9

Conclusion

The analysis of stationary linear subdivision algorithms presented in this book, summarizes and enhances the results of three decades of intense research and combines them into a full framework. While our understanding of C^1-algorithms is now almost complete, the generation and the analysis of algorithms of higher regularity still offers some challenges. Guided subdivision and the PTER-framework pave a path towards algorithms of higher regularity, that, for a long time, were considered to be not constructible. Various aspects of these new ideas have to be investigated, and the development is in full swing, at the time of writing.

In focusing on analytical aspects of subdivision surfaces from a differential geometric point of view, we left out a number of other interesting and important topics, such as the following.

- *Implementation issues:* In applications, subdivision is mostly considered a recipe for mesh refinement, rather than a recursion for generating sequences of rings. The availability of simple, efficient strategies for implementation [ZS00, SAUK04], even in the confines of the Graphics Processing Unit [BS02a, SP03, SJP05a], evaluation of refinable functions at arbitrary rational parameters [CDM91] and, for polynomial splines, at arbitrary parameters [Sta98a, Sta98c] and inclusion into the graphics pipeline [ea98, DKT98] largely account for the overwhelming success of subdivision in Computer Graphics.
- *Sharp(er) features:* By using modified weights for specially 'tagged' vertices or edges, it is possible to blend subdivision of space curves and subdivision surfaces to deliberately sharpen features and even reduce the smoothness of subdivision surfaces to represent creases or cusps [DKT98, Sch96]. In a similar way, subdivision algorithms can be adapted to match curves and boundaries [Nas91, Lev99c, Lev00, Nas03].
- *Multiresolution:* Based on the inherent hierarchy of finer and finer spaces, one can develop strategies for multiresolution editing of of subdivision surfaces [LDW97]. There are close relations to the study of wavelets, but this development is still in its infancy.

- *Applications in Scientific Computing:* Beyond the world of Computer Graphics, subdivision surfaces can be employed for the simulation of thin shells and plates [COS00, GKS02, Gri03, GTS02]. Also use in the boundary element method is conceivable.

The book of Warren and Weimer [WW02] offers an excellent compilation of these and other application-oriented issues.

Necessarily, the material presented here is a compromise between generality and specificity. Therefore, to conclude, we want to review the basic assumptions of our analysis framework, check applicability to the rich 'zoo' of subdivision algorithms in current use, and discuss possible generalizations. We consider in turn function spaces, types of recursion, and the underlying combinatorial structure.

9.1 Function spaces

The spaces of splines and rings that we considered throughout the book cover to a very wide range of algorithms. Many popular algorithms are generalizations of subdivision schemes for B-splines or box-splines [Vel01b, Vel01a, VZ01, ZS01, CC78, DS78, PR97]. According to the classical definition of the term 'spline', the rings are piecewise polynomial then. However, except were explicitly mentioned, e.g. in the degree estimate in Theorem 7.19/150, this concepts and proofs of this book never relied on the particular type of functions used to build the generating rings g_ℓ. In fact, Definitions 3.1/49 and 4.9/71 are very general. The only limiting factors on the function spaces are:

- The segments are C^1 and join *parametrically* smooth.
- All rings $\mathbf{x}^m$ can be represented by using *one and the same finite-dimensional* system G of generating rings.
- The generating rings form a partition of unity.

Therefore, the framework covers for instance all schemes where the g_ℓ are derived from tensoring univariate subdivision, such as the interpolatory algorithms discussed in [DGL87, DL99, HIDS02, Leb94, Kob96a]. Moreover, not even the tensor product structure is crucial. Using genuine bivariate subdivision schemes operating on a regular square grid, such as Simplest subdivision or non-polynomial variants thereof, is equally covered by the framework. By contrast, subdivision algorithms generating patches that join *geometrically smooth* are not considered by our analysis. In the terminology of this book, geometric continuity can be defined using non-rigid embeddings. The first algorithm of that type was suggest by Prautzsch for subdividing Freeform splines [Pra97]. Here, the limit consists of *finitely many* polynomial patches so that no limit analysis is required. Other algorithms exploiting geometric continuity can be found in [KP07b] and in [ZLLT06].

The last bullet above, requiring that generating rings form a partition of unity, and, similarly, that the rows of the subdivision matrix to sum to 1, are not strictly necessary but a relict of the fact that most algorithms were formulated by forming

local affine combinations of 'control points'. Only the following is actually needed. The ring with constant value 1 must be contained in the space spanned by G and it is mapped to itself by the recursion: for some vector e

$$Ge = 1, \quad Ae = e.$$

The changes required to adapt our analysis to this more general setting are marginal.

We emphasize that *not* stipulating *linear independence* of the generating system G is a crucial for generality. Example 4.14$_{76}$ gives a first indication that the class of overcomplete generating systems provides a much richer source of algorithms. In fact, by not assuming linear independence of G, our analysis applies *unchanged* to most kinds of *vector-valued or matrix-valued, Hermite or jet subdivision* algorithms, see for instance [XYD06, KMP06, CJ06]. The mapping to the framework of the book is as follows. All components of the vectors, matrices, or jets in question are collected in the single vector $\mathbf{Q}$, and all subdivision rules are encoded in a single subdivision matrix A. Since some of the coefficients, representing for instance derivative information, are not needed for the parametrization of the ring, (although they are involved in determining the coefficients of subsequent rings), the vector G of generating rings is simply padded with zeros at the corresponding places. The iteration $\mathbf{x}^m = GA^m\mathbf{Q}$ is then perfectly able to model the vector-valued or matrix-valued, Hermite or jet subdivision. All that remains to be done is to remove ineffective eigenvectors from A, if there are any, by the procedure of Theorem 4.20$_{82}$.

9.2 Recursion

Throughout, we assume that the recursion $\mathbf{Q}^m \to \mathbf{Q}^{m+1}$ is

- stationary and
- linear.

That is, there exists a square matrix A such that we can write

$$\mathbf{Q}^m = A\mathbf{Q}, \quad m \in \mathbb{N}_0.$$

Due to its simplicity, this class of algorithms is predominant in applications. *Non-stationary* linear algorithms may use a different rule $A(m)$ for each step,

$$\mathbf{Q}^{m+1} = A(m)\mathbf{Q}^m.$$

In the univariate case, such schemes arise in a natural way when subdividing L-splines. In particular, subdivision for exponential and trigonometric splines falls into that class. An important feature of special trigonometric splines is their ability to reproduce conic section, and in particular circles [MWW01, Sab04, CJar, ANM06]. Such schemes are *not* covered by our approach. Generally, the analysis of non-stationary algorithms is complicated, compared to our setup, by the need to characterize the joint spectrum of a sequence of subdivision matrices, i.e., the spectrum of arbitrary products of members of these families [Rio92, DL02].

Even less is known about *non-linear* algorithms. Here, we have

$$\mathbf{Q}^{m+1} = A(m, \mathbf{Q}^m)$$

for some non-linear function A. For instance, positions of new control points could be defined by geometric algorithms like the intersection of planes associated with the given points. Such an algorithm, which provably generates smooth surfaces, was suggested by Dyn, Levin, and Liu [DLL92]. Defining $\mathbf{Q}^{m+1}$ as the coefficients of the solution of a non-linear fairing problem for the corresponding ring yields another rich source of non-linear subdivision schemes. However, the analysis is far from being well understood. Further, non-linear schemes are encountered when generalizing subdivision to operate on manifolds or Lie groups. In the univariate case, there are some results in that direction due to Wallner and Dyn [WD05], while the regular bivariate case is currently studied by Grohs [Gro07].

Concerning the recursion, the framework is implicitly based on a further assumption. The algorithm are assumed to be *local* in the sense that only control points in the vicinity of an extraordinary point influence the recursion. This need not hold. There are algorithms, known as *variational subdivision*, where new control points are computed as the solution of a global, linear or non-linear fairing problem [HKD93a, Kob95b, Kob96b]. The visual results of these algorithms are impressive, but more or less nothing is known about the underlying theoretical properties.

9.3 Combinatorial structure

In discussing the topological and combinatorial structure of the space of the parameters, we have to, first of all, distinguish subdivision in the univariate, the bivariate, and the general multivariate setting. As summarized in Section 1.7/14, there exists an extensive literature covering the univariate case. Refinable shift-invariant constructions can be obtained in the multivariate setting by tensoring univariate subdivision, or by subdividing box splines [dHR93]. The non-shift invariant setting, in three variables, has been studied, for instance in [BMDS02, SHW04]. In this book, we focus on the bivariate case.

For local algorithms, it is admissible to reduce the analysis to a vicinity of an isolated singularity. In Chapter 4/63 we chose

- the local domain $\mathbf{S}_n := [0,1]^2 \times \mathbb{Z}_n$ to consist of n copies of the unit *square*, and
- a *binary refinement* of cells. That is, each square is subdivided into four new ones.

The algorithms of Catmull-Clark and Doo-Sabin, but also Simplest subdivision or 4–8-subdivision fit that pattern. However, many other popular algorithms, like Loop's subdivision [Loo87], the Butterfly algorithm [DLG90, ZSS96] or the three-direction box splines of Sabin's thesis [Sab77] are based on triangular patches. Here, the appropriate structure is obtained when replacing the unit square $\boldsymbol{\Sigma} = [0,1]^2$ by an equilateral triangle. Conceptually, the framework for such *triangular subdivision*

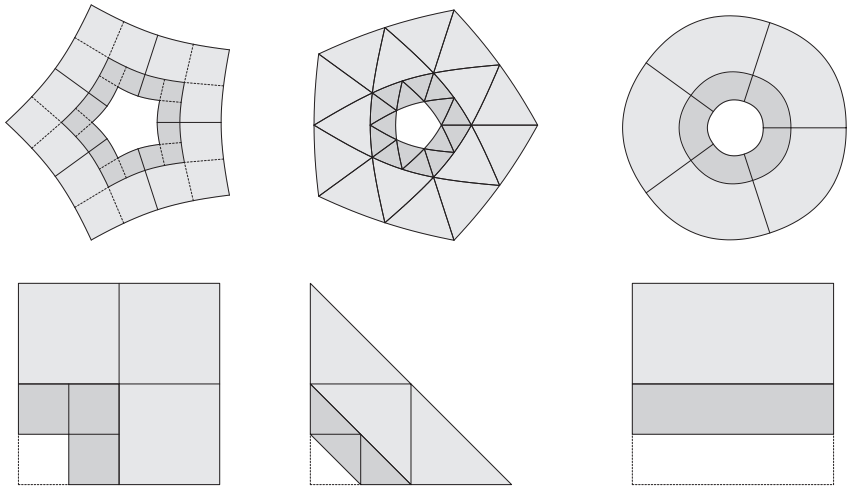

Fig. 9.1: Combinatorial Layout. (*left*) □-sprocket (Catmull-Clark subdivision) layout; (*middle*) △-sprocket (Loop subdivision) layout; (*right*) polar layout (9.1/186). Halving of the domain only the vertical direction. (*top*) Two nested characteristic rings. (*bottom*) $\mathbf{S}$ and $\mathbf{S}/2$ of the prolongation $\boldsymbol{\varphi}(\mathbf{S}/2) := \lambda\boldsymbol{\varphi}(\mathbf{S})$

algorithms is exactly the same. However, certain technical details concerning the neighbor relation and the contact conditions have to be adapted. The regular case for triangular subdivision is $n = 6$, corresponding to a lattice spanned by three directions. As in the quadrilateral case, the regular part $\boldsymbol{\Sigma}^0$ of the cells is obtained by subtracting one half of $\boldsymbol{\Sigma}$ from itself,

$$\boldsymbol{\Sigma}^0 := \boldsymbol{\Sigma}\backslash(\boldsymbol{\Sigma}/2).$$

As shown in Figure 9.1/185, $\boldsymbol{\Sigma}^0$ is a trapezoid rather than L-shaped.

Another generalization is to modify the number of new cells into which a given cell is subdivided. For binary refinement, each quadrilateral of triangular cell is split into $2 \times 2 = 4$ new ones. It is equally possible to use a $(\nu \times \nu)$-partition. In such a *ν-ary refinement*, we set

$$\boldsymbol{\Sigma}^0 := \boldsymbol{\Sigma}\backslash(\boldsymbol{\Sigma}/\nu).$$

For example, Kobbelt's $\sqrt{3}$-subdivision [Kob00] is based on *ternary refinement*, i.e., $\nu = 3$, and several other algorithms use the same or even finer tesselations [DIS03, OS03, Loo02b, NNP07, IDS05, LG00]. The analysis can proceed in exactly the same way as in the binary setting except for replacing, by $\nu^{\pm m}$, the factors $2^{\pm m}$ that are ubiquitous in the binary setup. The partition of the domain in ν-ary subdivision is uniform in that the domain is split evenly into subdomains. Goldman and Warren [GW93] extend uniform subdivision of curves to knot intervals that are in a globally geometric progression. Sederberg et al. [SSS98] propose Non-uniform Recursive Subdivision Surfaces where every edge of the initial domain is allowed to be scaled differently. This presents challenges, currently not met by the state of the

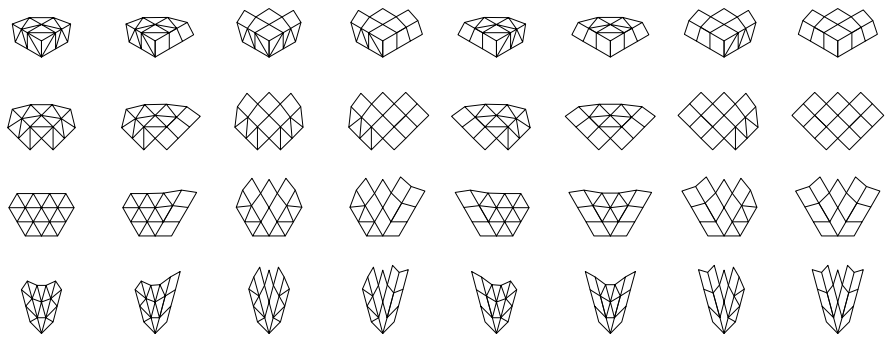

Fig. 9.2: Different patterns of control nets to be considered when proving smoothness of a tri/quad algorithm for (*from top*) $n = 3, 4, 6, 12$.

analysis. By contrast, the *non-uniform* subdivision algorithm [KP07a] has been fully analyzed within the framework presented in this book. Here the edges near a singularity are recursively subdivided in a ratio $\sigma : (1 - \sigma)$ where $\sigma \in (0, 1)$ is chosen to adjust the relative width of the rings and the 'speed of convergence'.

For high valence n, the rings of a subdivision surfaces considered so far, typically feature n sharp corners corresponding to the corners of the domain $\mathbf{S}_n$. Referring to Figure 6.4/120 *right*, we call this arrangement the *sprocket layout*. To remove related shape artifacts, the so-called *polar layout*, as described below, offers an interesting alternative. Recall that, in (3.13/54), we connect the cells $\boldsymbol{\Sigma}$ using the neighborhood relations $(\boldsymbol{\varepsilon}_4, j) \sim (\boldsymbol{\varepsilon}_1, j + 1), j \in \mathbb{Z}_n$. As illustrated in Figure 9.1/185, *right*, the cells can be glued together differently to form a ring structure. In *polar subdivision* [KP07c, KMP06, KP07d]

$$(\boldsymbol{\varepsilon}_4, j) \sim (\boldsymbol{\varepsilon}_2, j + 1), \quad j \in \mathbb{Z}_n. \tag{9.1}$$

That is, *opposite* edges of each cell are identified with the edges of its two neighbors. A ring with polar structure is therefore bounded on either side by a closed smooth curve. The papers on Guided subdivision [KP07b, KP08] juxtapose polar and sprocket layout. It turns out that polar subdivision nicely models features of high valence, especially rotationally extruded features. Again, for the analysis of polar subdivision, the neighbor relations and contact conditions have to be adapted, while the core methodology applies with minor modifications.

A further generalization is obtained when permitting a mix of *quadrilateral and triangular cells* to enclose an extraordinary knot [LL03, SL03, PS04, SW05]. Basically, one has to cope with two technical difficulties not encountered in the standard, symmetric setting. First, due to a lack of shift invariance, we cannot apply the Discrete Fourier Transform to simplify the analysis of spectral properties of the subdivision matrix. Rather, to obtain general results, a multitude of different patterns

has to be analyzed individually. Figure 9.2/186 gives an impression of the combinatorial complexity of the problem. Second, the smooth transition between neighboring patches of different type may be more difficult to establish than between triangles only or squares only. This problem, which is very interesting in its own right, is considered in [Rio92, DL02, LL03, PS04]. For example, [PS04] proves C^1-continuity and bounded curvature for a quad-tri algorithm.