Computer-Aided Design 58 (2015) 173–178

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Control vectors for splines☆

Jiří Kosinka [a,*], Malcolm A. Sabin [b], Neil A. Dodgson [a]

[a] Computer Laboratory, University of Cambridge, 15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom
[b] Numerical Geometry Ltd., 19 John Amner Close, Ely, Cambridge CB6 1DT, United Kingdom

HIGHLIGHTS

- We extend traditional splines based on control points by incorporating control vectors.
- Our paradigm allows combining several spline constructions into one formulation.
- We can model curves and surfaces that are not possible with existing techniques.

ARTICLE INFO

Keywords:
Spline
Curve
Surface
Subdivision
Control vector
Modelling

ABSTRACT

Traditionally, modelling using spline curves and surfaces is facilitated by control points. We propose to enhance the modelling process by the use of *control vectors*. This improves upon existing spline representations by providing such facilities as modelling with local (semi-sharp) creases, vanishing and diagonal features, and hierarchical editing. While our prime interest is in surfaces, most of the ideas are more simply described in the curve context. We demonstrate the advantages provided by control vectors on several curve and surface examples and explore avenues for future research on control vectors in the contexts of geometric modelling and finite element analysis based on splines, and B-splines and subdivision in particular.

1. Introduction

Splines have their roots in the lofting technique used in the shipbuilding and aircraft industries throughout the first half of the 20th century. The first mathematical reference to the notion of splines is accredited to the work of Schoenberg [1] on piecewise polynomial approximation. Later, De Casteljau, Bézier, and De Boor [2] contributed invaluably to the development of splines and B-splines in particular; see [3] for the full story.

In the curve case, the modern understanding of a spline can be, in its generality, captured mathematically by

$$\mathbf{c}(t) = \sum_{i=1}^{n} B_i(t)\mathbf{P}_i; \quad t \in [a, b], \tag{1}$$

where $\mathbf{P}_i$ are control points forming a control polygon and $B_i(t)$ form a set of blending functions that satisfy certain properties required by a particular application. To make the spline curve $\mathbf{c}(t)$ well-defined geometrically (i.e., in order to guarantee shape independence of the choice of origin), it is required that $\sum_{i=1}^{n} B_i(t) \equiv 1$ over $[a, b]$. In other words, the blending functions have to form a *partition of unity*. A desired property in some applications (such as finite element analysis) is linear independence of the blending functions; $B_i(t)$ then form a basis and are called basis functions. In the context of analysis, partition of unity can be relaxed to

$$\exists c_i \quad \text{such that} \sum_{i=1}^{n} B_i(t)c_i \equiv 1. \tag{2}$$

A spline curve is generally composed of many pieces of a particular type (e.g., polynomial, rational, trigonometric) joined together at *knots* with a certain continuity. The most popular examples include B-splines [2] (of which Bézier curves are a special case), trigonometric splines [4], interpolating splines [5], and subdivision curves [6].

The above approach can be easily generalised to surfaces. For example, in the tensor-product case we have

$$\mathbf{s}(u, v) = \sum_{i=1}^{n}\sum_{j=1}^{m} B_i(u)B_j(v)\mathbf{P}_{i,j}; \quad (u, v) \in [a, b] \times [c, d], \tag{3}$$

☆ This paper has been recommended for acceptance by Dr. Vadim Shapiro.

* Corresponding author.
E-mail addresses: jiri.kosinka@cl.cam.ac.uk (J. Kosinka), malcolm.sabin@btinternet.com (M.A. Sabin), neil.dodgson@cl.cam.ac.uk (N.A. Dodgson).

http://dx.doi.org/10.1016/j.cad.2014.08.028

where $\mathbf{P}_{i,j}$ form a rectangular control mesh and the univariate blending functions are reused. To be able to cover also triangular patches, volumetric splines, and other flavours of splines including subdivision, we adopt the general notation

$$\mathbf{c}(\mathbf{t}) = \sum_{i \in I} B_i(\mathbf{t})\mathbf{P}_i; \quad \mathbf{t} \in \Omega, \tag{4}$$

where I is an appropriately chosen index set and Ω is a suitable parameter space spanned by $\mathbf{t}$. The functions $B_i(\mathbf{t})$ form a set of blending functions that partition unity or satisfy (2) over Ω.

2. Control vectors for splines

Imagine a scenario where one needs to edit a local detail on a spline, but the blending functions cannot capture it since they are too coarse (with too large a support). Ideally, one would simply like to be able to add a single new control point associated with a blending function of a desired shape and support. However, due to the form of (4) and the partition of unity constraint, adding a new desired blending function is either impossible or difficult as other functions have to be modified as well while keeping the shape of the spline unmodified.

In the case of B-spline curves, one can use knot insertion to refine the space spanned by the B-splines locally. The situation gets much more complicated in the tensor-product surface setting as local refinement is not possible due to the rectangular structure. Several constructions have been proposed to deal with this: T-splines [7], hierarchical B-splines [8,9], truncated B-splines [10], LR B-splines [11], and T-meshes [12]. In these constructions, local refinement is possible as T-junctions are allowed. However, it proved difficult to maintain partition of unity and linear independence for these constructions without imposing restrictions on refinement strategies or moving to the rational setting by normalisation. In the case of THB-splines [10], basis functions are 'truncated' to maintain both properties, but at the expense of introducing functions that may have more than one maximum. While acceptable in analysis, this is undesirable in modelling as a control point's influence is no longer intuitive.

In the context of finite element analysis, (1) was extended to

$$\mathbf{c}(t) = \sum_{i=1}^{n} B_i(t)\mathbf{P}_i + \sum_{i=1}^{n} f(t)B_i(t)\mathbf{V}_i \tag{5}$$

with some function $f(t)$ well suited for a particular application/problem (a typical example is $f(t) = e^t$) and both $\mathbf{P}_i$ and $\mathbf{V}_i$ are treated as degrees of freedom. This approach and its generalisations were introduced in [13,14], and called Partition of Unity Finite Elements and Extended Finite Elements, respectively. In the latter case, the modification was motivated by introducing cracks ($f(t)$ would be a discontinuous function) without having to remesh. Recently, due to the popularity of Isogeometric Analysis (IgA for short; see [15]), such modifications are ever more important.

These modifications, however, break the partition of unity property and cannot be used for modelling directly, as also required by IgA. To amend the situation and to address the requirements in modelling and analysis, we propose to generalise (4) and (5) to

$$\mathbf{c}(\mathbf{t}) = \sum_{i \in I} B_i(\mathbf{t})\mathbf{P}_i + \sum_{j \in J} C_j(\mathbf{t})\mathbf{V}_j; \quad \mathbf{t} \in \Omega, \tag{6}$$

where $\sum_{i \in I} B_i(\mathbf{t}) \equiv 1$ still holds and $\mathbf{P}_i$ are control points, but $\mathbf{V}_j$ are understood as *control vectors*. While this may seem as semantics only, the transition from control points to control vectors allows the functions $C_j(\mathbf{t})$ to be incorporated in the spline definition; the control vectors do not transform as points, but as displacements. Thus, the partition of unity property no longer applies to the set of $C_j(\mathbf{t})$. However, since (2) still applies, this is not a problem for analysis.

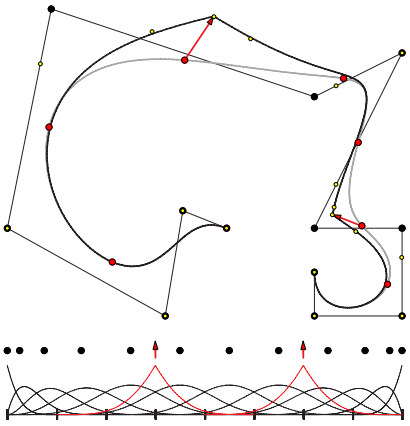

Fig. 1. A degree four spline curve with 12 control points and two non-zero control vectors (red arrows). The associated basis functions are shown in black. The unmodified underlying spline is shown in grey for reference. Control vectors are logically associated with knots. To express the same curve using standard quartic B-splines would require 18 control points (yellow). Each non-zero control vector would give rise to $d - 1 = 3$ extra control points. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Considering a spline as a sum of weighted points plus a sum of weighted control vectors, (6), is a paradigm that opens up a wealth of possibilities with minimal increased cost, as we demonstrate below. For future use, we denote $\mathbf{B}$ the collection of B_i, $i \in I$, and $\mathbf{C}$ the collection of C_j, $j \in J$.

In modelling and other applications, one would associate control vectors with desired blending functions and set their magnitudes to zero. This guarantees that the underlying spline given by $\sum_{i \in I} B_i(\mathbf{t})\mathbf{P}_i$ is recovered and the user (or an error estimator in the case of analysis) can then adjust control vectors to fine-tune the resulting shape or approximation. Moreover, several levels of control vectors can be added to obtain a hierarchical structure that facilitates multiresolution editing; see Fig. 4 for an example. For visualisation and demonstrative purposes, we focus on (local) sharp creases and multiresolution editing throughout this paper, but it should be emphasised that the full scope of our paradigm based on control vectors is not limited to these.

In modelling, the convex hull property is advantageous in some situations. In analysis, functions in $\mathbf{B}$ and $\mathbf{C}$ are required to be linearly independent. However, since the form of (6) is very general, the questions of linear independence and convex hulls need to be investigated on a case-by-case basis.

While control vectors can be applied in any spline scenario covered by (6), we focus only on the most important families used in modelling, animation, and IgA: B-splines (Sections 3 and 5) and subdivision based on B-splines (Section 4).

3. Control vectors for B-spline curves

We now investigate (6) for the case of B-spline curves. In the univariate case, we have $\mathbf{t} = t$ and $\Omega = [a, b]$ in (6). We specialise the basis functions to B-splines [2], but any type of splines can be employed (polynomial, rational, trigonometric, etc.) with $\mathbf{B}$ and $\mathbf{C}$ from the same or different families. While the degrees of $\mathbf{B}$ and $\mathbf{C}$ can be in general different, it is sensible to assume that both sets consist of B-splines of one degree d. Similarly, the knots of $\mathbf{B}$ and $\mathbf{C}$ may be completely unrelated, but it is reasonable to consider only cases where at least some of the knots are aligned.

Fig. 1 shows an example of a degree 4 spline curve, where the knots of $\mathbf{B}$ and $\mathbf{C}$ are shared. The B_i are degree four B-splines

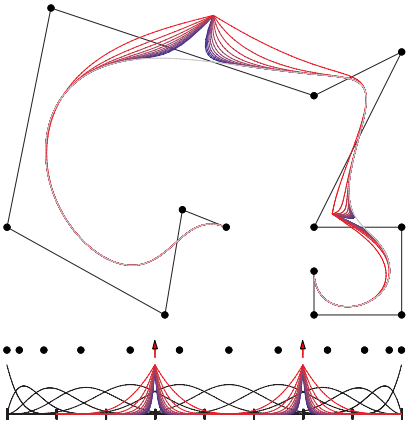

Fig. 2. This example uses the same underlying quartic spline as Fig. 1, but the basis functions associated with control vectors have been raised to a power of $1, 2, \ldots, 10$ (coloured from red to blue). Note that this curve cannot be generated using quartic B-splines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

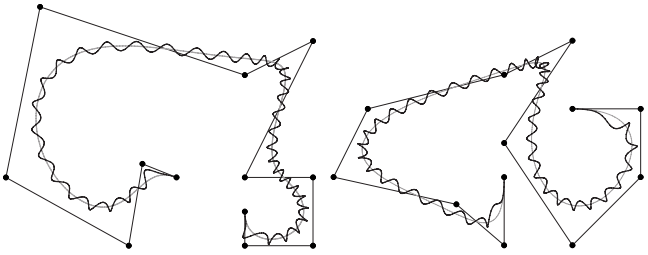

Fig. 3. Control vectors expressed in terms of a local frame. Original configuration (left) and after control points have been moved (right). Note that the details created by control vectors (not shown for clarity) follow the deformation automatically. Both **B** and **C** consist of cubic B-splines, but **C** are defined over a much finer knot vector than that of **B**. The underlying spline is shown in grey.

(bottom; shown in black) defined over an open-uniform knot vector. We show two control vectors, each associated with one of the red basis functions. In this case we have chosen functions which are themselves combinations of degree four B-splines with multiple knots to create sharp features. The resulting spline is shown in black. Note that control vectors (red arrows) are logically associated with knots, and the user can use them as extra degrees of freedom to model sharp features. In terms of user interface, we found it best to anchor control vectors at the images of their associated knots (red circles in the figure) and to initialise them to zero vectors. We note that the red sharp functions can be allowed to 'slide' interactively along the parameter space, resulting in a flexible modelling method.

To emphasise the flexibility of our paradigm, Fig. 2 depicts the same situation as Fig. 1, but this time the sharp (red) basis functions were raised to a power of $1, 2, \ldots, 10$. This demonstrates the power of control vectors. The underlying open-uniform quartic spline does not have to be modified in any way.

Yet another advantage of using control vectors is that they can be expressed in a local adapted frame (e.g. the Frenet frame) of the underlying spline; see Section 4 of [8]. This then allows the user to modify the overall shape of the curve using the original control points, while control vectors are then adjusted automatically according to shape changes performed via control points. An example is shown in Fig. 3. Note that this is closely related to displacement mapping in graphics [16], where individual vertices on a finely sampled or approximated curve (or surface) are displaced according to a texture. In contrast, our method uses splines to displace portions of a curve and is thus more compact and suitable for user-centred modelling and error-driven analysis.

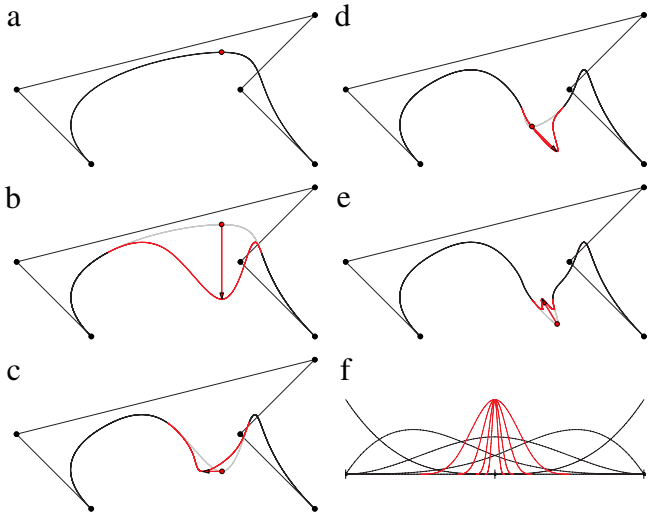

Fig. 4. The input cubic spline (a) and four more levels of hierarchical refinement (b)–(e) via control vectors. Basis functions are shown in (f). To achieve a more intuitive interface, the added basis functions (red in bottom right) have been scaled so that their maxima are equal to 1. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Control vectors and their associated blending functions can come from different levels in a hierarchy. This is demonstrated in Fig. 4 on a cubic spline which is edited on different levels of resolution. This is achieved by using B-splines defined over finer and finer knot vectors. In this particular example, the uniform cubic B-splines (in red) associated with control vectors have been scaled up by the factor of 3/2 to ensure intuitive modelling: ends of control vectors are interpolated.

In general, adding one control vector (or a set thereof) at a time creates a natural hierarchy with the associated spline spaces always nested irrespective of the choice of blending functions in **B** and **C**. For example, the spline spaces associated with (a)–(e) in Fig. 4 form a nested sequence. This is highly desirable in both modelling and analysis, as nestedness enables hierarchical editing and monotone decay of error.

We remark that while the examples presented above used uniform polynomial splines, the concept carries over straightforwardly to non-uniform settings and also to rational and other types of splines, including subdivision splines.

While it would be possible to generate the behaviour facilitated by control vectors using other curve-based mechanisms, we reiterate that the main purpose of this new method is to provide such behaviour for surfaces, where it offers a significant advantage over other mechanisms; see Section 5.

4. Control vectors for subdivision curves

We have seen that extending the traditional spline formulation based on control points to include control vectors as well offers more modelling flexibility. In the case of subdivision, we take the framework one step further and incorporate also the concept of semi-sharpness as defined by Pixar in [17]. But before that, we look at the problem of subdivision suitability.

Consider the univariate uniform setting and suppose that **B** is subdivision suitable, i.e., there exists a refined version **b** of **B** and a subdivision matrix **S** such that $\mathbf{B} = \mathbf{bS}$; cf. [18]. This is a natural assumption: the underlying basis has to support subdivision. In the following, we investigate two options:

- **B** and **C** are refinable separately. This allows us to combine e.g. cubic B-splines and the 6-point scheme [19]. In general, the schemes have to be topologically compatible, i.e., both have to

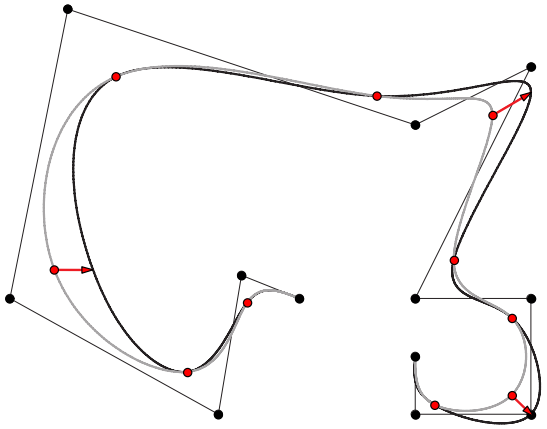

Fig. 5. Uniform cubic B-spline (grey) with the 6-point scheme added to it (black) via control vectors (red). As the 6-point scheme is interpolatory and both schemes are C^2, the resulting mix behaves intuitively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

be of the same arity, and both primal or dual. An example is shown in Fig. 5.

- The blending functions in **C** are refinable with respect to **B** ∪ **C**. An example of this, which generalises Pixar's semi-sharp creases, is shown in Section 4.1.

As in the case of B-splines, similar arguments apply in the non-uniform setting as well.

4.1. Generalising semi-sharp subdivision rules

It is well known that the subdivision mask in the case of the uniform cubic B-spline reads $[1, 4, 6, 4, 1]/8$; see [6]. This in turn leads to subdivision stencils $[1, 6, 1]/8$ and $[4, 4]/8$ for computing new control points, usually called new vertex-vertices and edge-vertices, respectively, from old ones in a uniform cubic spline after one step of subdivision.

If end-point interpolation is desired, the simplest modification of these subdivision rules is to modify the stencils for end-points from $[1, 6, 1]/8$ (smooth rule) to simply $[0, 8, 0]/8 = [0, 1, 0]$ (sharp rule). The same modification can be used to create a crease at any control point. One simply replaces its subdivision stencil by $[0, 1, 0]$. The corresponding point is then interpolated and a sharp crease is created; see Fig. 6.

The main idea concerning creases presented by Pixar [17] is that one can use the sharp rule only for a certain number of subdivision steps, denoted by s and called sharpness, and then switch back to the smooth rule for the rest of the subdivision process. An example with several values of sharpness is shown in Fig. 6. In this approach, no new control points are introduced (in contrast to knot insertion), but every control point P_i is associated with a sharpness s_i.

If a non-integer value of sharpness s is required, one can use linear interpolation between the results corresponding to $\lfloor s \rfloor$ and $\lceil s \rceil$. In the real world, creases on objects are typically semi-sharp: creases appear smooth when inspected from a close distance. Also, in graphics applications it is easier to deal with C^1 (rather than C^0) objects due to the use of normal maps. Semi-sharp creases are thus an important modelling ingredient.

Consider using the sharp rule at a point P for all subdivision steps, i.e., $s = \infty$ and P is interpolated. The associated basis function $D(t)$ is shown in Fig. 7, left. It is easy to verify (see e.g. [18]) that this function can be decomposed into two B-splines as shown in Fig. 7; this observation sparked this research programme. One of these is the original uniform B-spline $B(t)$ associated with the control point P, the other is the B-spline $C(t)$ with a triple knot,

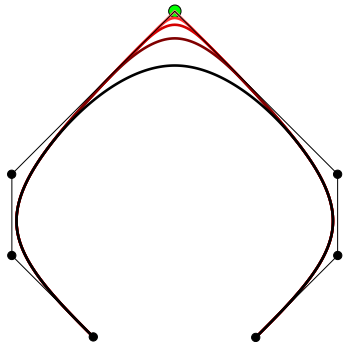

Fig. 6. Semi-sharp creases on a cubic spline. Sharpness values are $s \in \{0, 1, 2, 3, \infty\}$, associated with the green control point. Note that $s = 0$ gives the original smooth spline (black) and $s = \infty$ results in an infinitely sharp crease. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 7. Cubic basis functions and their relation. Left: Crease basis function. Middle: Uniform B-spline. Right: B-spline with a triple knot.

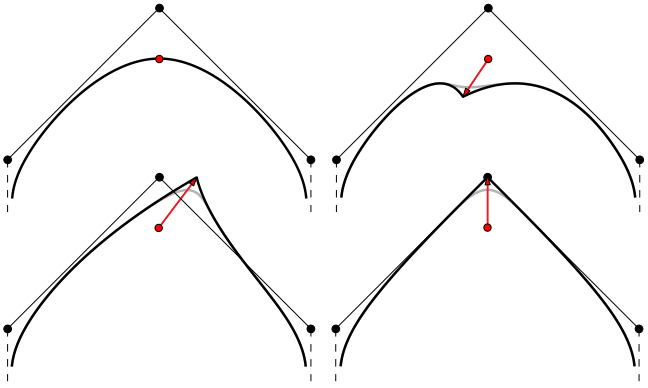

Fig. 8. Control vectors (red) controlling sharp creases (black; $s = \infty$) and semi-sharp creases (grey; $s = 2$) in cubic splines. Top left: $V = \mathbf{0}$ (or $s = 0$) gives the original smooth spline. Bottom right: Setting V appropriately recovers Pixar semi-sharp creases; cf. Fig. 6. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

defined over the knot vector $(-1, 0, 0, 0, 1)$. We associate the latter with a control vector.

More precisely, we have $D(t) = B(t) + \frac{1}{3}C(t)$ in the cubic case. Adding one sharp function per knot leads to

$$\mathbf{c}(t) = \sum_{i=1}^{n} B_{i,4}(t)P_i + \sum_{i=1}^{n} C_{i,4}(t)V_i, \tag{7}$$

where $B_{i,4}(t)$ are uniform cubic B-splines and $C_{i,4}(t)$ are sharp cubic B-splines with a knot of multiplicity 3 in the middle of their support. P_i are the original control points, whereas V_i are control vectors that control sharp creases in an intuitive way; see Fig. 8.

In order to be able to take full advantage of this new approach, including semi-sharp rules, we now show how to handle control vectors in uniform cubic subdivision.

We know that the subdivision mask is $[1, 4, 6, 4, 1]/8$. Thus, it only remains to determine how $C(t)$ is subdivided. Since $C(t)$ is a B-spline defined over the knot vector $(-2, 0, 0, 0, 2)$, it is easy to show that it gives rise to a scaled copy $c(t)$ of itself defined on $(-1, 0, 0, 0, 1)$ and a scaled copy $b(t)$ of the uniform B-spline over $(-2, -1, 0, 1, 2)$. The appropriate combination is $C(t) = \frac{3}{4}b(t) + \frac{1}{2}c(t)$.

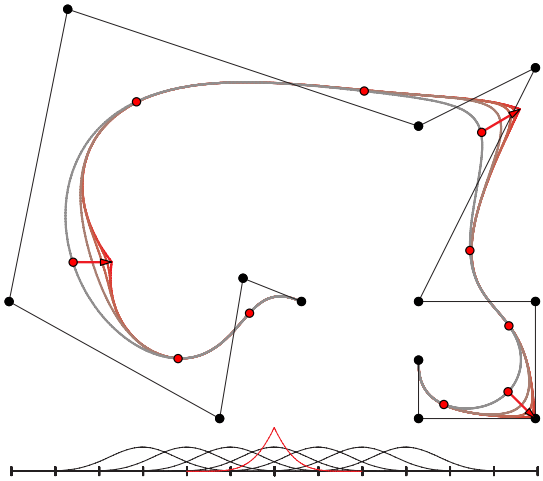

Fig. 9. Semi-sharp creases on a quintic spline. Quintic B-splines $B_{i,6}(t)$ (black) and one copy of the sharp basis function $C_{i,6}(t)$ (red) are also shown. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Consequently, a new vertex–vertex is computed according to $p_i = (P_{i-2} + 6P_i + P_{i+2} + 6V_i)/8$ and the new control vector as $v_i = V_i/2$. The rules for new edge-vertices remain unmodified. This is extremely simple to implement.

In terms of user interface, we found it most convenient to position the control vector V_i at the limit point corresponding to P_i. In the cubic case, this amounts to $(P_{i-1} + 4P_i + P_{i+1})/6$. Moreover, we initialise the position of the endpoint of V_i to P_i. This directly generalises Pixar creases. The user is then free to adjust the magnitude and orientation of V_i, which controls the crease; see Fig. 8.

It is straightforward to incorporate sharpness in the spirit of Pixar to our method. A value of s is associated with every control vector, initialised to zero. Our crease subdivision rules are then used only in the first s subdivision steps before switching to smooth rules (which is equivalent to setting V to zero at the appropriate level of subdivision); see the grey curves in Fig. 8. As in the Pixar method, non-integer values of s can be dealt with either by linear interpolation, or more conveniently in our formulation by scaling V down by s in the step when $s < 1$.

This new approach has the following properties:

- Linear independence of $\mathbf{B} \cup \mathbf{C}$: follows from the fact that all the functions involved are B-splines.
- Trivial initialisation: set all sharpness values to zero.
- Generalises univariate semi-sharp Pixar rules.
- Full cubic polynomial reproduction: Pixar creases miss quadratics; see [18].

According to our initial tests based on the approach developed in [18], the formulation in (7) can be generalised to support any degree. This is work in progress. An example for degree 5 is shown in Fig. 9.

5. Control vectors for surfaces

The univariate case is interesting in its own right for curve modelling and also since curves appear naturally on surfaces as boundaries and creases. All our results from the curve case generalise to tensor products in a straightforward fashion. However, control vectors offer much more than that.

We can now create features that are not aligned with knot lines in $\mathbf{B}$ by rotating the domain of (some of the functions of) $\mathbf{C}$. This is reminiscent of point-based splines [7]. An example is shown in Fig. 10. We emphasise that any angle can be used and that the crease can, as in the univariate case, be controlled by a semi-sharpness value. For rendering and analysis, the triangulation (partitioning into elements) of the surface (parameter space) needs to respect the features; see Fig. 10, left. Similarly, one can also combine quadrilateral and triangular splines, or any other available spline constructions.

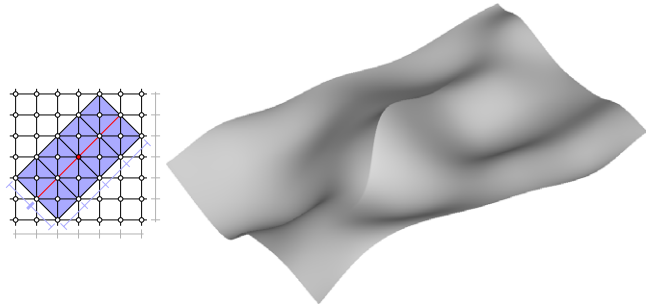

Fig. 10. A bicubic tensor-product spline with a vanishing diagonal crease added to it via a control vector. The associated basis function (whose support is shown in blue) is not aligned with the underlying rectangular structure. We emphasise that this surface cannot be generated using standard B-splines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

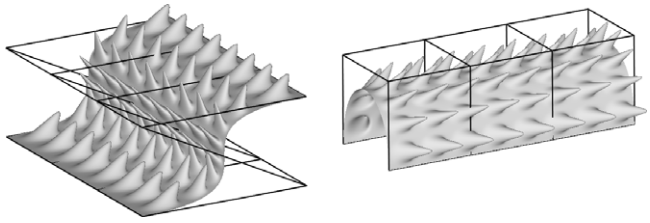

Fig. 11. An input surface with bumps added via control vectors (left) and after control points have been moved (right). As control vectors are represented in local surface-aligned frames, the bumps follow the deformation automatically.

As in the univariate case (cf. Fig. 3), control vectors can be expressed in a local coordinate frame, e.g. determined from the first derivatives and the normal vector of a regular surface; see Section 4 of [8]. A deformation governed solely by control points that respects local feature alignment via control vectors is shown in Fig. 11.

Adding a basis function (or a set thereof) associated with a control vector at a time results in a hierarchy of refinements and a sequence of nested spaces. Therefore, our framework, with tensor-product B-splines at the coarsest level represented by $\mathbf{B}$, can reproduce surfaces modelled by a number of existing methods including [8–10]. It should also be noted that any of these methods can be used to construct $\mathbf{B}$ and then continue refining/adding features of various types via control vectors associated with functions in $\mathbf{C}$.

Subdivision surfaces offer hierarchical refinement via multiresolution editing. On the other hand, adding semi-sharp creases and other features often requires a modification of standard subdivision rules, as e.g. in the case of Pixar [17]. Our initial tests show that our univariate subdivision scheme based on (7) can be extended to Catmull–Clark subdivision surfaces [20] and a control vector can be attached to any control point that is not an extraordinary point, i.e., a point of valency other than four [21]. Control vectors associated with extraordinary vertices present an interesting avenue for future research.

It should also be noted that our framework is not limited to surfaces and subdivision schemes based on quadrilaterals. For example, in the spirit of Fig. 5, the combination of the approximating Loop subdivision [22] with the interpolatory butterfly scheme [23], both based on triangular meshes, is appealing.

6. Conclusion

We have proposed an extension to the existing spline paradigm based on control points by including support for control vectors. Our examples demonstrate that our framework offers numerous new modelling scenarios with close links to analysis provided by the isogeometric concept. We have only touched on the full potential of control vectors and opened up new avenues for future work.

We envision that the use of control vectors in splines, and NURBS and subdivision in particular, will greatly enhance model design and analysis in the future.

Acknowledgements

The authors thank EPSRC for supporting this work through Grant EP/H030115/1 and the anonymous reviewers for their helpful insights.

References

[1] Schoenberg IJ. Contributions to the problem of approximation of equidistant data by analytic functions. Quart Appl Math 1946;4: 45–99 and 112–141.
[2] de Boor C. On calculating with B-splines. J Approx Theory 1972;6(1):50–62.
[3] Boehm W, Müller A. On de Casteljau's algorithm. Comput Aided Geom 1999; 16(7):587–605.
[4] Schoenberg IJ. On trigonometric spline interpolation. J. Math. Mech. 1964;13: 795–825.
[5] Subbotin YN. Interpolating splines, In: Cieselski Z, Musielak J, editors Approximation theory, Reidel, 1975, pp. 221–34.
[6] Sabin M. Analysis and design of univariate subdivision schemes. Springer; 2010. ISBN: 978-3-642-13647-4.
[7] Sederberg TW, Zheng J, Bakenov A, Nasri A. T-splines and T-NURCCs. ACM Trans. Graph. 2003;22:477–84.
[8] Forsey DR, Bartels RH. Hierarchical B-spline refinement. In: SIGGRAPH '88. ACM; 1988. p. 205–12.
[9] Vuong A-V, Giannelli C, Jüttler B, Simeon B. A hierarchical approach to adaptive local refinement in isogeometric analysis. Comput Methods Appl Mech Engrg 2011;200(49–52):3554–67.
[10] Giannelli C, Jüttler B, Speleers H. THB-splines: the truncated basis for hierarchical splines. Comput Aided Geom Design 2012;29(7):485–98.
[11] Dokken T, Lyche T, Pettersen K. Polynomial splines over locally refined box-partitions. Comput Aided Geom Design 2013;30(3):331–56.
[12] Deng J, Chen F, Li X, Hu C, Tong W, Yang Z, Feng Y. Polynomial splines over hierarchical T-meshes. Graph. Models 2008;70(4):76–86.
[13] Melenk J, Babuška I. The partition of unity finite element method: Basic theory and applications. Comput Methods Appl Mech Engrg 1996;139(1–4):289–314.
[14] Belytschko T, Black T. Elastic crack growth in finite elements with minimal remeshing. Internat J Numer Methods Engrg 1999;45(5):601–20.
[15] Hughes T, Cottrell J, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Engrg 2005;194(39–41):4135–95.
[16] Cook RL. Shade trees, SIGGRAPH '84, ACM, 1984, pp. 223–31.
[17] DeRose T, Kass M, Truong T. Subdivision surfaces in character animation. In: SIGGRAPH '98. ACM; 1998. p. 85–94.
[18] Kosinka J, Sabin MA, Dodgson NA. Creases and boundary conditions for subdivision curves. Graph Models 2014;76(5):240–51.
[19] Weissman A. A 6-point interpolatory subdivision scheme for curve design. (M.s. thesis), Tel-Aviv University; 1989.
[20] Catmull E, Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. Comput-Aided Des 1978;10(6):350–5.
[21] Kosinka J, Sabin MA, Dodgson NA. Semi-sharp creases on subdivision curves and surfaces. Comput Graph Forum 2014;33(5):217–26.
[22] Loop C. Smooth subdivision surfaces based on triangles. (M.S. mathematics thesis), University of Utah; 1987.
[23] Dyn N, Levin D, Gregory JA. A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans. Graph. 1990;9(2):160–9.