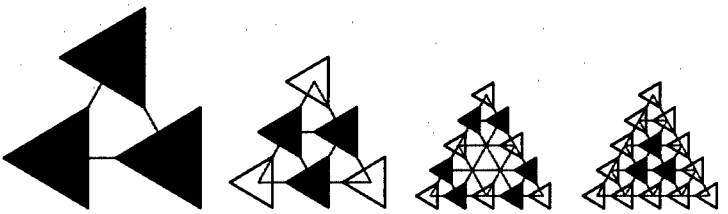

FIG. 4. Consecutive iteration steps.

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 (2.2) and (3.2):

$$\beta'_{i,j}\vec{\delta}'_{i,j} = \frac{1}{4}(\beta_i\vec{\delta}_i + \beta_j\vec{\delta}_j), \tag{3.5}$$

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

$$\beta_{i,j}\vec{\delta}_{i,j} = \frac{1}{2}(\beta_i\vec{\delta}_i + \beta_j\vec{\delta}_j). \tag{3.6}$$

Note that $\vec{\delta}_{i,j}$ has been used instead of $\vec{\delta}'_{i,j}$. This is again an overdetermined 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 4.4).

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 until a certain stopping criterion is satisfied. Finally, during a post–iteration step, the interior 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.

4.1 An initial solution

The initial solution (Figure 4, leftmost) is easily obtained by solving (3.4) in the least squares sense for each edge V_iV_j. If we assume that $\vec{\gamma}_i \neq \vec{\gamma}_j$, then

$$\alpha_i = \frac{1}{D}\left((\vec{\gamma}_i \cdot \mathbf{q_{i,j}}) - (\vec{\gamma}_\mathbf{j} \cdot \mathbf{q_{i,j}})(\vec{\gamma}_\mathbf{i} \cdot \vec{\gamma}_\mathbf{j})\right), \tag{4.1}$$

$$\alpha_j \;=\; \frac{1}{D}\left(-(\vec{\gamma}_j \cdot \mathbf{q_{i,j}}) + (\vec{\gamma}_i \cdot \mathbf{q_{i,j}})(\vec{\gamma}_i \cdot \vec{\gamma}_j)\right), \tag{4.2}$$

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, T_i is completely determined. The β–factors can be calculated by writing (3.3) for both edges incident with the vertex and eliminating $\mathbf{c_2}$, respectively $\mathbf{c_1}$, e.g., for Figure 3, right,

$$\beta_1 = \alpha_2(\vec{\gamma}_2 \cdot \vec{\delta}_1), \quad \beta_2 = -\alpha_1(\vec{\gamma}_1 \cdot \vec{\delta}_2). \tag{4.3}$$

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 $\vec{\gamma}_i = \vec{\gamma}_j$, (3.4) 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 $\alpha_i = \alpha_j = \frac{1}{2}|V_iV_j|$.

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

$$\begin{cases} \alpha_i\vec{\gamma}_i - \alpha_k\vec{\gamma}_k &= \mathbf{q_{i,k}}, \\ \alpha_k\vec{\gamma}_k - \alpha_j\vec{\gamma}_j &= \mathbf{q_{k,j}}, \end{cases} \tag{4.4}$$

where α_i and α_j are known. Thus, α_k can be obtained as the least-squares solution of (4.4):

$$\alpha_k = \frac{1}{2}(\vec{\gamma}_k \cdot (\alpha_i\vec{\gamma}_i - \mathbf{q_{i,k}} + \mathbf{q_{k,j}} - \alpha_j\vec{\gamma}_j)). \tag{4.5}$$

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:

$$\begin{cases} \beta_{i,k}\vec{\delta}_{i,k} &= \frac{1}{2}(\beta_i\vec{\delta}_i + \beta_k\vec{\delta}_k), \\ \beta_{k,j}\vec{\delta}_{k,j} &= \frac{1}{2}(\beta_k\vec{\delta}_k + \beta_j\vec{\delta}_j), \end{cases} \tag{4.6}$$

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 (3.6), i.e., by subdivision and upscaling.

4.3 The interior control points

Finally, as soon as the user supplied edge curves have been approximated well enough, the interior control points at the eventual refinement level have to be calculated. We will

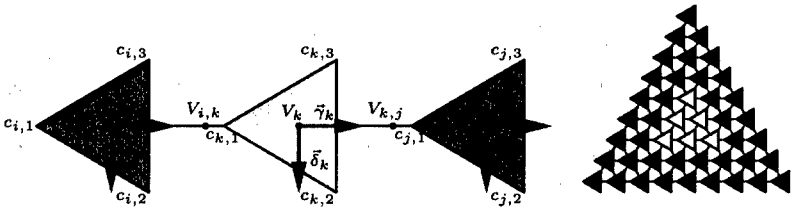

FIG. 5. The refinement and post-iteration steps.

FIG. 6. The hole and the triangular patches.

discuss three possibilities by the help of an example; Figure 6 shows a hole (left) and two filling patches (right).

Copy From Initial. The interior control points are obtained directly from the initial solution by subdivision. This guarantees that the interior 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, middle.

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 7, right.

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

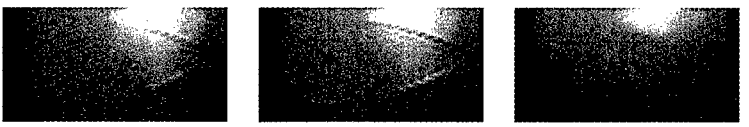

FIG. 7. Copy from initial solution and averaging (4 iterations); instant update (3 iterations).

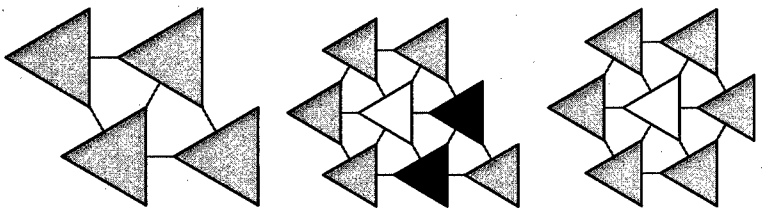

FIG. 8. Cases with 4, 5 and 6 boundary curves.

4.4 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 8 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 (4.4) to the unknown α_i, α_j and α_k. The β–factors of the outer control poygons are obtained as usual; for the middle polygon one can apply (3.6). Also, for the cases of 5 and 6 boundary curves, an interior control triangle (unshaded) has to be calculated for the initial solution. This can be done by averaging the six surrounding control polygons.

Bibliography

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. Windmolders, J., Dierckx, P. (1999), Subdivision of Uniform Powell–Sabin splines, Computer Aided Geometric Design 16, 301-315.
6. Windmolders, J. and P. Dierckx, NURPS for Special Effects and Quadrics: Oslo 2000, Tom Lyche and L. L. Schumaker (eds.), Vanderbilt Press, Nashville 2001.