Colour Interpolants for Polygonal Gradient Meshes

Hettinga, Gerben J.; Brals, René; Kosinka, Jiří

Published in:
Computer aided geometric design

DOI:
10.1016/j.cagd.2019.101769

Publication date:
2019

Colour Interpolants for Polygonal Gradient Meshes

Gerben J. Hettinga [1], René Brals [1], Jiří Kosinka [1]

[1] Bernoulli Institute, University of Groningen, The Netherlands

Abstract

The gradient mesh is a powerful vector graphics primitive capable of representing detailed and scalable images. Borrowing techniques from 3D graphics such as subdivision surfaces and generalised barycentric coordinates, it has been recently extended from its original form supporting only rectangular arrays to (gradient) meshes of arbitrary manifold topology.

We investigate and compare several formulations of the polygonal gradient mesh primitive capable of interpolating colour and colour gradients specified at the vertices of a 2D mesh of arbitrary manifold topology. Our study includes the subdivision based, topologically unrestricted gradient meshes [20] and the cubic mean value interpolant [19], as well as two newly-proposed techniques based on multisided parametric patches building on the Gregory generalised Bézier patch and the Charrot-Gregory corner interpolator. We adjust these patches from their original geometric 3D setting such that they have the same colour interpolation capabilities as the existing polygonal gradient mesh primitives. We compare all four techniques with respect to visual quality, performance, mathematical continuity, and editability.

Keywords: *vector graphics; colour interpolation; generalised barycentric coordinates; multisided patches.*

1 Introduction

The interpolation of colour through the use of a mesh-like structure has many interesting applications in vector graphics, as opposed to raster graphics, and image vectorisation [3], the conversion of a raster image to a vector representation. Many of the known algorithms prevalent in CAGD for interpolation of 2D and 3D data can be adjusted and adapted for colour interpolation. However, the interpolation of colour has some additional challenges, often requiring adjustments in the way data is interpolated, making it an interesting problem. In essence, colour interpolation in vector graphics is done in a 5-dimensional space wherein both positional data, the (x, y) coordinates in the image plane, and the colour values, often specified as (r, g, b) triples of red, green and blue, should be interpolated. However, colour spaces, including RGB spaces, are bounded by their gamut and care must be taken to stay within it. Otherwise, colour overflow or clipping may occur, resulting in sharp discontinuities in colour. At the same time, colour values specified by the user, or sampled from the underlying image in the setting of vectorisation, have to be interpolated, as otherwise washed-out colours are obtained instead; this poses yet another challenge.

In vector graphics, it is important to be able to deliver a high quality image at any resolution. Moreover, the vector representation should subsequently be useful for an end-user. The user should still be able to easily edit the image through the manipulation of primitives, the basic building blocks of vector graphics representations. These primitives can be anything from simple lines or polylines, squares, circles, or linear and elliptical colour gradients. However, it is preferable to have support for more complex colour variation such that a wider range of smooth colour transitions can be modelled, such as a blush apparent on a human face or the subtle change in shading on the hood of a car.

Current vector graphics authoring tools such as Adobe Illustrator, CorelDRAW and Inkscape make use of the (traditional) gradient mesh primitive limited to a mesh with rectangular topology, supporting smoothly varying colour transitions and curved geometry through the use of tangent handles or vectors. Adding detail to the mesh requires a global mesh refinement step, as the refinements will propagate through-out the mesh, owing to the rigid rectangular topology restriction. Recently, Barendrecht et al. [2] have developed exact local refinement to alleviate this restriction. Still, it remains a problem to fit a gradient mesh to an arbitrarily shaped region.

In most cases it might be better to use a planar mesh of arbitrary manifold topology, which we call simply a polygonal mesh in this paper, rather than to twist, tweak and overlay rectangular arrays to obtain the desired result. Polygonal (gradient) meshes [19, 20] have a number of advantages with respect to smooth colour interpolation and the application of the meshes. For instance, polygonal meshes allow for a sparser representation than an ordinary quadrilateral mesh, as they support multisided patches and allow for any number of patches to meet at a vertex. With this, just as in 3D modelling where quad-dominant meshes are (often inevitably) allowed to contain triangles, pentagons, and also vertices where other than four quads meet, it allows for a more intuitive and organic design process as the designer is not restricted by the rectangular topology. For vectorisation processes it might be advantageous to use arbitrary topology as this does not impose the problem of fitting a rectangular mesh structure to arbitrarily shaped regions.

In this paper, we investigate and compare several multisided colour interpolants that expand on and generalise the abilities of the traditional gradient mesh primitive. We call these types of meshes generalised gradient meshes. We compare the existing topologically unrestricted gradient mesh [20] and the cubic mean value interpolant [19], as well as the Gregory generalised Bézier patch [15] and the Charrot-Gregory corner interpolator patch [6], both of which we make suitable for fully local, per-patch, globally G^1 smooth colour interpolation.

In Section 2 we describe related work and also the techniques for topologically unrestricted gradient meshes [20] and the cubic mean value interpolant [19] in detail. We also deal with the general constructions for the (Gregory) generalised Bézier patch and the Charrot-Gregory corner interpolator. Then in Section 3 we detail how to adjust these (multisided) patch constructions for fully local colour interpolation with the help of generalised barycentric coordinates. We discuss how the method of Chiyokura and Kimura [7] can be adapted to handle colour interpolation and how we increase the utility of these primitives by allowing non-convex corners. Then in Section 4 we compare and evaluate the new and known techniques to each other on a range of different meshes, and with respect to performance. The paper is concluded in Section 5.

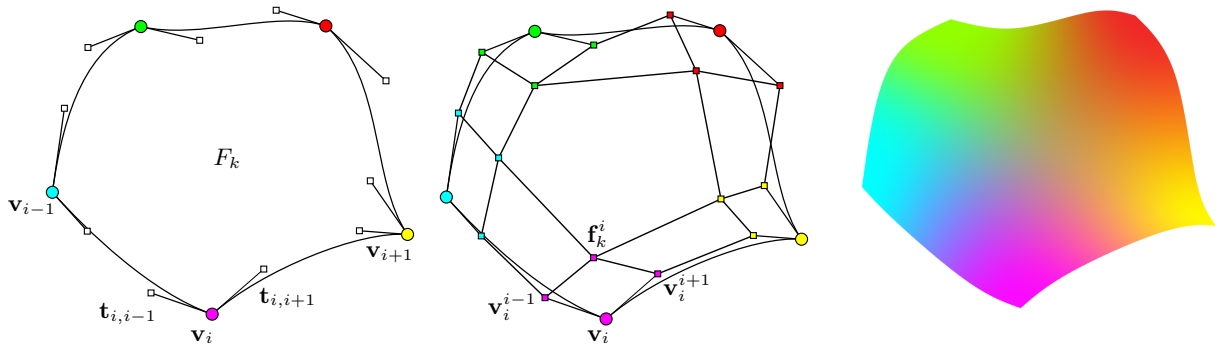


Figure 1: *Left: The input data given to the topologically unrestricted gradient mesh (one face, F_k, is shown): vertices and gradients for each edge. Middle: The construction of the new vertices from the data proceeds via a modified linear ternary subdivision step. Notice that the new vertices inherit the colour value of their logically closest original vertex; this ensures colour interpolation. Right: Subsequently, Catmull-Clark subdivision, run in $\mathbb{R}^5$ to accommodate position and colour, is used to obtain the final image.*

2 Related Work and Preliminaries

In this section we describe several existing works and some of the building blocks of our new solutions for multisided colour interpolation.

2.1 Gradient Mesh

A traditional gradient mesh is a rectangular array of quads given by (colour) vertices $\mathbf{v} = [x, y; r, g, b] \in \mathbb{R}^{2+3}$ and associated colour gradient handles, understood as vectors $\mathbf{t}_u$ and $\mathbf{t}_v$, also in $\mathbb{R}^5$. The gradients are only spatial gradients, meaning that they only provide derivatives for the position (with respect to a (u, v) parametrisation) rather than the colour. The colour component of the colour gradients is set to zero, i.e., the (control) points given by $\mathbf{v} + \mathbf{t}_u$ and $\mathbf{v} + \mathbf{t}_v$ have the same colour as $\mathbf{v}$. This assures interpolation of colour specified at vertices and also that the actual colour interpolated over each patch never leaves the prescribed gamut, as the boundary curves provide monotonic colour interpolation between its two end-point colour values. When adjusting the gradients, the surface changes in a way that displaces the geometry and thus changes the extent and orientation of the colour gradient originating from each vertex.

Note that in the context of vector graphics, the term 'gradient' is interpreted rather loosely and refers to the propagation and transitions of colour. In the strict mathematical sense, the gradient of the parametrisation of the patch is given by $\nabla\mathbf{v} = (\mathbf{t}_u, \mathbf{t}_v)$, and this is nicely aligned in the setting of traditional gradient meshes. But as this analogy breaks down in our setting where a vertex can have an arbitrary number of incident edges and thus 'gradients' emanating from it, we use the term gradient in the informal, loose sense.

Traditionally, the gradient mesh is represented as a rectangular mesh of Ferguson patches [10, 2]. In this paper, we choose to represent it as a mesh of quadrilateral Bézier patches such that the inner control points of the boundary curves provide the same tangent handle functionality. The patches provide a C^1 geometry interpolation and colour surface with the restriction that the gradient handles $\mathbf{t}_u$ and $\mathbf{t}_v$ at shared vertices are also shared, i.e., remain co-linear and of equal magnitude. In essence, the designer manipulates only the curve network of the gradient mesh through the manipulation of vertex positions and the gradient handles, and this in turn controls the spread of colour (or colour gradient) in the interior of the mesh.

This quadrilateral gradient mesh primitive is used in many vector graphics authoring applications such as Adobe Illustrator [30], CorelDRAW [8] and Inkscape [1]. It has also been used as a basis for various image vectorisation algorithms [25, 18].

When modelling or vectorising highly complex images, a lot of small primitives need to be created in areas of the image where a lot of detail is present. The regular structure of the mesh quickly increases the topological complexity of the mesh as only global refinement is possible. To this end, exact local refinement was brought to the gradient mesh primitive [2], such that it supports T-junctions and also branching. Still, the gradient mesh remains mainly regular in structure. This still complicates the construction of a sparse mesh for an image area with an arbitrarily shaped, possibly non-convex, boundary.

To alleviate this, several constructions which support arbitrary (gradient) meshes have been proposed, as described below.

2.2 Topologically Unrestricted Gradient Meshes

The topologically unrestricted gradient mesh [20] is a gradient mesh of arbitrary topology, meaning that it can contain faces with more or less than four sides and have any number of faces meeting at a vertex. This gradient mesh is evaluated by means of Catmull-Clark subdivision [5], although it first requires a special ternary subdivision step to enforce the interpolation of colour by introducing a ring of vertices around each initial vertex with the same colour; see Figure 1. This mimics the colour component of the gradients of the traditional gradient mesh and offers the user a similar interface for manipulating colour propagation.

This ring of vertices around $\mathbf{v}_i$ is constructed from the specified outgoing gradient handles $\mathbf{t}_{i,j}$ at $\mathbf{v}_i$ and the centroids $\mathbf{c}_k$ of incident faces F_k. It creates a new vertex for each outgoing edge by simply translating the vertex by its outgoing gradient vector, $\mathbf{v}_i^j = \mathbf{v}_i + \mathbf{t}_{i,j}$, where $\mathbf{t}_{i,j}$ is the gradient vector along the edge between vertices $\mathbf{v}_i$ and $\mathbf{v}_j$. The face points $\mathbf{f}_k^i$ are constructed by taking into account the adjacent vertices (labelled as $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$ in Figure 1, left) of the current vertex $\mathbf{v}_i$ with respect to the incident face F_k:

$$\mathbf{f}_k^i = (1-d_1)(1-d_2)\mathbf{v}_i + d_1 d_2 \mathbf{c}_k + (1-d_1)d_2\mathbf{d}_{i-1} + d_1(1-d_2)\mathbf{d}_{i+1},$$

where $d_j = 2\frac{||\mathbf{t}_{i,j}||}{||\mathbf{v}_i - \mathbf{v}_j||}$, $\mathbf{d}_j = \mathbf{v}_i + \frac{||(\mathbf{v}_i+\mathbf{v}_j)||}{2}\hat{\mathbf{t}}_j$, $\hat{\mathbf{t}}_{i,j}$ is $\mathbf{t}_{i,j}$ normalised, and $\mathbf{c}_k$ is the centroid of face F_k.

The ternary step creates a flat colour spot at the control vertices of the mesh as the colour components of the new one-ring vertices inherit the colour value of the centre vertex. In doing this, the interpolation of colour is ensured at the vertex. After this procedure the resulting mesh can be further subdivided using ordinary binary Catmull-Clark subdivision. The resulting geometry and colour surface will be C^2 nearly everywhere and G^1 at extraordinary vertices.

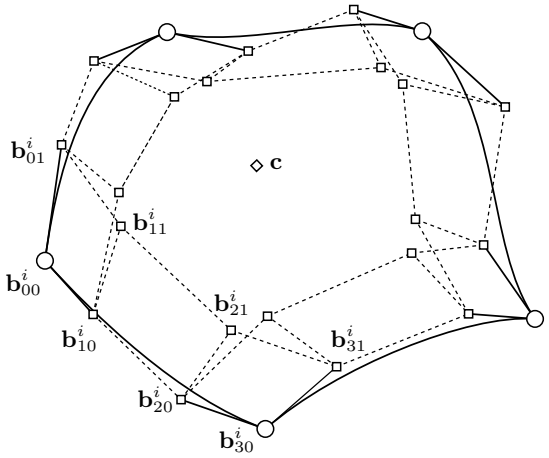


Figure 2: *The control net of a cubic generalised Bézier patch. The labelling of the control points corresponding to one side is given, the rest is labelled analogously.*

Recently, the viability of using this as a vector graphics primitive has been extended by allowing for local and hierarchical editing [28] via control vectors [17] and by specifying the colour and position/displacement of vertices at any subdivision level. For visibility of the hierarchical edits, it requires the mesh to be subdivided to at least the level in which the edits are placed.

2.3 Cubic Mean Value Coordinates

The cubic mean value interpolant [19] is a multisided interpolant capable of interpolating vertex positions, associated function values, as well as gradients. It is based on a closed-form version of the Hermite interpolant [13], which is in turn based on mean value coordinates [11].

The interpolant interpolates values f_i defined at the vertices $\mathbf{v}_i$ along with gradient constraints which are, at each vertex, decomposed into along-edge derivatives f_i^+ and f_i^-, and across-edge (i.e., outgoing normal) derivatives h_i^- and h_i^+. On edges, f is piece-wise cubic and thus perfectly matches the setting of the traditional gradient mesh interpolant. Overall, the cubic mean value interpolant is a weighted sum of its control values and derivatives, and takes the following form over an n-sided polygonal face:

$$f = \sum_{i=0}^{n-1} a_i f_i + \sum_{i=0}^{n-1} \sum_{s\in\{-,+\}} b_i^s f_i^s + \sum_{i=0}^{n-1} \sum_{s\in\{-,+\}} c_i^s h_i^s.$$

The exact forms of the a_i, b_i and c_i weight functions, cubic mean value coordinates, can be found in the original paper [19]. Recently, a quadratic variant has appeared [4].

The usefulness of the interpolant is extended by allowing the polygonal shape to be altered by specifying tangent vectors at vertices of the polygon such that the edges of the polygon become cubic Bézier curves. Using these gradients, the cubic mean value interpolant achieves the same expressiveness as the gradient mesh primitive, as the edges allow free-form curve design. The auxiliary gradient handles define the derivatives f_i^+ and f_i^- and in turn the outward normal components h_i^- and h_i^+ of the gradient at each vertex. As in the case of the subdivision-based approach, all the colour components of the gradients/derivatives at the vertices are set to zero and the colour at vertices is specified via f_i.

Ideally, one would like to use the derivatives f_i^+ and f_i^- completely freely, even at a vertex of arbitrary valency in a polygonal mesh. However, the method is not able to guarantee smooth interpolation when an arbitrary number of edges meet at a shared vertex since the gradient constraints along (and across) the incident edges may not be compatible, which results in only C^0 continuity at these extraordinary points. This can be fixed by finding a certain best-fitting gradient to all the gradient constraints and adjusting these constraints to match the found gradient, thus achieving global G^1 continuity over the whole mesh, but at the expense of changing some of the user-specified gradient handles.

The authors of [19] use the cubic mean value gradient mesh primitive as an adaptive primitive in that it can be used to simplify and coarsen an existing traditional gradient mesh by greedily merging adjacent primitives. As the interpolant can handle faces of any valency, this is perfectly possible even for non-convex regions. However, as in the case of standard mean value coordinates, some of the coordinates may become negative in non-convex regions, which then leads to C^0 colour surfaces as the interpolated colour has to be clamped to its gamut; see also [21].

2.4 Gregory Generalised Bézier Patches

Multisided parametric patches are usually parametrised using some form of generalised barycentric coordinates. Generalised barycentric coordinates [12] provide a system of coordinates in which any point on a planar polygon can be expressed as a weighted combination of the polygon's vertices.

More specifically, let ϕ_i be functions defined over a polygon P with cyclically ordered vertices $\mathbf{v}_i$, $i = 1 \dots n$. The ϕ_i are called generalised barycentric coordinates with respect to P if they partition unity over P, i.e., $\sum_{i=1}^n \phi_i = 1$, and if they satisfy the so-called barycentric property $\sum_{i=1}^n \phi_i(\mathbf{p})\mathbf{v}_i = \mathbf{p}$ for any $\mathbf{p} \in P$. It is also often required that the ϕ_i are non-negative over P, i.e., $\phi_i \geq 0, \forall i$, especially in the context of colour interpolation. When continuous, the coordinate functions linearly interpolate on the boundary, and at the vertices of P they satisfy the Lagrange property: $\phi_i(\mathbf{v}_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta. Prime examples of generalised barycentic coordinates include Wachspress coordinates [29] and the already mentioned mean value coordinates [11], but many other types exist [12].

Generalised Bézier patches [27, 15, 22], or GB patches, are multisided control point patches that generalise tensor-product Bézier patches to a patch with an arbitrary number of sides. Although these patches can be of any degree, in this paper we are only interested in the cubic variant. Generalised barycentric coordinates are used to define local parameters that interpolate Bézier ribbons and blending functions that weigh the contributions of these ribbons. An n-sided GB patch is a combination of n

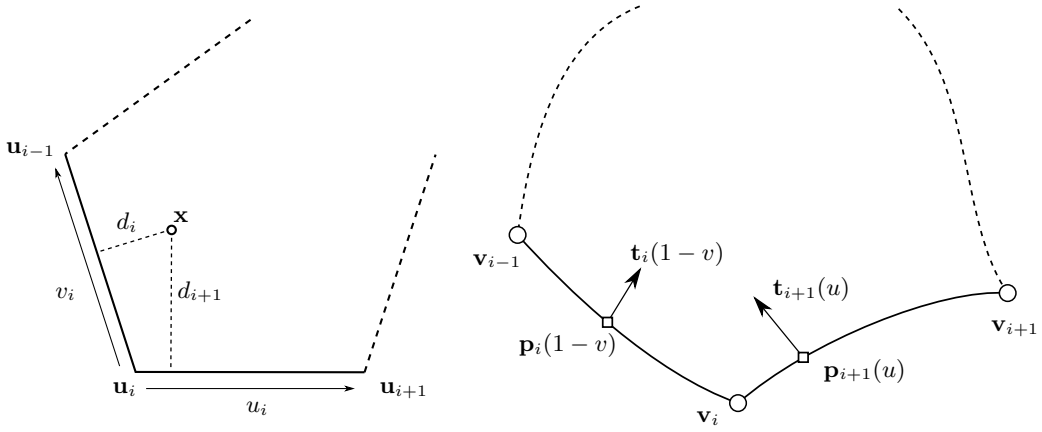


Figure 3: *Left: The (regular) polygonal parameter space of the Charrot-Gregory patch with distances to edges used to establish local coordinate systems at each corner. Right: A corner interpolator* $\mathbf{c}_i$ *corresponding to* $\mathbf{v}_i$ *is given by adjacent (cubic) edges and associated cross-tangent fields.*

Bézier ribbons, a specific section of a tensor-product Bézier surface defined by taking into account $l = (d+1)$ div 2 rows of $d+1$ control points, where d is the degree of the ribbon. Each ribbon is defined using Bernstein basis polynomials $B_m^d(\cdot)$ of degree d as

$$\mathbf{r}_i^d(s_i, h_i) = \sum_{j=0}^{d} \sum_{k=0}^{l-1} \mu_{jk}^i \mathbf{b}_{jk}^i B_j^d(s_i) B_k^d(h_i).$$

For ribbon $\mathbf{r}_i$ of a patch, which corresponds to the edge $\mathbf{e}_i$ spanned from vertex $\mathbf{v}_{i-1}$ to $\mathbf{v}_i$, local parameter functions, having the same behaviour as bilinear coordinates, are defined using generalised barycentric coordinates ϕ_i as

$$s_i = \frac{\phi_i}{\phi_i + \phi_{i-1}}, \quad h_i = 1 - \phi_i - \phi_{i-1}.$$

Here, s_i varies from 0 to 1 on $\mathbf{e}_i$, and h_i is 0 on $\mathbf{e}_i$ but increases towards the interior of the patch and behaves linearly on $\mathbf{e}_{i-1}$ and $\mathbf{e}_{i+1}$. The control points $\mathbf{b}_{jk}^i$ are oriented and labelled from $\mathbf{e}_i$; see Figure 2. The scalar functions μ_{jk}^i blend control points which are shared by multiple sides such that they are weighted by a linear combination of Bernstein polynomials. For $d = 3$, they are defined as

$$\mu_{jk}^i = \begin{cases} \frac{h_{i-1}}{h_i + h_{i-1}} & j < 2 \\ 1 & 2 \le j \le d-1 \\ \frac{h_{i+1}}{h_i + h_{i+1}} & j > d-1 \end{cases}. \tag{1}$$

As the sum of the resulting blending functions is not equal to one, a central control point $\mathbf{c}$ can be introduced to ensure partition of unity

$$\sum_{i=1}^{n} \mathbf{r}_i^d(s_i, h_i) + \left(1 - \sum_{i=1}^{n} \sum_{k=0}^{l-1} \sum_{j=0}^{d} \mu_{jk}^i B_j^d(s_i) B_k^d(h_i)\right) \mathbf{c}.$$

Alternatively, the weight deficiency can be solved by simply normalising the GB patch

$$\frac{\sum_{i=1}^{n} \mathbf{r}_i^d(s_i, h_i)}{\sum_{i=1}^{n} \sum_{k=0}^{l-1} \sum_{j=0}^{d} \mu_{jk}^i B_j^d(s_i) B_k^d(h_i)}.$$

This then completes the GB patch definition.

The Gregory generalised Bézier patch [15] or generalised Gregory patch is a simple extension of the ordinary GB patch and generalises the quadrilateral Gregory patch [7, Section 5.1]. The boundary curves and ribbons are given as cubic Bézier curves and surfaces, respectively. It removes the need for twist-compatibility of the adjacent Bézier ribbons within a patch such that the patches can join with tangent-plane (G^1) continuity at extraordinary vertices. It does so by removing the requirement that adjacent tangent ribbons explicitly share their inner control points in the first row from the boundary. By allowing this, the patches are able to express different twist vectors at the vertices, and the blending functions μ_{jk}^i then ensure that the control points in patch corners are blended much like the rational blending found in the original quadrilateral version of the Gregory patch [7].

While (Gregory) GB patches blend edge interpolators (ribbons), it is also possible to approach the problem of constructing multisided patches by making use of corner interpolators, as described in the next section.

2.5 The Charrot-Gregory Corner Patch

The Charrot-Gregory corner interpolation patch [6] is an n-sided parametric patch which blends n corner interpolator functions. Consider a multisided patch bounded by (cubic) curves $\mathbf{p}_i(u)$, $u \in [0,1]$ connecting vertices $\mathbf{v}_{i-1}$ and $\mathbf{v}_i$. Each curve is associated with a differentiable vector function $\mathbf{t}_i(u)$, $u \in [0,1]$ which expresses the transversal tangent field along $\mathbf{p}_i(u)$; see Figure 3. At their endpoints, these tangent fields are assumed to agree with their adjacent curves, i.e.,

$$\begin{aligned} \mathbf{t}_i(0) &= -\tfrac{\partial}{\partial u} \mathbf{p}_{i-1}(u)|_{u=1}, \\ \mathbf{t}_i(1) &= \tfrac{\partial}{\partial u} \mathbf{p}_{i+1}(u)|_{u=0}. \end{aligned} \tag{2}$$

The Charrot-Gregory corner interpolator combines two adjacent curves $\mathbf{p}_i(u)$ and $\mathbf{p}_{i+1}(u)$, i.e., $\mathbf{p}_i(1) = \mathbf{p}_{i+1}(0) = \mathbf{v}_i$, and their transversal tangent fields $\mathbf{t}_i(u)$ and $\mathbf{t}_{i+1}(u)$ into the corner interpolator

$$\mathbf{c}_i(u,v) = -\mathbf{p}_{i+1}(0) - v\mathbf{t}_{i+1}(0) - u\mathbf{t}_i(0) + \mathbf{p}_{i+1}(u) + v\mathbf{t}_{i+1}(u) + \mathbf{p}_i(1-v) + u\mathbf{t}_i(1-v) - uv\frac{v\frac{\partial}{\partial v}\mathbf{t}_i(v)|_{v=1} + u\frac{\partial}{\partial u}\mathbf{t}_{i+1}(u)|_{u=0}}{u+v},$$

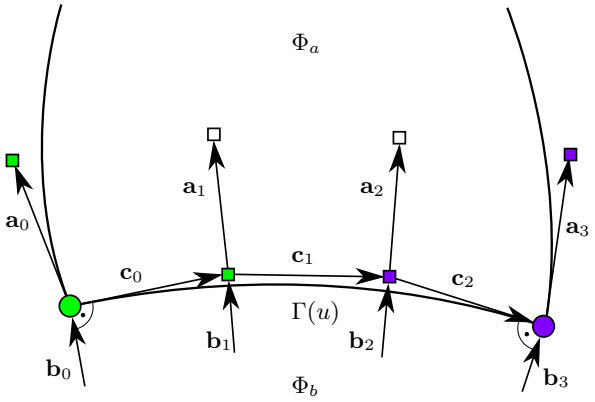


Figure 4: *A schematic view of the boundary conditions between the basis patch Φ_b and the actual patch Φ_a. The colour components of the control points are visualised.*

which interpolates $\mathbf{p}_i(u)$ and $\mathbf{p}_{i+1}(u)$ and their respective tangent fields. The vectors $\frac{\partial}{\partial v}\mathbf{t}_i(v)|_{v=1}$ and $\frac{\partial}{\partial u}\mathbf{t}_{i+1}(u)|_{u=0}$ signify the twist vectors at $\mathbf{v}_i$, which are rationally blended to resolve the twist-compatibility condition. To complete the patch definition, each corner patch is weighted by

$$w_i(\mathbf{x}) = \frac{\prod_{j\neq i,i+1} d_j^2}{\sum_{k=0}^{n-1}\prod_{l\neq k,k+1} d_l^2}, \tag{3}$$

where $\mathbf{x}$ is a point in the (regular) polygonal domain of the patch and d_m is the perpendicular distance from $\mathbf{x}$ to the polygon's edge corresponding to $\mathbf{p}_m(u)$, as shown in Figure 3. In turn, each corner interpolator is supplied with local variables $u_i(\mathbf{x}) = \frac{d_i}{d_i+d_{i+2}}$ and $v_i(\mathbf{x}) = \frac{d_{i+1}}{d_{i-1}+d_{i+1}}$. The final patch definition then becomes

$$\sum_{i=0}^{n-1} w_i(\mathbf{x})\mathbf{c}_i(u_i(\mathbf{x}), v_i(\mathbf{x})),$$

which interpolates the supplied boundary data positionally and tangentially. We modify this definition in the next section to make the patch suitable for polygonal gradient mesh colour interpolation.

3 Local G^1 Colour Interpolation

In this section we detail how to adjust the (Gregory) generalised Bézier patch and the Charrot-Gregory corner interpolation patch in order to smoothly interpolate colour over a planar domain partitioned into a polygonal mesh. More precisely, we aim for a global colour surface with at least G^1 continuity, but the individual (colour) patches should depend only on local data, on a per-patch basis, so that the patches can be processed in the modern graphics pipeline [14] easily and efficiently. As with traditional quadrilateral gradient meshes, the colour component of the gradient handles(s) defined at a vertex inherit(s) the colour value of that vertex. In this way, colour is smoothly interpolated along the cubic boundary curves and within patches.

The actual data, typically all expressed via control points, for the two patch types are exactly the same. A Bézier ribbon provides the tangent functions along the curves for the Charrot-Gregory patch, as well as the control net for the Gregory generalised Bézier patch. We construct the ribbons fully locally, a patch at a time, from the (user) provided data using the method of Chiyokura and Kimura.

3.1 The Method of Chiyokura and Kimura for Colour Interpolation

The method of Chiyokura and Kimura [7] joins together two Bézier patches with tangent plane continuity by only considering shared positional data. The original approach is used for patches in $\mathbb{R}^3$, whereas in our setting we work in a special space with two positional and three colour dimensions. However, we can tailor the method to our needs.

Consider two adjacent quadrilateral gradient mesh patches $\Phi_a(u,v)$ and $\Phi_b(u,v)$ with a shared cubic Bézier boundary curve $\Gamma(u)$, $u\in[0,1]$, with cross-boundary derivatives $\partial\Gamma_a(u)$ and $\partial\Gamma_b(u)$ along $\Gamma(u)$, respectively. Similar to the control points of the patches, the derivatives also live in $\mathbb{R}^{2+3}$.

The task now is to satisfy the following sufficient condition for G^1 continuity along $\Gamma(u)$:

$$\det\left(\partial\Gamma(u), \partial\Gamma_a(u), \partial\Gamma_b(u)\right) = 0, \quad u\in[0,1].$$

This condition can be expressed in terms of suitably constructed scalar-valued functions $\alpha(u)$, $\beta(u)$ and $\gamma(u)$ such that:

$$\alpha(u)\partial\Gamma(u) + \beta(u)\partial\Gamma_a(u) + \gamma(u)\partial\Gamma_b(u) = 0. \tag{4}$$

In the original method, an auxiliary patch Φ_b is used, and called the basis patch. This patch is constructed from the common boundary data so that a G^1 connection across $\Gamma(u)$ can be determined fully locally; see Figure 4. In the original 3D setting, the basis patch is constructed with the help of normal vectors at the endpoints of $\Gamma(u)$. However, in the colour interpolation setting, the 'normal' vector should be determined in $\mathbb{R}^{2+3}$. In principle, it is possible to have non-zero colour gradients at vertices by estimating the local colour gradient from the colours of adjacent vertices. This is particularly easy for traditional gradient meshes as the gradient components can be individually estimated for both parametric directions using monotonic cubic interpolation as suggested in [18, 25]. For extraordinary vertices this step becomes more involved and it is unclear how to obtain gradients which

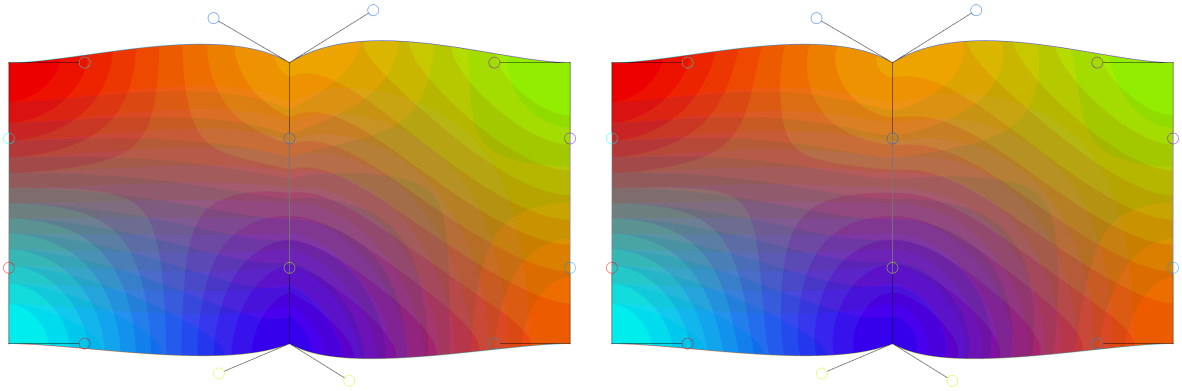


Figure 5: *A simple gradient mesh composed of two quads. Left: A traditional gradient mesh (based on Ferguson/Bézier patches) results in non-smooth transitions at the shared edge, which is due to the three (linearly) independent gradient handles at the shared vertices. Right: In contrast, the Gregory gradient mesh produces a smooth image over the entire mesh.*

preserve monotonicity in the G^1 setting. We thus set the colour components of the gradient vectors at the vertices of the polygon (and the whole mesh) to zero. Consequently, the basis patch vectors $\mathbf{b}_0$ and $\mathbf{b}_3$ can be determined simply by rotating the vectors $\mathbf{c}_0$ and $\mathbf{c}_2$ by ninety degrees in the image plane. We can then express the given vectors of the actual patch Φ_a as $\mathbf{a}_0 = k_0\mathbf{b}_0 + h_0\mathbf{c}_0$ and $\mathbf{a}_3 = k_1\mathbf{b}_3 + h_1\mathbf{c}_2$, which determines the values of k_0, k_1 and h_0, h_1.

We then proceed by completing the definition of the basis patch (or rather its ribbon at $\Gamma(u)$ only), by assuming that $\mathbf{b}_1$ and $\mathbf{b}_2$ can be determined by linear interpolation $\mathbf{b}_1 = \frac{2}{3}\mathbf{b}_0 + \frac{1}{3}\mathbf{b}_3$, $\mathbf{b}_2 = \frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_3$. Finally, the missing ribbon vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ of the actual patch Φ_a are computed as

$$\mathbf{a}_1 = (k_1 - k_0)\frac{\mathbf{b}_0}{3} + k_0\mathbf{b}_1 + 2h_0\frac{\mathbf{c}_1}{3} + h_1\frac{\mathbf{c}_0}{3}, \quad \mathbf{a}_2 = k_1\mathbf{b}_2 - (k_1 - k_0)\frac{\mathbf{b}_3}{3} + h_0\frac{\mathbf{c}_2}{3} + 2h_1\frac{\mathbf{c}_1}{3}.$$

It is hardly surprising that 2D patches can be connected with tangent plane continuity at their shared edge, but in our setting of vector graphics, the important part is the colour component, especially that of the internal control points of Φ_a (white squares in Figure 4). Since the colour component of the gradients at vertices is zero, the method simplifies in this regard. All the colour components of vectors $\mathbf{c}_0, \mathbf{c}_2$, and $\mathbf{a}_0, \mathbf{a}_3$ are zero. Therefore the only vector that weighs in the calculation of the colour component of the internal control points is $\mathbf{c}_1$, the middle control polygon leg of $\Gamma(u)$. However, $\mathbf{c}_1$ is simply the difference between the colours at the end points of $\Gamma(u)$. This leads to

$$\mathbf{a}_1^c = 2h_0\frac{\mathbf{c}_1^c}{3}, \quad \mathbf{a}_2^c = 2h_1\frac{\mathbf{c}_1^c}{3}$$

in colour space (signified by the superscript c), which enables us to smoothly interpolate colour over polygonal gradient meshes.

3.2 Quadrilateral Gregory Gradient Patches

The use of the method of Chiyokura and Kimura for G^1 colour connection as described above paves the way for the use of quadrilateral Gregory patches [7, Section 5.1] as a gradient mesh primitive. Using this primitive instead of Ferguson patches or tensor-product Bézier patches has advantages. First, it is possible to guarantee smooth colour transitions between patches with arbitrary gradients, meaning that the co-linearity property of gradient handles in the traditional setting is not necessary. Secondly, it allows for an arbitrary number of patches to meet at a vertex while still guaranteeing smoothness of the resulting colour surface. This improves the versatility of the gradient mesh primitive as it is no longer limited to strictly regular topology.

In Figure 5, we compare the traditional gradient mesh with the Gregory gradient mesh on a simple example to illustrate their differences. The mesh consists out of two adjacent quadrilateral patches. The gradient vectors have been set in such a way that the resulting curve network is only C^0. In this case, it is impossible for the traditional gradient mesh to yield a G^1 colour surface. The Gregory gradient mesh is able to create a G^1 colour surface even though the gradient vectors at the endpoints of the common boundary curve are not co-linear, but they are still co-planar in the $\mathbb{R}^{2+3}$ space. The use of colour banding clearly reveals that the Gregory gradient mesh produces a smooth colour surface, whereas the traditional gradient mesh shows sharp corners in the bands, clearly indicating only C^0 continuity of the colour surface at the shared edge.

Of course, quadrilateral patches still have only limited abilities as they cannot easily represent triangular or pentagonal regions, at least not without introducing overlapping control points and/or forcing the Jacobian of the 2D geometry to vanish. This is exactly where the colour interpolation versions of the (Gregory) generalised Bézier patch and the Gregory corner interpolation patch come in.

3.3 Multisided Colour Interpolants

To increase the usefulness and broaden the range of shapes that a single multisided polygon can model, we would like to support both convex and non-convex polygons. To this end, we make use of mean value coordinates [11] to parametrise the domain of the multisided patches, both for the generalised Bézier patches and the corner interpolator patch. As the geometry domain of the gradient meshes is two dimensional in our setting, it is possible to use the actual polygon as its own domain for parametrisation. In the usual 3D setting, a regular domain is often used, or a best fitting planar domain is computed on a per-polygon basis [22].

For the generalised Bézier patches this simply means that we employ mean value coordinates; see Section 2.4. It is also possible to use harmonic coordinates [16, 22], but harmonic coordinates, unlike mean value coordinates, do not admit closed-form evaluation and have to be computed iteratively via rasterising or triangulating the polygon first. This significantly complicates the process of delivering scale-invariant evaluations of the primitives, especially as a higher resolution parametrisation might have to be computed on the fly. The use of mean value coordinates has the consequence that in the event that a polygon is concave, negative values of ϕ_i might occur. This affects the interpolation of colour as the colour component can traverse out of the gamut, as already mentioned in Section 2.3. We simply clamp the value of the negative coordinates to 0 in the calculation of the local

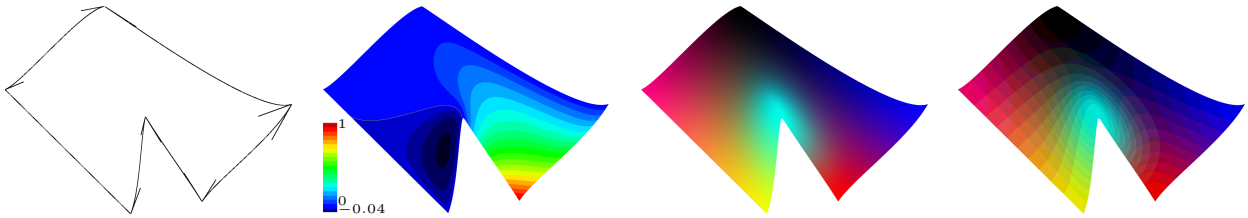


Figure 6: *Left to right: Input gradient mesh (non-convex polygon), mean value barycentric coordinate function of the bottom right vertex visualised on-top of a GB patch, and the resulting colour interpolation of the GB patch with non-convex adjustment, and also with colour banding. Negative values in the coordinate visualisation have been scaled to increase their visibility, and the zero-contour is highlighted in grey.*

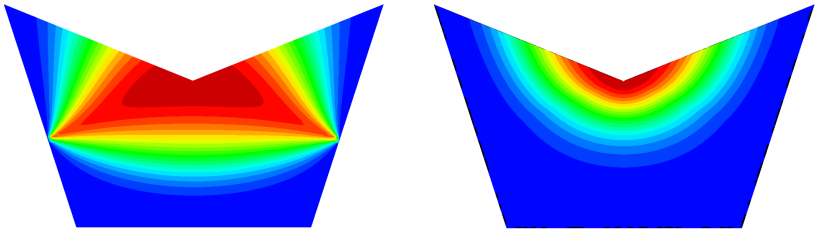


Figure 7: *Left: The original parametrisation of the Charrot-Gregory patch that uses perpendicular distances to edges applied to a non-convex polygon. Right: Our technique that uses 'distance' functions based on mean value coordinates provides a better parametrisation.*

parameters s_i and h_i for each ribbon. This leads to only C^0 continuity at the zero contour of the offending coordinate function, but this is hardly visible as the negative values are typically close to zero. In Figure 6 the interpolation of colour on a concave patch is evaluated. Even though some of the coordinates are negative in some parts of the polygon, the resulting colour surface still appears as smooth as in the case of convex polygons.

For convex polygons, mean value coordinates in combination with generalised Bézier patches yield adequate results. However, when the gradients specified at a vertex make the vertex not strictly convex, i.e., the outward normal components of the gradients lie in the same half-circle, fold-overs will occur. Salvi and Várady [22] describe a method that can alleviate the problem. The gradients at the end-points of the individual ribbons can be flipped in direction such that the tangent vector of the ribbon will always point inward, into the polygon. The same solution can be applied to the method of Chiyokura and Kimura by finding a common direction for the end point vectors $\mathbf{a}_0$ and $\mathbf{a}_3$. We found we can do this locally, by flipping the tangent vectors as follows. Consider the tangent vectors $\mathbf{t}_{i,i-1}$ and $\mathbf{t}_{i,i+1}$ of some vertex $\mathbf{v}_i$. We can compare the outward normal component $\mathbf{n}_{i,i-1}$ of the tangent vector $\mathbf{t}_{i,i-1}$ with $\mathbf{t}_{i,i+1}$ to determine the sign of $\mathbf{t}_{i,i-1}$ to be used in the method of Chiyokura and Kimura by checking whether $\mathbf{n}_{i,i-1} \cdot \mathbf{t}_{i,i+1} > 0$. We also employ the suggestion of Salvi and Várady [22] to adjust the blending functions μ^i_{jk} of (1) to use squared terms to increase the smoothness of the surface.

For the corner interpolator patch this becomes a more involved step, as the original patch is based on orthogonal distances to edges of a domain polygon and the local parameters are defined by means of the radial sweep-line construction [6]. To be able to use the corner interpolator patch, we need to make these parametrisations compatible with non-convex domains. The weighting function and the local parameters (Section 2.5) introduce singularities into the construction at concave corners.

Alternative formulations have been suggested by Várady et al. [26], however these are computationally involved. We propose to use generalised barycentric coordinates to construct both the weights for the corner interpolators as well as the local parameters.

To this end, we make use of the same functions that compromise the local parameters for the generalised Bézier patch. In particular, we are interested in the parameter $h_i = 1 - \phi_i - \phi_{i-1}$, where the ϕ_i are mean value coordinates, replacing the edge-distance functions used in the original construction. The functions are zero on the edge $\mathbf{v}_i\mathbf{v}_{i-1}$, and increase linearly on edges $\mathbf{v}_{i-1}\mathbf{v}_i$ and $\mathbf{v}_{i+1}\mathbf{v}_{i+2}$, as desired.

We replace the d_i functions in (3) with the functions h_i. For the local parameters u_i and v_i, we substitute the functions s_i and h_i, respectively. This new weight function and local parameters have the same properties as the original functions, i.e., G^1 continuity is still maintained across adjacent patches, but are now suitable for non-convex domains, as illustrated in Figure 7.

Unfortunately, the same adjustment to the tangent vectors at non-convex corners as employed in the generalised Bézier case does not help in the case of the corner interpolator patch. This is because the necessary condition (2) on the tangent fields along the edge curves is no longer satisfied. This somewhat limits the usability of the corner interpolator gradient mesh as it can only support strictly convex tangent vectors, meaning that the internal angle of the tangent vectors must be below 180 degrees. Still, the polygonal domain itself can be non-convex as Figure 7 and Figure 11 (bottom row, second column) show. A full investigation into the injectivity of the resulting planar patches is beyond the scope of this paper.

4 Results and Discussion

We compare and evaluate the multisided colour interpolants in several ways. This includes suitability of the patches for colour interpolation over convex and concave patches, and comparisons in terms of visual fidelity of the resulting colour surfaces, computational/rendering performance, and editability from the user perspective.

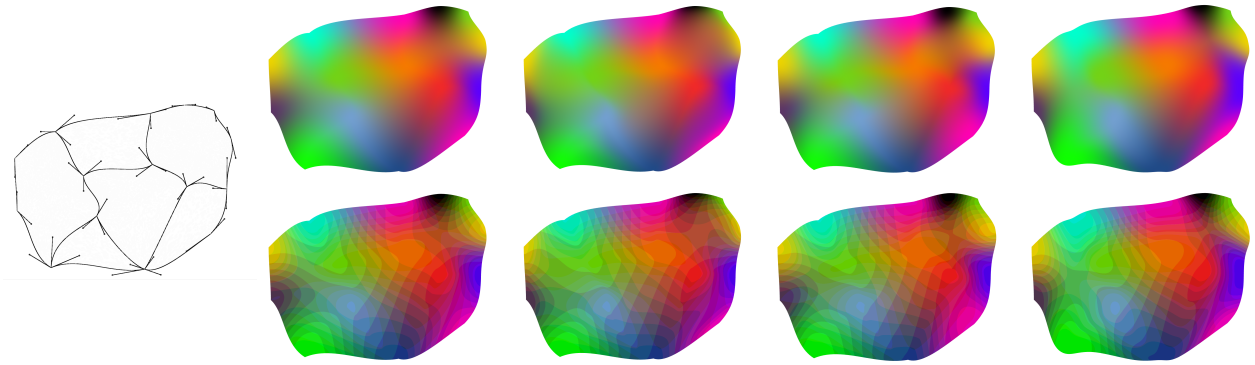


Figure 8: *A polygonal gradient mesh (far left) consisting of convex faces of various valencies. Top row: Colour interpolation. Bottom row: Iso-bands of the individual channels of the interpolants. From left to right: Gregory gradient mesh, Charrot-Gregory gradient mesh, cubic mean value coordinate gradient mesh, and subdivision-based topologically unrestricted gradient mesh.*

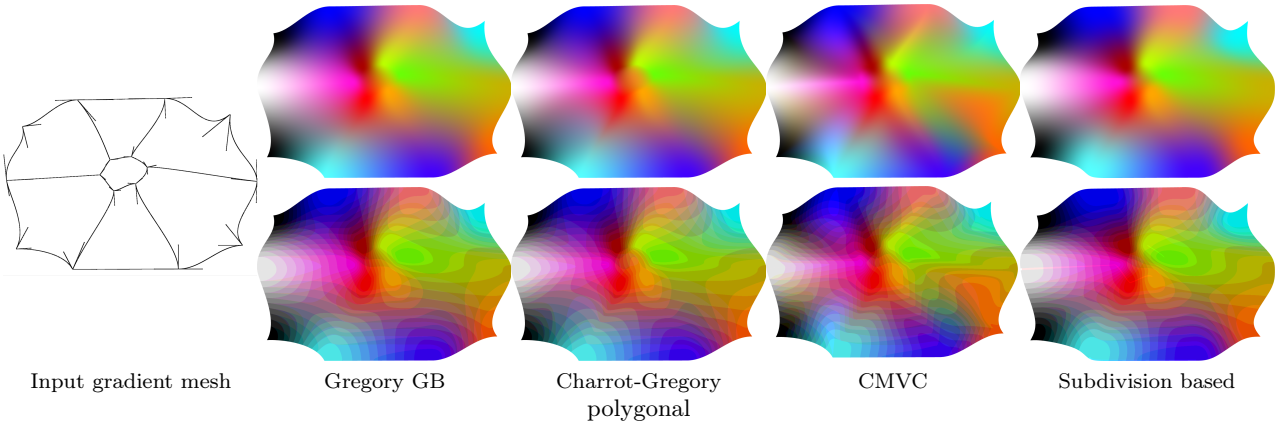


Figure 9: *A polygonal gradient mesh (far left) consisting of convex faces of various valencies, but with one relatively smaller hexagon (central polygon). Top row: Colour interpolation. Bottom row: Iso-bands of the individual channels of the interpolants. From left to right: Gregory gradient mesh, Charrot-Gregory gradient mesh, cubic mean value coordinate gradient mesh, and subdivision-based topologically unrestricted gradient mesh.*

4.1 Colour Interpolation on Convex Patches

In Figure 8 we show the results for all the four considered techniques on a simple gradient mesh consisting of faces of different valencies and with many extraordinary vertices. This mesh cannot be modelled with an ordinary gradient mesh.

Observe that the Gregory generalised Bézier and Charrot-Gregory colour interpolation techniques give similar visual results. Even though the methods create only a G^1 colour surface, the visual quality of the interpolation is very high. This is especially apparent in the colour iso-band visualisation, showing comparable smoothness to that of the iso-bands of the subdivision based technique.

The cubic mean value coordinate interpolant shows the sharpest turns in the iso-band edges. Also, as the gradient constraints at the vertices are not necessarily C^1 compatible, it introduces visual artefacts there. Here, the continuity is only C^0 as we made the patches interpolate all the user-specified gradient handles in order to obtain a fair comparison. Nevertheless, along the edges we still observe smooth transitions in colour. It is possible to fix the C^1 discontinuity at extraordinary vertices as mentioned in Section 2.3. This would change the gradient handles originating from this vertex and thus diminish the freedom a designer has in specifying these. This trade-off is not present in any of the other considered methods.

If we assess the visual quality of the interpolation methods by the inability to see the underlying mesh structure (a criterion typically employed also in the 3D geometry setting), then it is clear that the topologically unrestricted gradient mesh performs this task in the best manner. This is due to the higher smoothness of the colour surface of this technique as it is C^2 almost everywhere.

Introducing polygons into the gradient mesh which are of a smaller size compared to adjacent polygons should affect colour interpolation in an intuitive way. One such situation is created in the gradient mesh featured in Figure 9, where a smaller hexagonal polygon is surrounded by larger polygons. In this case both the Gregory GB patch and the subdivision-based gradient meshes perform well, as they do not reveal the underlying mesh. The Gregory-Charrot mesh also performs well although the colour interpolation around the vertices of the central hexagon show sharper transitions in colour as evidenced by the sharper corners in the colour iso-lines. The cubic mean value interpolant shows the most diverging behaviour as it introduces unexpected colours with non-intuitive propagation in the interior of the polygons as well as sections which reveal the underlying mesh.

4.2 Colour Interpolation on Concave Patches

When designing a polygonal gradient mesh, it is not unthinkable that concave faces might occur inside the mesh. Automatic image vectorisation or gradient mesh simplification is also likely to create concave faces [19]. In Figure 10 we assess the abilities of the different interpolants with respect to concave faces.

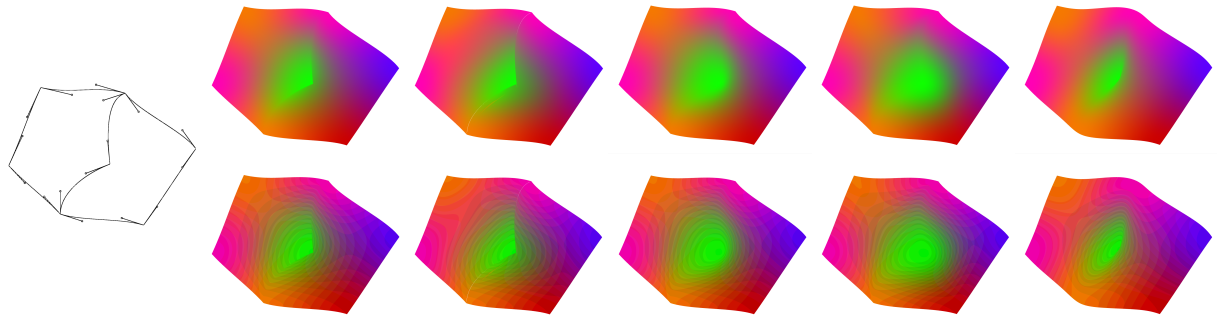


Figure 10: *A simple gradient mesh (far left) consisting out of two pentagonal faces, one of which is concave. Top row: Colour interpolation. Bottom row: Iso-bands of the individual channels of the interpolated colours. From left to right: Gregory gradient mesh, Charrot-Gregory gradient mesh, Gregory gradient mesh with non-convexity adjustments, cubic mean value coordinate gradient mesh, and subdivision-based topologically unrestricted gradient mesh.*

As expected, in the case of concave corners most of the considered techniques fail to deliver a smooth colour surface as the geometry folds over. This is the case for the Gregory GB gradient mesh, the Charrot-Gregory gradient mesh, and the topologically unrestricted gradient mesh. The last still shows the most acceptable colour surface among these three methods as the folding over is less severe. However, increasing the concave angle also increases the amount of folding introduced. Both the Gregory GB gradient mesh with the non-convexity adjustments and the cubic mean value coordinate gradient mesh perform well in this setting. The iso-bands of the colour surface remain smooth and no fold-overs are introduced around the offending vertex, and the cubic mean value coordinate gradient mesh creates slightly smoother results in this case. This shows that these two gradient mesh types have comparable abilities when it comes to colour interpolation on concave faces.

It is important to have consistency in colour when manipulating a gradient mesh through translation of vertices or gradient handles. In Figure 11 we show an example of a gradient mesh consisting out of a single hexagonal patch. In the example, a single vertex is dragged so that the initially convex patch becomes concave. For the subdivision-based gradient mesh we can see that the colours around all vertices remain constant and that the colour region around the translated vertex gets 'thinner'. For the Charrot-Gregory patch and the Gregory GB patch the colours remain constant even when the polygon becomes concave. Somewhat strange results are obtained with the cubic mean value interpolant: different colours seem to appear and disappear in the interior of the patch as the vertex is moved. This effect increases in magnitude when some edges become relatively short due to the translation of a vertex, even if the gradients are shortened accordingly; cf. Figure 9.

4.3 Performance

The performance of the different interpolants is an interesting criterion to evaluate the different techniques on, especially in regard to their potential applications in vector graphics authoring tools. It is important for interactivity that the gradient mesh can render instantly so that the user has immediate feedback as they edit the gradient mesh through dragging vertices and tangent handles or adjusting vertex colours.

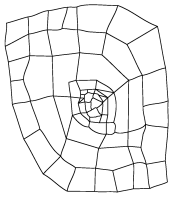
We evaluate the performance of the different techniques based on the steps involved in rendering the representations. This can be divided in the time it takes to create the mesh structure, and the time it takes to render the actual mesh on the screen. The gradient mesh structure is contained in a half-edge mesh structure that stores all positional data, colours and gradient handles. From this structure the different representations of (polygonal) gradient meshes are generated.

For the Gregory GB patch and the Gregory-Charrot corner interpolator patch we can directly transfer the data contained in the mesh structure into patches of their respective mesh structures. We then use the multisided tessellation method presented in [14] to render the patches efficiently on the GPU. For 'highly' concave faces, this triangulation strategy of the domain polygon does not work as some of the triangles may cover areas outside the domain polygon. In this case we resort to evaluating these patches on the CPU side.

The cubic mean value coordinate mesh is again rendered on the CPU side as we used the publicly available implementation supplied by the authors of [19]. Still, the patches can be constructed fully locally and could potentially be rendered using the same tessellation method mentioned above. However, we did not attempt to port the provided code to shader code. Instead, we used OpenMP parallelisation to increase the performance to exploit the locality of the method.

Table 1: *Performance evaluation of rendering a gradient mesh with 65 faces of various valencies (far right). Build time refers to the time it takes to create and gather the data to push it to GPU buffers, whereas render time refers to the time it takes to render the data on screen (using OpenGL). All the gradient meshes were rendered at approximately equal triangulation density at 1080p resolution.*

Method	Build time	Render time	Triangle count
Gregory GB gradient mesh	1 ms	0.52 ms	343541
Gregory GB gradient mesh (max. tess.)	1 ms	1.09 ms	1148928
Charrot-Gregory gradient mesh	1 ms	1.20 ms	343541
Charrot-Gregory gradient mesh (max. tess.)	1 ms	2.41 ms	1148928
Cubic mean value coordinates gradient mesh	491 ms	1.05 ms	281600
Topologically unrestricted gradient mesh	590 ms	1.20 ms	330624



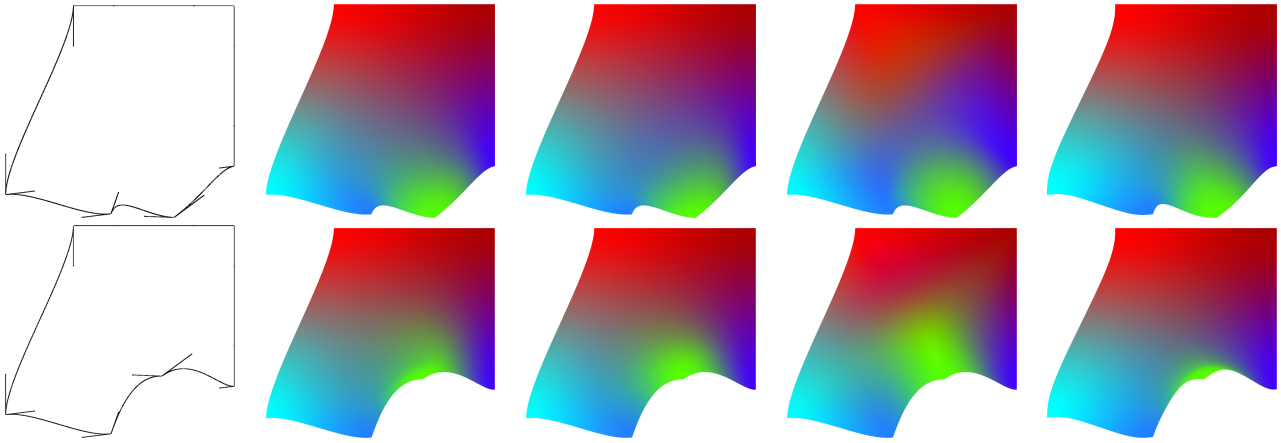


Figure 11: *A single hexagonal patch and the effect of translating one of the vertices so that the patch transitions from a convex shape (top row) to a concave shape (bottom row).*

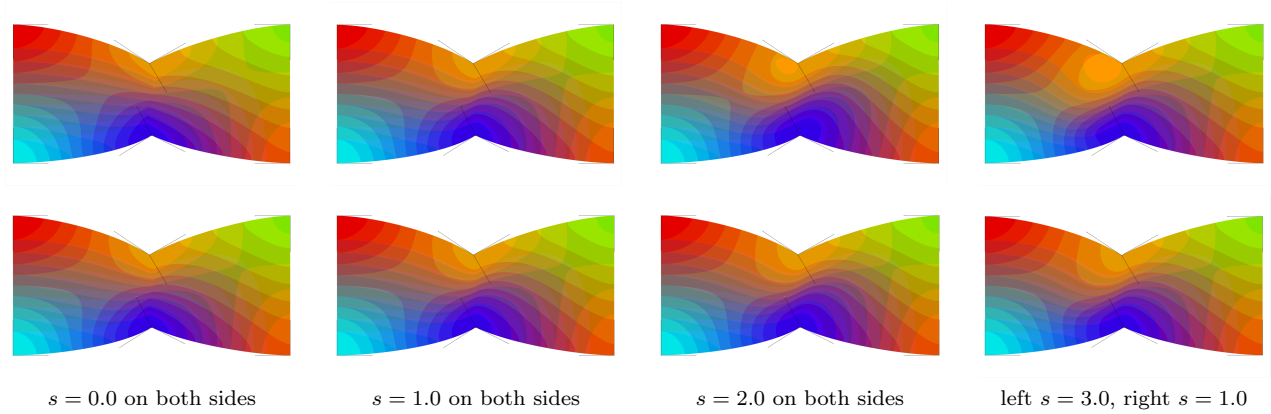


Figure 12: *The same Gregory gradient meshes with different values for the parameter s for the common edge of the two quadrilateral faces rendered with showing the iso-bands of the individual colour channels. Top row: Our scaling method. Bottom row: The scaling approach of [24].*

The subdivision-based topologically unrestricted gradient mesh requires the most involved computations as the Catmull-Clark subdivision operator is a global refinement step. On top of this the mesh must first be pre-processed using the modified ternary subdivision step discussed in Section 2.2. For every adjustment the user makes to the underlying mesh structure, the full mesh has to be subdivided again from the ground up, including the ternary subdivision step. For high subdivision levels this might be detrimental to interactivity, however, due to the sparseness of most gradient meshes this need not be an issue [20] and a GPU-based implementation is in principle possible.

We conducted a test to check the viability of all the techniques in the event of interactively editing a mesh. In this context, interactively editing the mesh is understood as dragging vertices or tangent handles to change the geometry of the gradient mesh, or modifying colours. This is a use-case any designer would go through many times when designing a gradient mesh. We conducted the performance comparison on a computer with two Intel Xeon E5-2630 processors (@2.3 GHz) and an NVIDIA Titan V graphics card with 12 GB of video memory. Our results are reported in Table 1.

We can see that the two newly proposed gradient meshes are very efficient, both in rendering and in the time it takes to transfer their data to the GPU. We rendered them at two different densities, one such that the level of detail is comparable to the other methods and once using the maximum tessellation rate of 64×64 (current hardware limit). The performance of the Gregory gradient mesh is approximately twice as fast as that of the Charrot-Gregory gradient mesh. It shows the viability of using these new primitives in vector graphics authoring software, as they greatly improve the performance with respect to the other methods. With the new primitives, gradient meshes can be modified in real time, giving the user instant feedback at every step during design. The performance could be increased even further by employing adaptive tessellation strategies [23].

4.4 Editability

All the discussed techniques have similar abilities when it comes to smooth colour interpolation, with the exception of concave vertices, as shown above. Although clearly important, smooth interpolation is often not enough to accommodate designer needs as sharp colour transitions are typically needed to increase the visibility of features. While it is possible to increase the density of the mesh areas where a fast transition in colour is needed, this also increases the complexity of the mesh, whereas we are after as sparse as possible a representation.

The use of semi-sharp creases [9] was already mentioned as a means to introduce features in [20], as well as to create hard colour transitions in the mesh. We determined a simple way to offer a similar ability in the Gregory gradient mesh, namely by scaling the tangent vectors of the individual Bézier ribbons. As previously shown by Salvi [22], there is no apparent restriction on the orientation of the tangent vectors used in the ribbons with respect to their adjacent edge curves. This same principle

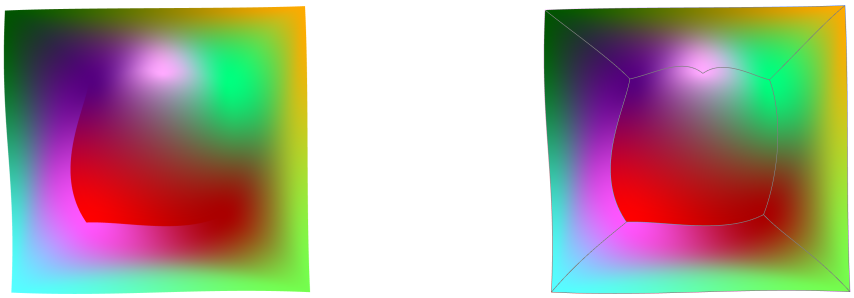


Figure 13: *A simple gradient mesh with 2 pentagons and 3 quads showing sharp colour transitions rendered using Gregory GB gradient meshes with the non-convexity adjustment. Notice how the sharp colour transitions blend back into smooth colour transitions along the (hard) edges.*

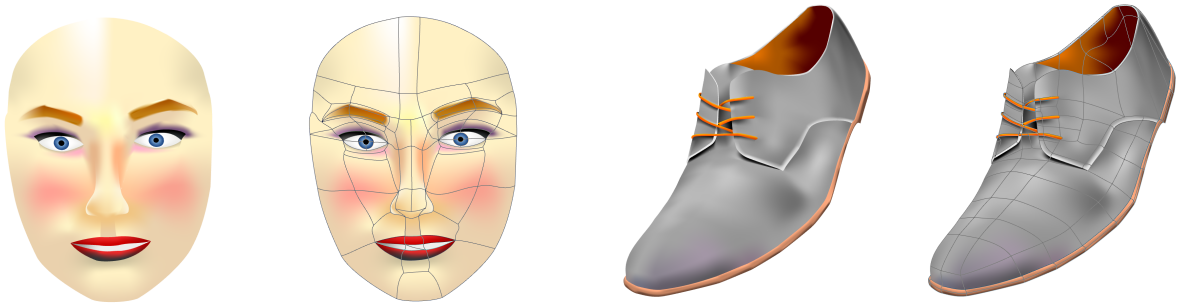


Figure 14: *Two meshes rendered using Gregory GB gradient meshes with the non-convexity adjustment (face model: 101 faces in total, 12 triangles, 61 quads, 24 pentagons and 4 hexagons; shoe model: 166 faces in total, 20 triangles, 115 quads, 29 pentagons and 2 hexagons). Both the colour interpolants and the curve networks are shown. Note that apart from smooth colour interpolation also sharp transitions have been modelled.*

applies to the magnitude of the tangent vectors. Therefore, we can scale the vectors $\mathbf{a}_0$ and $\mathbf{a}_3$ used in determining the inner control points of the Bézier ribbon by an arbitrary constant factor $s > 0$ to increase or decrease the influence a ribbon has on local colour propagation into the patch. This is similar to adjustments found for the two middle vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ of the basis patch by Shirman & Sequin [24] for quadrilateral Gregory patches. The difference for the Gregory GB patches is that blending functions are also used to blend control point positions for the edge curves, whereas quadrilateral Gregory patches only blend internal control points.

The effect of the parameter s is shown in Figure 12 for two connected quadrilateral Gregory gradient mesh primitives. Note that manipulating the magnitude in the case of only using the centre vectors of the basis patch changes the extent in the centre of the ribbon, whereas our adjustment changes it for a wider region of the ribbon. The effect is still quite subtle, but it is possible to see that we can get sharper transitions for $s = 0$ as our method also scales the endpoint vectors of the basis patch. Increasing the parameter value increases the extent and flatness of the colour transition over the boundary. Increasing the extent too much for adjacent edges then leads to fold-overs.

We can also model hard transitions simply by allowing vertices to have multiple colours, one per incident face [2, 28]. Figure 13 shows a simple polygonal gradient mesh where a sharp transition is created for one of the middle vertices of the mesh. It clearly shows a sharp transition in colour along the user-specified hard edges emanating from this vertex. The sharp colour transition blends back into a smooth colour transition at the end of the sharp edges.

Topological editing of the mesh structure is supported for all mesh types. Adding, removing, and arbitrarily splitting faces and edges can be achieved without loss of smoothness for all the techniques, with the possible exception of cubic mean value coordinates meshes due to the gradient incompatibility issue. The hierarchical editing of topologically unrestricted gradient meshes [28] is not easily matched by any one of the other techniques.

Nevertheless, it is possible to create a transition of high detail areas to low detail areas with non-hierarchical methods. For instance, this can be achieved by turning a quadrilateral face into a pentagonal face by subdividing one of its edges. One can see many such situations arising in the realistic gradient meshes in Figure 14 rendered using Gregory GB gradient meshes. Notice the complicated topology introduced around the eyes of the 'face' mesh and around the shoelaces of the 'shoe' mesh.

5 Conclusion

We have modified the Gregory GB patch and the Gregory-Charrot corner interpolator patch to make them suitable for colour interpolation and to serve as the underlying primitive for a generalised gradient mesh. The patches can interpolate colour just as the traditional Ferguson patch-based gradient mesh, but now also for arbitrary manifold topology meshes. The patches can be constructed fully locally through the use of the method of Chiyokura and Kimura, which has been adjusted for G^1 colour interpolation. In addition, the use of mean value coordinates increases the usability of the new primitives by supporting also concave faces. The Gregory GB gradient mesh has an advantage as it allows the direction in its individual ribbons to differ and thus avoid fold-overs at concave corners.

We have compared the patches to the existing cubic mean value interpolant and the topologically unrestricted gradient mesh. We found that in terms of visual quality they are comparable to the subdivision based topologically unrestricted gradient mesh even though the continuity of the colour surface is only G^1. The new techniques are also very efficient as they can be directly

rendered using hardware tessellation methods, and are much more efficient than the existing two methods as they require only minimal mesh data. In the case of simple geometric editing of an existing gradient mesh, they behave optimally for interactivity. The new polygonal gradient mesh primitives are a good alternative to the two existing techniques, and increase the viability of the use of arbitrary manifold topology gradient meshes in existing and future vector graphics authoring software.

Acknowledgements

Some of the material presented in Section 3 is based on the second author's BSc thesis at the University of Groningen. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

References

[1] Mesh gradients - Inkscape. http://wiki.inkscape.org/wiki/index.php/Mesh_Gradients. Accessed: 2019-03-29.

[2] P. J. Barendrecht, M. Luinstra, J. Hogervorst, and J. Kosinka. Locally refinable gradient meshes supporting branching and sharp colour transitions. *The Visual Computer*, 34(6-8):949–960, 2018.

[3] P. Barla and A. Bousseau. Gradient Art: Creation and Vectorization. In P. R. John Colomosse, editor, *Image and Video based Artistic Stylization*. Springer, Nov. 2012.

[4] R. Beatson, M. S. Floater, and C. E. Køashagen. Hermite mean value interpolation on polygons. *Computer Aided Geometric Design*, 60:18–27, 2018.

[5] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355, 1978.

[6] P. Charrot and J. A. Gregory. A pentagonal surface patch for computer aided geometric design. *Computer Aided Geometric Design*, 1(1):87–94, 1984.

[7] H. Chiyokura and F. Kimura. Design of solids with free-form surfaces. *ACM SIGGRAPH Computer Graphics*, 17(3):289–298, 1983.

[8] F. D. Coburn and P. McCormick. *CorelDRAW 9: The Official Guide*. Osborne McGraw-Hill, 2017.

[9] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, 1998.

[10] J. Ferguson. Multivariable curve interpolation. *Journal of the ACM (JACM)*, 11(2):221–228, 1964.

[11] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[12] M. S. Floater. Generalized barycentric coordinates and applications. *Acta Numerica*, 24:161–214, 2015.

[13] M. S. Floater and C. Schulz. Pointwise radial minimization: Hermite interpolation on arbitrary domains. *Computer Graphics Forum*, 27(5):1505–1512, 2008.

[14] G. J. Hettinga, P. J. Barendrecht, and J. Kosinka. A Comparison of GPU Tessellation Strategies for Multisided Patches. In O. Diamanti and A. Vaxman, editors, *EG 2018 - Short Papers*. The Eurographics Association, 2018.

[15] G. J. Hettinga and J. Kosinka. Multisided generalisations of Gregory patches. *Computer Aided Geometric Design*, 62:166–180, 2018.

[16] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26(3), July 2007.

[17] J. Kosinka, M. A. Sabin, and N. A. Dodgson. Control vectors for splines. *Computer-Aided Design*, 58:173–178, 2015. Solid and Physical Modeling 2014.

[18] Y.-K. Lai, S.-M. Hu, and R. R. Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. In *ACM Transactions on Graphics (TOG)*, volume 28, pages 85:1–85:8. ACM, 2009.

[19] X.-Y. Li, T. Ju, and S.-M. Hu. Cubic mean value coordinates. *ACM Trans. Graph.*, 32(4):126:1–126:10, July 2013.

[20] H. Lieng, J. Kosinka, J. Shen, and N. A. Dodgson. A colour interpolation scheme for topologically unrestricted gradient meshes. In *Computer Graphics Forum*, volume 36, pages 112–121. Wiley Online Library, 2017.

[21] Y. Lipman, J. Kopf, D. Cohen-Or, and D. Levin. GPU-assisted Positive Mean Value Coordinates for Mesh Deformations. In A. Belyaev and M. Garland, editors, *Geometry Processing*. The Eurographics Association, 2007.

[22] P. Salvi and T. Várady. Multi-sided Bézier surfaces over concave polygonal domains. *Computers & Graphics*, 74:56–65, 2018.

[23] H. Schäfer, M. Nießner, B. Keinert, M. Stamminger, and C. Loop. State of the Art Report on Real-time Rendering with Hardware Tessellation. In S. Lefebvre and M. Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

[24] L. A. Shirman and C. H. Séquin. Local surface interpolation with shape parameters between adjoining Gregory patches. *Computer Aided Geometric Design*, 7(5):375–388, 1990.

[25] J. Sun, L. Liang, F. Wen, and H.-Y. Shum. Image vectorization using optimized gradient meshes. In *ACM Transactions on Graphics (TOG)*, volume 26, pages 11:1–11:7. ACM, 2007.

[26] T. Várady, A. Rockwood, and P. Salvi. Transfinite surface interpolation over irregular n-sided domains. *Computer-Aided Design*, 43(11):1330–1340, 2011. Solid and Physical Modeling 2011.

[27] T. Várady, P. Salvi, and G. Karikó. A multi-sided Bézier patch with a simple control structure. In *Computer Graphics Forum*, volume 35, pages 307–317. Wiley Online Library, 2016.

[28] T. W. Verstraaten and J. Kosinka. Local and hierarchical refinement for subdivision gradient meshes. In *Computer Graphics Forum*, volume 37, pages 373–383. Wiley Online Library, 2018.

[29] E. Wachspress. *A Rational Finite Element Basis*. Academic Press, 1975.

[30] B. Wood. *Adobe Illustrator CC Classroom in a Book*. Adobe Press, 2017.