Pseudo-Spline Subdivision Surfaces

Chongyang Deng¹ and Kai Hormann²

¹School of Science, Hangzhou Dianzi University, Hangzhou 310018, China
²Faculty of Informatics, Università della Svizzera italiana, 6904 Lugano, Switzerland

Abstract

Pseudo-splines provide a rich family of subdivision schemes with a wide range of choices that meet various demands for balancing the approximation power, the length of the support, and the regularity of the limit functions. Special cases of pseudo-splines include uniform odd-degree B-splines and the interpolatory 2n-point subdivision schemes, and the other pseudo-splines fill the gap between these two families. In this paper we show how the refinement step of a pseudo-spline subdivision scheme can be implemented efficiently using repeated local operations, which require only the data in the direct neighbourhood of each vertex, and how to generalize this concept to quadrilateral meshes with arbitrary topology. The resulting pseudo-spline surfaces can be arbitrarily smooth in regular mesh regions and $C^1$ at extraordinary vertices.

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

1. Introduction

Subdivision curves and surfaces are widely used in computer graphics, geometric modelling, and computer animation, because they allow the user to model smooth curves and surfaces by manipulating a small set of control vertices, similar to NURBS modelling. However, unlike NURBS, there is no constraint on the connectivity of the control vertices in the surface setting, which makes subdivision more appropriate for modelling complex surfaces.

Subdivision schemes with higher order continuity usually produce visually smoother surfaces than those with lower order, but they have a larger support and their implementation has to face two difficulties: on the one hand, the computation can be inefficient if the relevant subdivision masks are not properly designed, and on the other hand, complex subdivision masks need to be handled for computing new vertices near extraordinary vertices. In some special cases, these two problems can be solved by using repeated local operations, which require only the data in the direct neighbourhood of a vertex.

The most prominent example is the Lane–Riesenfeld algorithm [LR80], which efficiently refines the control polygon of a uniform B-spline curve of arbitrary degree using only local averaging operations. This idea was later extended to surfaces for refining quadrilateral control meshes of arbitrary topology [Pra98, ZS01, WW01, Sta01]. In regular regions, all these schemes generate tensor-product B-spline...
In a nutshell, each subdivision step first refines the control mesh with a simple refinement subdivision step and then updates the vertices of the refined mesh with a number of local operations that depend only on the direct neighbourhood of each vertex. This strategy is more general than the one described by Deng and Ma [DM13] and it is also simpler, because after the initial refinement stage, all new vertices are treated in the same way, so that there is no need to distinguish between different vertex types. The resulting pseudo-spline surfaces are tensor products of univariate pseudo-splines around regular vertices, thus inheriting their order of continuity, and around extraordinary vertices the limit surface turns out to be $C^1$ (see Section 4). The pseudo-spline parameters then provide the user with intuitive control for selecting a limit surface that is visually smooth and still close to the initial control mesh (see Section 5).

The main contributions of this paper are:

- decomposing the pseudo-spline refinement rule into repeated local operations;
- extending this idea to quadrilateral meshes;
- developing special local rules for extraordinary vertices which guarantee the limit surfaces to be $C^1$-continuous.

## 2. Pseudo-spline curves

Starting from some initial control polygon $P^0$, $i \in \mathbb{Z}$, a subdivision scheme generates the refined polygons $P^k$, $k \in \mathbb{N}$ recursively according to the refinement equation

$$P^k_j = \sum_{j \in \mathbb{Z}} a_{i-2j} P^k_j, \quad i \in \mathbb{Z},$$

where $a = (a_i)_{i \in \mathbb{Z}}$ is the subdivision mask. The $z$-transform of the mask $a$,

$$a(z) = \sum_{i \in \mathbb{Z}} a_i z^i,$$

is usually called the symbol of the scheme.

The pseudo-spline schemes are defined by the symbols [DS07, DHSS08]

$$d_n(z) = 2\sigma(z)^n \sum_{j=0}^{n} \binom{n+j-1}{j} \delta(z)^j,$$

where

$$\sigma(z) = \frac{(1+z)^2}{4z} \quad \text{and} \quad \delta(z) = -\frac{(1-z)^2}{4z}.$$  

Note that $d_n(z) = 2\sigma(z)^n$ is the symbol of the $(2n-1)$-degree uniform B-spline scheme with $C^{2n-2}$-continuous limit curves [LR80] and that $d_n^{-1}(z)$ is the symbol of the interpolatory 2$n$-point subdivision scheme [DD89] with $C^{2n}$-continuous limit curves, where $r_n \approx 0.415n$ [Dau92, Eir92]. For fixed $n$, the regularity of the limit curves decreases as $l$ increases, and for fixed $l$ the regularity increases with $n$ [DS07, FM13].
Since multiplying the symbol with the smoothing factor $\sigma(z)$ corresponds to updating each vertex with the weighted average of itself and its two neighbours \[WW01\], the crucial point of implementing the subdivision step of the pseudo-spline with symbol $a$ is the decomposition of

$$b_1(z) = 2\sigma(z) \sum_{j=0}^{l} \binom{n+j-1}{j} \delta(z)^j$$

into a sequence of local operations.

**Theorem 1** The symbol $b_m(z)$ satisfies the recursive formula

$$b_{m+1}(z) = b_m(z) + \gamma_m \delta(z) d_m(z)$$

with $d_m(z) = b_m(z) - b_{m-1}(z)$ and $\gamma_m = (n + m)/(m + 1)$.

**Proof** By the definition of $b_m(z)$ in (2),

$$d_m(z) = b_m(z) - b_{m-1}(z) = 2\sigma(z) \binom{n+m-1}{m} \delta(z)^m,$$

and (3) then follows because

$$b_{m+1}(z) - b_m(z) = 2\sigma(z) \binom{n+m}{m} \delta(z)^{m+1} = \gamma_m \delta(z) d_m(z).$$

Following Theorem 1 and letting

$$c_m(z) = \sigma(z)^{m-1} b_1(z),$$

the symbol $a_m(z)$ is given by the sequence

$$2\sigma = b_1(z) \rightarrow b_1(z) \rightarrow \cdots \rightarrow b_1(z)$$

$$a_m(z) = c_0(z) \leftarrow \cdots \leftarrow c_1(z)$$

where the arrows in the top row correspond to the recursive formula (3) and the arrows in the bottom row correspond to a multiplication with $\sigma(z)$.

This sequence leads to the following computations for subdividing the polygon $P^l$. We first refine $P^l$ by inserting all edge midpoints, giving the intermediate polygon $Q^0$ with vertices

$$Q_{2j}^0 = P_j^l, \quad Q_{2j+1}^0 = (P_j^l + P_{j+1}^l)/2, \quad j \in \mathbb{Z}.$$ 

This refinement corresponds to the initial symbol $b_0(z) = 2\sigma(z)$ in (5). We now proceed to update the vertices of $Q^0$ by iteratively applying the $l$ local operations as derived from (3),

$$Q_j^m = Q_j^m - Q_j^{m-1},$$

$$Q_j^{m+1} = Q_j^m + \gamma_m (-D_j^{m-1} + 2D_j^m - D_j^{m+1})/4,$$

for $j \in \mathbb{Z}$ and $m = 0, \ldots, l - 1$, with $Q_j^{-1} = 0$. After these $l$ steps we obtain the intermediate polygon $Q^l$ which corresponds to a subdivision step with the symbol $b_1(z) = c_1(z)$.

It remains to set $R_j^1 = Q_j^l$ and to carry out the $n - 1$ local operations which relate to the bottom row of (5) by iteratively computing

$$R_j^{n+1} = (R_j^{n} + 2R_j^{n+1})/4$$

for $j \in \mathbb{Z}$ and $m = 0, \ldots, n - 1$, to finally get the vertices $P_{j+1}^{l+1} = R_j^l$ of the refined polygon $P^{l+1}$.

Figure 2 illustrates this procedure for the interpolatory 6-point subdivision scheme, which is the pseudo-spline scheme with symbol $a_2(z)$. Figure 3 further shows some examples of limit curves for the pseudo-spline schemes. Remember that the choice $l = 0$ gives uniform B-splines of degree $2n - 1$ (green curves) and that $l = n - 1$ gives the limit curves of the interpolatory 2n-point scheme (red curves), and note how varying $l$ blends between these two limit curves. It is even possible to consider non-integer values of $l$ by adapting the ideas in \[RS08\] for continuously blending between the subdivision rules of integer-valued pseudo-spline schemes, but investigating this approach, which would also work in the surface setting, is beyond the scope of this paper.

Since each local operation enlarges the support by 1 both to the left and to the right, it is not hard to see that the support of the basic limit function of the pseudo-spline with symbol...
\[ a_n^i(z) = [-n - l, n + l], \] but Figure 4 shows that the practical support is considerably smaller.

3. Pseudo-spline surfaces

The same idea can also be applied to efficiently compute pseudo-spline surfaces from quadrilateral control meshes. Let us first consider the tensor product setting, where all vertices \( P_{i,j}^l, i, j \in \mathbb{Z} \) of the initial control mesh \( M^0 \) are regular with valency 4 and the vertices of the refined meshes \( M^k, \) \( k \in \mathbb{N} \) are generated recursively by the refinement equation

\[
P_{i,j}^{l+1} = \sum_{m,n \in \mathbb{Z}} a_{i-2m,j-2n} P_{i,j}^l, \quad i, j \in \mathbb{Z},
\]

where \( a = (a_{i,j})_{i,j} \subset \mathbb{Z}^2 \) is the bivariate subdivision mask and its z-transform

\[ a(x,y) = \sum_{i,j \in \mathbb{Z}} a_{i,j} x^i y^j \]

is the bivariate symbol of the scheme. Defining the bivariate symbols

\[
\sigma(x,y) = \sigma(x) \sigma(y), \\
\delta(x,y) = \delta(x) \delta(y), \\
\chi(x,y) = \delta(x) + \delta(y)
\]

and

\[
b_m(x,y) = b_m(x) b_m(y), \\
d_m(x,y) = d_m(x) d_m(y), \\
e_m(x,y) = b_m(x) - b_{m-1}(x,y),
\]

the symbol of the tensor product pseudo-spline scheme can be written as

\[ a_n^i(x,y) = a_n^i(x) a_n^i(y) = \sigma(x)^n \delta(y)^i b_i(x), \]

and, like in the case, the crucial step is to decompose the symbol \( b(x,y) \) into a sequence of local operations.

**Theorem 2** The symbol \( b_m(x,y) \) satisfies the recursive formula

\[
b_{m+1}(x,y) = b_m(x,y) + d_{m+1}(x,y) \\
- \gamma_m \delta(x) d_m(x,y) \\
+ \gamma_m \chi(x,y) e_m(x,y) \\
- \gamma_m \gamma_{m-1} \delta(x,y) e_{m-1}(x,y),
\]

\[ \text{Equation (6)} \]

**Proof** By Theorem 1 and noting that (4) implies

\[ d_{m+1}(x,y) = \gamma_m^2 \delta(x,y) d_m(x,y), \]

we have

\[ b_{m+1}(x,y) = (b_m(x) + \gamma_m \delta(x) b_m(x) - b_{m-1}(x,y)) \\
\times (b_m(y) + \gamma_m \delta(y) b_m(y) - b_{m-1}(y)) \\
= b_m(x,y) + d_{m+1}(x,y) + \gamma_m \chi(x,y) b_m(x,y) \\
- \gamma_m [b_m(x) \delta(x) b_{m-1}(y) + b_m(y) \delta(x) b_{m-1}(x)].
\]

Equation (6) then follows because

\[
b_m(x) \delta(x) b_{m-1}(y) + b_m(y) \delta(y) b_{m-1}(x) = \\
[b_{m-1}(x) + \gamma_m \delta(x) d_{m-1}(x) \delta(y) b_{m-1}(y) \\
+ [b_{m-1}(y) + \gamma_m \delta(y) d_{m-1}(y) \delta(x) b_{m-1}(x)] \\
= \chi(x,y) b_{m-1}(x,y) \\
+ \gamma_m \delta(x,y) [d_{m-1}(x) b_{m-1}(y) + d_{m-1}(y) b_{m-1}(x)]
\]

and, using again (7),

\[
d_{m-1}(x) b_{m-1}(y) + d_{m-1}(y) b_{m-1}(x) = \\
b_{m-1}(x,y) + d_{m-1}(x,y) - b_{m-2}(x,y) \\
e_{m-1}(x,y) + d_m(x,y) / \gamma_m \delta(x,y) \\
\]

\[ \square \]

Therefore, letting

\[ c_m(x,y) = c_m(x) c_m(y) = \sigma(x)^m \delta(y)^n b_n(x), \]

the symbol \( a_n^i(x,y) \) is given by the sequence

\[ 4 \sigma(x,y) = b_0(x,y) \rightarrow b_1(x,y) \rightarrow \cdots \rightarrow b_1(x,y). \]

\[ (8) \]

\[ a_n^i(x,y) = c_n(x,y) \cdots c_2(x,y) \leftarrow c_1(x,y) \]

similar to how it is done in the univariate setting in (5).

In the tensor product setting,

\[ \sigma(x,y) = \frac{1}{4} \left( x + \frac{1}{2} + y + \frac{1}{2} \right) + \frac{1}{4} \left( xy + \frac{1}{2} + \frac{1}{2} \right), \]

\[ \delta(x,y) = \frac{1}{4} \left( x + \frac{1}{2} + y + \frac{1}{2} \right) + \frac{1}{4} \left( xy + \frac{1}{2} + \frac{1}{2} \right), \]

\[ \chi(x,y) = \frac{1}{4} \left( x + \frac{1}{2} + y + \frac{1}{2} \right), \]

which corresponds to the masks shown in Figure 6.

For an extraordinary vertex with valency \( N \), we adopt the weights used in the repeated averaging algorithm for the surface generalization of odd-degree B-spline subdivision [ZS01] and generalize the mask \( \sigma(x,y) \) as shown in Figure 7. Following the pattern of the regular masks, we set the weights of the generalized mask \( \delta(x,y) \) to the same values,
### C. Deng & K. Hormann / Pseudo-Spline Subdivision Surfaces

![Figure 5](image-url)

**Figure 5:** Splitting the subdivision step of the pseudo-spline with symbol $a_3^2(x,y)$ to refine $P^k$ to $P^{k+1}$ into 5 repeated local operations. In each step, the mesh before the operation is shown in red and the mesh after the operation in black.

![Figure 6](image-url)

**Figure 6:** Masks for local operations at regular vertices.

![Figure 7](image-url)

**Figure 7:** Masks for local operations at extraordinary vertices.

Figure 8: Local indices of a vertex and its neighbours.

except that we negate the weights of the direct neighbours, and an intuitive generalization of the mask for $\chi(x,y)$ is the one in Figure 7. Note that these generalized masks are identical to the masks in Figure 6 for $N = 4$ and satisfy the identity

$$\delta(x,y) = \sigma(x,y) + \chi(x,y) - 1,$$

just as in the regular setting.

According to Theorem 2 and the sequence diagram in (8), we perform the following computations for refining the mesh $M^k$ with vertices $P^k$. Analogously to the curve setting, we first refine $M^k$ by inserting all edge midpoints and the barycentres of all faces, giving the intermediate mesh $N^0$ with vertices

$$Q^0_f = P^k_f,$$

copied from $M^k$, new edge vertices

$$Q^0_e = (P^k_{e1} + P^k_{e2})/2,$$

one for each edge $e = [P^k_{e1}, P^k_{e2}]$ in $M^k$, and new face vertices

$$Q^0_f = (P^k_{f1} + P^k_{f2} + P^k_{f3} + P^k_{f4})/4,$$

one for each quadrilateral $f = \{P^k_{f1}, P^k_{f2}, P^k_{f3}, P^k_{f4}\}$ in $M^k$. The vertices of $N^0$ are connected by splitting the quadrilaterals in $M^k$ with the classical 1-to-4 rule, so that $N^0$ is again a quadrilateral mesh. This refinement corresponds to the initial symbol $b_0(x,y) = \mathcal{A}_4\sigma(x,y)$ in (8), and we now proceed to update the vertices of $N^0$ iteratively applying the $l$ local operations as derived from (6) to all vertices.

Without loss of generality let $Q^0_0$ be a vertex in $N^0$ with valency $N$ and neighbours $Q^0_1, \ldots, Q^0_{2N}$ as shown in Figure 8. Equations (7) and (6) then translate to computing

$$D^0_{m+1} = \frac{\gamma_m}{N} \sum_{i=0}^{2N} (D^0_{m} - 2D^0_{m+1} + D^0_{m+2})/4,$$

$$E^m_i = Q^m_i - Q^m_{i-1}, \quad i = 0, \ldots, 2N,$$

$$Q^0_{m+1} = Q^0_0 + D^0_{m+1} + (\gamma_m/\gamma_{m-1})D^0_{m}$$

$$+ \frac{\gamma_m}{N} \sum_{i=1}^{2N} (E^m_{i-1} - E^m_{2i-1})$$

$$- \frac{\gamma_m}{N} \sum_{i=0}^{2N} (E^m_{i-1} - 2E^m_{2i-1} + E^m_{4i-1})/4,$$

for $m = 0, \ldots, l - 1$, with initial values $D^0_0 = Q^0_0$, $Q^{-1}_0 = 0$ and the convention that $\gamma_0/\gamma_{-1} = 0$ and $\gamma_{-1} = 0$. After these $l$ steps we obtain the intermediate mesh $N^l$ which corresponds to a subdivision step with the symbol $b_l(x,y) = e_l(x,y)$.

It remains to set $R^1_0 = Q^1_0$ and to carry out the $n - 1$ local operations which relate to the bottom row of (8) by iteratively computing

$$R^0_{m+1} = \frac{1}{N} \sum_{i=1}^{2N} (R^0_m + 2R^0_{m-1} + R^0_{4m-1})/4.$$
Figure 9: Pseudo-spline surfaces with fixed \( n \) and varying \( l \).

Figure 10: Examples of basic limit functions for different valencies \( N \) and pseudo-spline surfaces with \((n,l) = (5,0)\) (top), \((n,l) = (4,1)\) (middle), and \((n,l) = (3,2)\) (bottom). All functions have the same support size, namely the 5-ring neighbourhood.
for $m = 1, \ldots, n - 1$, to finally get the vertices $I^{k+1}_j = Q^n_j$ of the refined mesh $M^{k+1}$.

Figure 5 illustrates this procedure for the pseudo-spline subdivision scheme with symbol $a(x, y)$, which generalizes the 6-point interpolatory subdivision scheme. Figures 1 and 9 further show some examples of limit surfaces for the pseudo-spline surface schemes.

3.1. Properties

Since each local operation enlarges the support exactly by one 1-ring neighbourhood, it follows that the support of the basic limit function of the pseudo-spline surface scheme with parameters $n$ and $l$ extends to the $(n + l)$-ring neighbourhood. Some examples are shown in Figure 10.

Pseudo-spline surfaces generalize several known subdivision schemes. For example, choosing $l = 0$ reproduces the approximating B-spline schemes by Zorin and Schröder [ZS01] and the interpolatory scheme by Li and Ma [LM07] is reproduced for $n = 2$ and $l = 1$. More generally, the pseudo-spline schemes with $n \geq 2$ and $l = n - 1$, which are the surface analogues of the interpolatory 2n-point schemes, are identical to the alternative schemes by Deng and Ma [DM13] for regular vertices. Consequently, pseudo-spline surfaces can generate exact tori in the limit as $n \to \infty$ as shown in [DM13] and illustrated in Figure 11. Around irregular vertices, the subdivision rules are different and while the limit surfaces of [DM13] interpolate all initial vertices, pseudo-spline surfaces interpolate only regular initial vertices with regular neighbours.

The Catmull–Clark scheme [CC78] can also be reproduced by choosing $n = 2$ and $l = 0$ and by slightly modifying the generalized mask $\sigma(x, y)$ in Figure 7, that is, by replacing the weights $\frac{1}{N}$ of the direct neighbours with $\frac{1}{N}$ and the weights $\frac{1}{N}$ of the opposite neighbours with $\frac{1}{N}$. However, we found that our choice of $\sigma(x, y)$ gives better curvature behaviour of the limit surfaces for general $n$ and $l$.

We actually tried different ways of generalizing the masks in Figure 6 for extraordinary vertices, using the obvious symmetries of the weights, the summation of weights to one for $\sigma$ and $\chi$ and to zero for $\delta$, and identity (9) as constraints, but found that the choice of weights given in Figure 7 performed best.

4. Smoothness analysis

The smoothness of pseudo-spline surfaces at regular vertices is the same as that of the corresponding univariate pseudo-spline and increases for increasing $n$ and decreasing $l$. At extraordinary vertices, we verified numerically that the surfaces are $C^1$-continuous for $2 \leq n \leq 5$, $1 \leq l \leq n - 1$ and valency $3 \leq N \leq 50$.

More precisely, we follow [PR08] and first used Matlab to verify that the leading eigenvalues of the subdivision matrix satisfy

$$1 = \lambda_0 > \lambda_1 = \lambda_2 > |\lambda_3|.$$ 

Secondly, we checked that the characteristic maps are regular, using Theorem 5.25 of [PR08] and a numerical checking.
method similar to that of Augsdörfer et al. \cite{ACDS09}. Figure 12 shows some typical examples of these characteristic maps. Thirdly, we checked that the Fourier indices \cite{PR04} of the subdominant eigenvalue $\lambda_2$ are $\{1, N - 1\}$, which then guarantees the injectivity of the characteristic maps. Overall, this is sufficient to conclude $C^1$-continuity of the limit surface.

At extraordinary vertices, our schemes do not satisfy the necessary condition $(\lambda_1)^2 = \lambda_2$ for bounded curvature and curvature continuity and so the limit surfaces are not $C^3$. However, the smoothness improves for increasing $n$ and $l$, because $\lambda_2$ approaches $(\lambda_1)^2$ as shown in Figure 14, except at vertices with valency $N = 3$.

5. Numerical examples

We implemented and tested our algorithm for computing pseudo-spline surfaces for a variety of input meshes. The shaded versions in Figures 1 and 9, as well as the reflection lines and Gaussian curvature plots in Figure 13 show the behaviour of the limit surface for a fixed number of $n = 4$ smoothing steps and varying $l$. On the one hand, $l = 0$ gives a degree-7 B-spline surface, which is $C^6$ except at extraordinary vertices, but quite far from the control mesh. On the other hand, $l = 3$ gives a surface analogue of the interpolatory 8-point scheme, which is $C^3$ around regular vertices and interpolates most of the control vertices. It is comparable to the surface generated by the 8-point scheme of Deng and Ma \cite{DM13}, which has the same smoothness, but our surface tends to give sharper edges and a smoother shape around regular vertices.

We observe that the smoothness as well as the distance between the control vertices and the limit surface vary intuitively between $l = 0$ and $l = n - 1$ (see Figure 15), thus providing a useful control parameter to the user. The other
control parameter \( n \) can be used to generally increase the smoothness of the limit surfaces, as shown by Dong and Shen [DS07, Table 1] and Floater and Muntingh [FM13, Theorem 3], except at extraordinary vertices. For example, if the user desires interpolating limit surfaces, then increasing \( n \) while choosing \( l = n - 1 \) gives successively smoother results, as shown in Figure 16. As mentioned above, the resulting limit surfaces do not interpolate extraordinary control, and the distance increases with \( n \) (see Figure 17).

A convenient consequence of our decomposition approach to generating pseudo-spline surfaces is that we can handle arbitrary polygonal input meshes, because the first local operation of the first subdivision step is midpoint refinement, which splits any initial polygonal face into quadrilaterals by connecting its barycentre with the edge midpoints. All subsequent local operations and subdivision steps then operate on quadrilaterals without the need to change anything in the algorithm. An example is shown in Figure 18.

6. Conclusion

We have shown in this paper how to generate pseudo-spline surfaces by iteratively subdividing an initial quadrilateral mesh with arbitrary topology. These subdivision surfaces are interesting for modelling, because they come with intuitive control parameters for blending between interpolating and approximating surfaces with arbitrarily high order of continuity in regular mesh regions and \( C^1 \)-continuity at extraordinary vertices. This considerably enlarges the design space of possible limit surfaces and it is even possible to set these control parameters to different values in different parts of the initial mesh, as shown in Figure 19. It remains future work, however, to design appropriate subdivision rules for the transition regions to avoid the curvature artefacts visible in this example.

The advantage of our approach is that each subdivision step is based on the repeated application of the masks \( \sigma, \delta, \chi \), which require only the information from the direct neighbourhood of each vertex, even though the overall subdivision masks can have large support and would be cumbersome to generalize to extraordinary vertices.

We believe that the locality of the masks is also advantageous for implementing the subdivision process on the GPU.
Figure 19: Pseudo-spline surface with varying parameters $n$ and $l$ in different parts: $(n,l) = (3,0)$ in the light grey part, $(n,l) = (3,1)$ in the adjacent grey part, and $(n,l) = (3,2)$ in the dark grey part.

and will investigate this matter in future work. We further plan to extend the method to handle boundaries and features like corners and creases.

A potential limitation of our approach is that the surface analogues of the interpolatory $2n$-point schemes do not necessarily interpolate extraordinary input vertices, but it might be possible to fix this by adapting the weights $\gamma_k$ appropriately. It might further be possible to adapt the ideas of Augsdörfer et al. [ADS06] and to tune the weights $\gamma_k$ and the generalized local masks in Figure 7 such that the curvature of the limit surfaces becomes bounded.

Acknowledgements

This work is supported by NSFC (61370166, 61379072, 61003194), the Open Project Program of the State Key Lab (Grant No. A1304), Zhejiang University. We would like to thank Bay Raitt for the models used in Figure 1 and Figure 9 (top row), and Wenbiao Jin for helping with the visualization of the surfaces.

References


[Da92] DAUBECHIES I.: Ten Lectures on Wavelets. SIAM, 1992. 2


[LLLYL08] LIN S., LUO X., YOU F., LI Z.: Deducing interpolating subdivision schemes from approximating subdivision schemes. ACM Trans. Graph. 27, 5 (2008), Article 146, 7 pages. 2


© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.