Volume xx (200y), Number z, pp. 1–11

A colour interpolation scheme for topologically unrestricted gradient meshes

Henrik Lieng[1,2], Jiří Kosinka[1,3], Jingjing Shen[1], Neil A. Dodgson[1]

1. University of Cambridge, UK
2. Oslo and Akershus University College of Applied Sciences, Norway
3. Johann Bernoulli Institute, University of Groningen, The Netherlands

Abstract

Gradient meshes are a 2D vector graphics primitive where colour is interpolated between mesh vertices. The current implementations of gradient meshes are restricted to rectangular mesh topology. Our new interpolation method relaxes this restriction by supporting arbitrary manifold topology of the input gradient mesh. Our method is based on the Catmull-Clark subdivision scheme, which is well known to support arbitrary mesh topology in 3D. We adapt this scheme to support gradient mesh colour interpolation, adding extensions to handle interpolation of colours of the control points, interpolation only inside the given colour space, and emulation of gradient constraints seen in related closed-form solutions. These extensions make subdivision a viable option for interpolating arbitrary-topology gradient meshes for 2D vector graphics.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Generation, Graphics Utilities

1. Introduction

Vector graphics provides powerful tools for drawing vivid, scalable, 2D imagery. In commercial products, Adobe Illustrator's *gradient mesh* tool (and the equivalent *mesh fill* tool in Corel's CorelDRAW) is one of the more powerful tools. It allows artists to draw colour gradients in the interior of vector objects via control meshes. The underlying vector primitive associated with the tool is defined by a rectangular grid, where colours and colour gradients are associated with the mesh control points. Both colour and colour gradients can be manipulated by the user.

The gradient mesh tool has been employed by both artists and researchers. In art, it allows highly skilled, experienced artists to produce photorealistic imagery using vector graphics. Such imagery is unfeasible to achieve with the other tools available in a vector graphic product. However, producing such high-quality vector graphics in this way is widely regarded as complicated and tedious, while mastering the tool is considered rewarding since complex imagery can be accomplished. In research, the gradient mesh tool has inspired researchers to propose solutions to challenging colour interpolation problems. In particular, multiple solutions to the problem of vectorising an input photograph to a vector-based representation have been proposed using gradient meshes (e.g. [SLWS07, LHM09]).

The technical solution, used by Adobe and Corel, to interpolate the colours and colour gradients has not been published. As with other approaches in the literature concerned with gradient meshes (e.g. [SLWS07]), we assume that Ferguson patches [Fer64] are used for interpolating the interior pixels. Ferguson's framework provides bi-cubic interpolatory C^1 patches where gradients at each grid point can be edited. Ferguson patches are not suitable for 3D geometry, owing to the local 'flatness' at the corners of the patches [RA90], but have proven to be excellent for colour interpolation in 2D vector graphics, where they can be adapted to support manipulation of colours and colour gradients via rectangular control meshes, as demonstrated by Sun et al. [SLWS07].

As implemented in Illustrator, gradient meshes are topologically restricted to a rectangular grid, which limits the representable shapes to topological rectangles and annuli (achieved by looping the rectangle). We propose a colour interpolation scheme that relaxes this rectangular-grid restriction. Our solution supports gradient meshes of *arbitrary* manifold topologies as input. Figure 1(top right) shows two

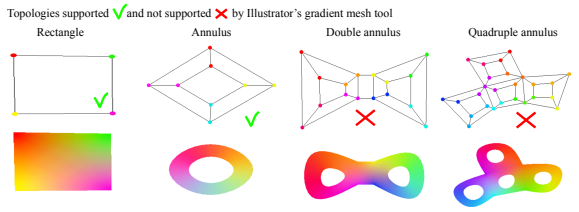

Figure 1: *Illustrator's gradient mesh tool, used for vivid colourings in vector graphics, is constrained to rectangular and annular control meshes (annuli are achieved by looping rectangles). The colour images were rendered with our method, which* does *support arbitrary surface topology.*

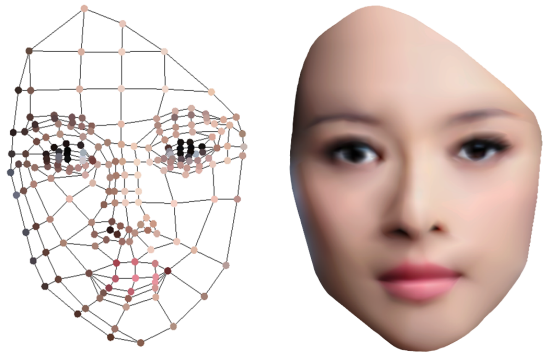

Figure 2: *A sparse, topologically-unrestricted gradient mesh evaluated with our method.*

simple topological layouts that cannot be achieved in the regular setting. Figure 2 illustrates a result of our method on a gradient mesh incompatible with Illustrator's gradient mesh tool.

The technical contribution of our work is our use of subdivision methods to achieve our goal. We note that it is well-known that certain subdivision methods can interpolate meshes of arbitrary manifold topology and that such methods can be used to interpolate data in 'other' dimensions than the 3D geometric space, such as in colour and texture dimensions [DKT98]. However, previous interpolation methods are unsuitable for the problem of interpolating *gradient meshes* of arbitrary mesh topologies in the setting of 2D vector graphics (Section 3.1) because either they wash-out the colours or they extrapolate colours beyond the colour gamut. Our key insight (Section 4) is that the input gradient mesh can be initially subdivided with separate subdivision rules in colour and geometry dimensions. Catmull-Clark subdivision [CC78] can then be performed to create a solution according to the problem specification (Section 3). In this initial subdivision procedure, a standard mechanism is applied in colour space to force colours to be interpolated. We then propose a special set of rules in the geometry dimensions to create a solution according to the expected behaviour of a gradient mesh. We are not aware of any previous method that applies such special rules in a subdivision procedure for the purpose of interpolating a gradient mesh for 2D vector graphics.

2. Related work

The gradient mesh primitive has been used extensively in previous work, targeting the two main approaches to vector graphics creation: manual and automatic [BB13]. Our work does not directly address these applications, as our aim is to improve the underlying colour interpolation problem. We envision that our method could be used to improve them in the future (Section 7). We note that modifications to the gradient mesh primitive have been proposed, notably by Xia et al. [XLY09], that use triangular patches instead of quads. However, such modifications are more appropriate for image vectorisation and are not suitable as a general gradient mesh interpolation scheme [BB13, §6.3]. By contrast, our solution offers colour interpolation of *arbitrary-topology* gradient meshes, making our method attractive as a general interpolation method for gradient meshes.

Our work is related to the definition of *complex gradients*, that is, colour gradients that provide more degrees of freedom compared to linear gradients in vector graphics. There are two established, and fundamentally different, vector primitives that provide such complex gradients: gradient meshes and *diffusion curves* [OBB*13]. We acknowledge that diffusion curves represent an attractive alternative to gradient meshes, especially related to its simple type of input (curves). We view these two primitives not as competitors, but as two primitives that complement one another: having both in the designer's toolbox strengthens the designer's abilities and efficiency.

3. Preparation

Our input is a mesh of arbitrary topology. Each mesh control point has an associated colour and associated *colour gradients* that relate to the way in which the colour propagates to the point's neighbourhood. There is thus a five-dimensional space (two spatial and three colour dimensions) with controllable constraints on the colour gradients in that space.

We wish to achieve a similar framework to the standard solution (Ferguson patches, which are limited to interpolating rectangular gradient meshes) that supports arbitrary topology. To achieve this, we impose the following conditions:

- **Condition 1**: The colour function must interpolate the original control points in colour space.
- **Condition 2**: The colour function must not stray outside the given colour space.
- **Condition 3**: Both geometry and colour function must be smooth (C^1) everywhere, except where lower continuities are explicitly specified.

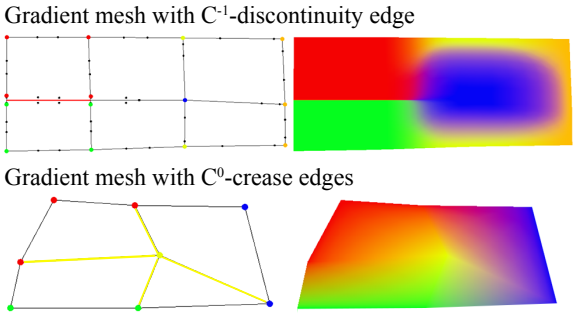

Figure 3: *Our input is a gradient mesh (left). Our output is an appropriate interpolation of the colours (right). The values stored at each mesh vertex are its colour (coloured discs) and its colour gradients (indicated by the locations of the smaller black discs). Additionally, edges can be specified as smooth (C^1, grey), crease (C^0, yellow), or discontinuous (C^{-1}, red). A discontinuous edge has separate colour and gradient control associated with either side of the edge.*

- **Condition 4**: The colour function must satisfy the colour gradients specified at each control point.

For any potential solution, it is straightforward to verify whether Conditions 1–3 are met. Condition 4 is more challenging because the term 'colour gradient' in vector graphics does not map exactly to the mathematical concept of gradient. In the following, we describe how we have interpreted this term in our work and in our evaluation. [See the supplementary document, Section 1, for further information supporting the claims made below.]

The naïve approach is to assume that the colour gradient specified at each mesh control point directly corresponds to the gradient of the resulting 5D surface. However, when we consider the constraints manipulated by the user in Illustrator's gradient mesh tool, we find they are defined only in the geometric domain (that is, in the *xy*-image plane) and cannot be manipulated in colour space. The map from these constraints to Ferguson patches' derivative constraints, which obviously require assignments in all dimensions, is therefore ill-defined.

We therefore chose to interpret the colour gradient constraints in a manner similar to that used in Illustrator. In Illustrator, derivatives in colour space seem to be set irrespective of the gradient constraints. That is, for all control points, the colour derivatives are fixed. Thus, the colour gradient constraints influence only the geometric co-ordinates and derivatives. Our analysis is that such treatment is, in fact, preferable because it is easier to avoid colour saturation when we use fixed colour derivatives. Thus, the concept of a 'colour gradient' does not correspond well to the mathematical definition of a gradient: a colour gradient represents simply some colour propagation between two or more colours.

In addition to the above, we further extend gradient meshes by supporting manipulation of the sharpness of the surface points related to the edges of the control mesh. Illustrator's gradient mesh tool does not support this latter feature, but a recent vectorisation approach [LHFY12] has demonstrated useful scenarios with C^0 creases (discontinuities of the first derivative) and C^{-1} discontinuities of the resulting surface. Thus, we allow specification of creases and discontinuities at edges.

In summary, our problem can be described as follows. Given a gradient mesh of arbitrary manifold topology as input (Figure 3(left)), where control points are associated with colours (we use, as the default, the CIE *Lab* colour space) and colour gradients, and edges are specified as being *smooth*, *crease*, or *discontinuous*, the resulting 5D surface (x, y, L, a, b) that represents the interpolation of the input data must satisfy Conditions 1–4 whilst emulating the behaviour of the current gradient mesh tool (Figure 3(right)).

3.1. Potential solutions

There are two types of approaches that are well-known to support interpolation over control meshes of arbitrary manifold topology: *Hermite interpolation* and *subdivision*. We investigated both and found that naïve application of either is inappropriate, but that the latter can be suitably modified to produce a good solution.

Hermite interpolation schemes have evolved to support arbitrary simple polygons, achieved by generalising barycentric co-ordinates [FS08, LJH13]. To achieve smooth colour interpolation over arbitrary meshes, each face can be separately interpolated, whilst ensuring that the gradient defined by the derivative constraints at each vertex is continuous. As mentioned previously, however, colour gradient constraints do not exactly match gradient or derivative constraints; a mapping, similar to Illustrator's approach in the regular setting, must therefore be defined.

However, there is a problem with how to specify derivatives at irregular vertices. The colour gradient constraints used by gradient meshes should ideally relate to the *edges* in the mesh. Thus, a control point of n-valency has n associated colour gradient constraints: one for each edge. The Hermite solution requires at most two directionally independent vectors to specify the gradient at each control point. Therefore adopting Hermite interpolation would restrict the artist's freedom more than we would like. Indeed, Ferguson patches also have this restriction.

The second alternative, subdivision, is well-known to support arbitrary manifold topology of the input control mesh and many subdivision schemes have been proposed to define surfaces of certain properties. Subdivision schemes can be classified as either *approximating* schemes, that produce surfaces that do not necessarily pass through the control points,

and *interpolatory* schemes, that produce surfaces that are guaranteed to interpolate the control points.

Both of these are, naïvely, inappropriate for our problem: approximating schemes do not necessarily interpolate their data points (breaching Condition 1), and interpolating schemes do not necessarily keep the underlying surface inside the convex hull of its control mesh (breaching Condition 2). The visual disadvantages are that approximating schemes tend to produce washed-out colours and interpolating schemes tend to produce C^0 creases due to colour clipping whenever colours are interpolated outside the gamut. [See the supplementary material for more information supporting these claims.]

A further problem with subdivision is that derivatives are not easily manipulated. This is because the input has only position information and does not have gradient information. Thus, standard subdivision schemes do not natively support manipulation of colour gradients (breaching Condition 4).

We therefore chose to extend a particular subdivision scheme to provide a mechanism that *emulates* the behaviour of derivative constraints. Our solution, described in the next section, offers a type of control and behaviour that cannot be easily replicated with schemes restricted to values and derivatives, like Hermite interpolation. In the regular setting, this behaviour is similar to that seen by Illustrator's gradient mesh tool (Section 6), thus justifying our choice of employing subdivision rather than Hermite interpolation.

Finally, we note that previous researchers have tackled a similar problem [BB13] in (gradient) mesh vectorisation: topology-preserving gradient meshes extracted from photographs; e.g. [LHM09, XLY09, LHFY12]. This problem is different to ours because the input data is a photograph (or user-specified image regions), that might give rise to challenging topologies, such as annuli. Thus, this problem is concerned with creating a valid gradient mesh according to a chosen colour interpolation method (Ferguson patches for [LHM09, XLY09] and Loop subdivision surfaces for [LHFY12]). By contrast, our problem is to take, *as input*, a gradient mesh of arbitrary topology and produce a solution according to Conditions 1–4. We note that previous methods in vectorisation cannot be used to solve our problem as they employ interpolation methods that are unsuitable for it (Ferguson patches, which restrict us to quadrilaterals and rectangular topology, and Loop subdivision, which washes-out the colours).

4. Method

We now describe our interpolation scheme. We want a solution that allows a control mesh that may contain vertices of arbitrary valency and polygons with an arbitrary number of edges. Recall that we want to associate each mesh control point with colour gradient constraints that specify the propagation of the control point's colour to its neighbourhood.

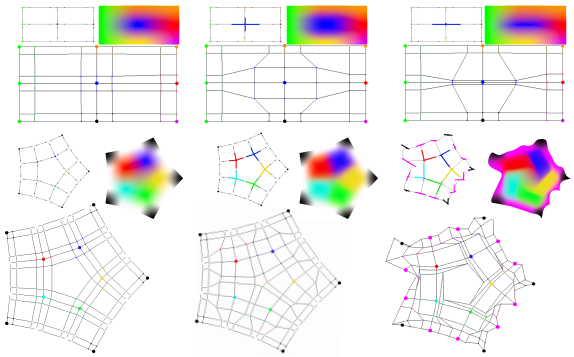

Figure 4: *Six examples, each showing the input mesh (top left), with gradient constraints highlighted with bold coloured lines, the resulting image (top right), and the control mesh (bottom) computed by our first, ternary, step of subdivision. Notice how the colour gradient constraints influence the local propagation of the colours and the boundary of the figures.*

Because derivative constraints are not easily defined and satisfied in the irregular setting, we create a mechanism that emulates the behaviour of such constraints. A 2D vector, referred to as a colour gradient constraint, is associated with each incident edge of all control points (Figure 4). The local colour gradient around a control point is thus controlled by the point's colour, its neighbours' colours, and its colour gradient constraints.

To achieve Conditions 1–3, we split the treatment of geometric and colour co-ordinates. That is, separate subdivision schemes are employed for the colour and geometric dimensions, with the constraint that the two subdivision schemes generate the same topology. The colour co-ordinates are produced according to a set of fixed rules (Sections 4.2 and 4.3). The geometric co-ordinates can be produced in various ways, which depend on user input and algorithmic decisions (Section 4.5).

4.1. Overview

To explain the genesis of our method for colour interpolation, we start by recalling (Section 3.1) that approximating subdivision schemes do not generally interpolate the data (Figure 5(a)). We then observe that a uniform cubic B-spline surface or a Catmull-Clark subdivision surface can be forced to interpolate colour data if the surrounding ring of vertices all have the same colour. This is most easily seen in the univariate (curve) case (Figure 5(c)), where placing vertices of equal colour either side of a vertex forces the resulting curve to interpolate that colour.

This observation leads to our proposed method. That is: for every vertex in the original mesh, we introduce a ring of vertices around it that have the same colour. This forces

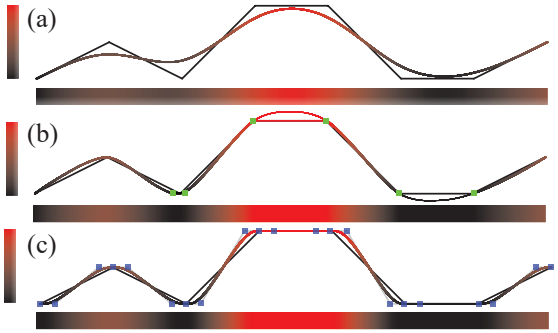

Figure 5: *Colouring a horizontal bar with (a) a uniform cubic B-spline curve, (b) the interpolatory 4-point subdivision scheme [DLG87], and (c) our solution. The blue squares (c) show the control points defined in the special initial subdivision step. The alternative subdivision schemes do not satisfy all conditions: The uniform B-spline curve (a) does not interpolate the original colours (breaching Condition 1). The interpolatory curve (b) strays outside the colour space (breaching Condition 2).*

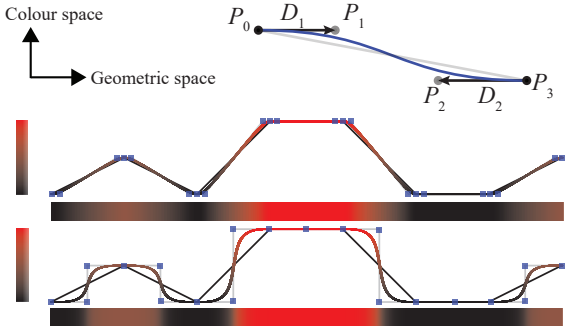

Figure 6: *Influence of the colour gradient constraints. (a) Illustration of subdivision of an edge. (b) Short vectors produce curves that approximate the control polygon more closely. (c) Long vectors result in larger influence of the given colour.*

interpolation (Condition 1) while maintaining the property that the final surface lies within the valid colour space (Condition 2). The refined control mesh therefore defines a Catmull-Clark surface that satisfies Conditions 1 and 2. Additionally, Catmull-Clark subdivision is well-known to produce surfaces with at least C^1 continuity everywhere (Condition 3), where the sharpness at edges can be optionally manipulated to produce C^0 creases and C^{-1} discontinuities.

We are still left with several questions: at what geometric location should we place these new vertices to satisfy Condition 4? (Sections 4.2 and 4.3); are the results plausible? (Section 5); does this method produce similar behaviour to the current gradient mesh tool? (Section 6).

4.2. A univariate subdivision scheme for curves

To address the question of geometric location, we first consider the univariate (curve) case.

Given a control polygon, our method runs a single special ternary subdivision step that creates a new set of control points that define a uniform cubic B-spline curve satisfying the given constraints. There are two types of points in the subdivided polygon. We refer to new points associated with vertices as *vertex points*, and new points associated with edges as *edge points*. New vertex points are defined at the same co-ordinates, both in colour and geometry, as the original vertex. That is: the original vertices do not move and are included in the refined polygon.

New edge points are defined differently for colour and geometric co-ordinates. For a given edge, two new edge points are defined. Let $P_i = (x_i, y_i, L_i, a_i, b_i)$ be a point in our 5D space (2D in geometry, 3D in colour). Let P_0 and P_3 be the original vertices defining the edge. Each vertex has associated colour gradient constraints: D_1 associated with P_0 and D_2 associated with P_3. These vectors have zero magnitude in the three colour dimensions. The new edge points, P_1 and P_2, are calculated by:

$$P_1 = P_0 + D_1,$$
$$P_2 = P_3 + D_2,$$

giving a geometric offset in (x, y) to each new point. In colour space, each original point is surrounded by points of the same colour, so $(L_1, a_1, b_1) = (L_0, a_0, b_0)$ and $(L_2, a_2, b_2) = (L_3, a_3, b_3)$.

Any standard B-spline rendering method can be employed to produce the resulting curve from the refined polygon. Figure 5 compares our solution with interpolatory and approximating subdivision. Figure 6 demonstrates the effect of the colour gradient constraints.

4.3. A bivariate subdivision scheme for surfaces

In this section, we generalise the univariate solution to bivariate subdivision.

As in the univariate case, the first step is a single modified linear ternary subdivision step which produces a refined control mesh. In addition to edge and vertex points, which are analogous to those in the curve case, there are new vertices associated with old faces, which are referred to as *face points*.

The linear ternary subdivision step subdivides each face of n-valency into a smaller face of n-valency, surrounded by quadrilaterals. Each edge of the original face is associated with three new quadrilaterals on each of its two sides. This single-step subdivision procedure is shown in Figure 7(d).

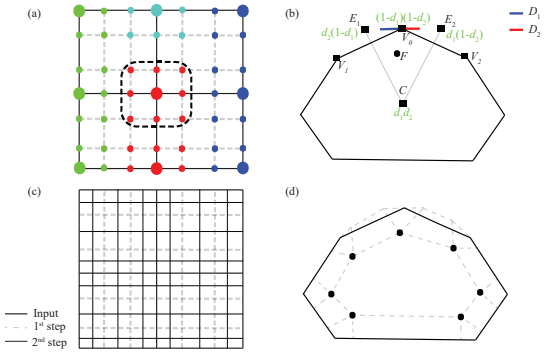

Figure 7: *Our solution performs a single step of modified linear ternary subdivision followed by standard Catmull-Clark subdivision. (a) In colour space, all new edge, vertex, and face points associated with an original point (large disks) are assigned the colour of that point. Thus, the one-ring of points around an original point all have the same colour. (b) The geometric location of a new face point F (black disk) is defined by bilinear interpolation over a quadrilateral defined by the original vertex V, the points $E_{\{1,2\}}$, and the centroid C of the face. The weights of the bilinear interpolation (green) are defined by the related colour gradient constraints, D_1 and D_2. (c) The mesh configuration of a 2×2 grid after two steps of subdivision. (d) The first step produces a centre face of the same valency as the original face, surrounded by quadrilaterals.*

The remaining geometric co-ordinates are described next. New edge and vertex points are defined as in the univariate solution. New face points are defined as follows. Given a control point V_0, two of its neighbours $V_{\{1,2\}}$, and one of its incident faces, where the centroid of that face is denoted as C, the geometric (x,y) location of the new face point F is defined via bilinear interpolation as (Figure 7(b)):

$$F = (1-d_1)(1-d_2)V_0 + d_1 d_2 C + d_2(1-d_1)E_1 + d_1(1-d_2)E_2,$$

where the intermediate points, $E_{\{1,2\}}$, and the weights, $d_{\{1,2\}}$, are defined as:

$$\begin{aligned} E_1 &= V_0 + \|(V_0 - V_1)\| \hat{D}_1/2, \\ E_2 &= V_0 + \|(V_0 - V_2)\| \hat{D}_2/2, \\ d_1 &= 2\|D_1\| / \|(V_0 - V_1)\|, \\ d_2 &= 2\|D_2\| / \|(V_0 - V_2)\|, \end{aligned}$$

where $\hat{D}_{\{1,2\}}$ denote the normalised vectors of $D_{\{1,2\}}$.

The refined control mesh defines a Catmull-Clark surface and any standard method (e.g. [CC78, NLMD12]) can therefore be employed to produce it.

Implementation details

We implemented Pixar's version [DKT98] of Catmull-Clark subdivision, which supports sharp and boundary edges. We have found it sufficient to perform two subdivision steps in our examples, but more steps might be necessary for high-resolution images. We then render the refined, pixel-dense, meshes with OpenGL. Thus, our implementation approximates the subdivision surface by rendering a pixel-dense piecewise linear mesh. Note that in practice, one could use an existing optimised library such as Pixar's OpenSubdiv [NLMD12] for this task.

If the input gradient mesh is rectangular and has no crease or discontinuous edges, our method produces a set of bi-cubic Bézier patches and standard methods to produce Bézier patches can optionally be employed.

To produce C^0 creases and C^{-1} discontinuities along original edges, Pixar's sharp rules [DKT98] are employed. Discontinuous edges need to be associated with separate colours on each side of the edge. That is, control points are split along discontinuous edges, where the geometric co-ordinates remain the same and the colour co-ordinates are different. These control points have been split in the software that produces the input control mesh; thus we assume that the colours on each side of a discontinuous edge have already been defined. Figure 3 shows an example of how discontinuous edges can be represented in a user interface.

Further, original vertices on the boundary of the mesh are interpolated to emulate curve editing in vector graphics. This is achieved by applying the univariate rules along the boundary; see Section 4.2. Thus, colour gradient constraints along the mesh boundary influence both the colour gradients and the geometric shape of the underlying figure.

4.4. Satisfaction of conditions

Condition 1 is satisfied because the resulting surface is forced, by the ring of points of identical colour introduced in the ternary step, to interpolate the colour of the original control points. This is guaranteed because, on all subsequent Catmull-Clark steps, the new colours of the control point and of the 1-ring of new points around it are determined by its previous colour and the colours of the 1-ring of old points around it. Those points are all the same colour and therefore the control point and the new 1-ring also stay that same colour. In the limit, as the polygons get increasingly tiny, only the control point itself and an infinitesimal region around it are forced to stay that colour. Condition 2 is satisfied due to the convex hull property of Catmull-Clark subdivision. Condition 3 is satisfied because the limit surface of the Catmull-Clark subdivision is guaranteed to be C^1 and we render a version of the surface where individual polygons are of the same size as pixels, making it visually indistinguishable from the limit surface. Condition 4 is implicitly satisfied because we

have the freedom to choose what is meant by "satisfy the colour gradients" when we implement the algorithm.

4.5. Options

There are several aspects of our solution that are open for discussion. We address an alternative subdivision scheme, the various options available in the geometric space, boundary treatment, and the question of flat spots in colour space, and folds.

Two-step binary refinement An alternative to the ternary step is instead to perform two binary subdivision steps. This was our initial solution to the problem, but we now regard it as inferior to the ternary solution. An advantage of two binary steps is that it produces a pure quadrilateral mesh, thus replacing the relatively large central polygon with smaller quads. As with the ternary step, the 1-ring of faces around original control points is unique for all original control points. It is therefore possible to achieve the conditions in a similar manner to our ternary solution. However, it is also necessary to define co-ordinates for a set of additional points. Additionally, our experiments show that the visual results are marginally smoother with the single-step ternary scheme, especially for larger colour gradients. [See the supplementary web page for comparisons with this alternative subdivision scheme.]

Geometric co-ordinates Our solution does not interpolate the *geometric* co-ordinates of the input control points. In contrast, Ferguson patches *do* interpolate both colour and geometric co-ordinates. We have not noticed visual differences between the two approaches (Section 6), probably owing to the fact that the limit point V^∞ (i.e., the point on the limit surface) corresponding to a vertex V^0 in the input mesh lies close to V^0 due to the initial ternary step and the local convex hull property of Catmull-Clark subdivision. However, we note that the initial ternary step can be tuned to ensure that geometric co-ordinates are interpolated, i.e., $V^\infty = V^0$ for all vertices. This could be achieved by using the limit stencil given by Loop et al. [LSNC09, §3.2], which imposes a single linear condition on the 1-ring of V^1, i.e., the vertex's location after the initial ternary step. Moreover, since these 1-rings do not overlap, the linear conditions for all V^1 do not form a global system; they are independent of each other and can thus be easily incorporated.

Treatment at the surface boundary The careful reader will notice that the boundaries of the two methods are slightly different; see Figure 8. This is because Illustrator's gradient meshes are bounded by cubic Bézier (Ferguson) curves, while our meshes are bounded by uniform cubic B-spline curves. For the user, the interface is the same: moving control points and adjusting colour gradients change the shape and position of the boundary. We expect that users would have little trouble with the slight differences in boundary behaviour. If it were thought that users would find it disturbing then it is possible to modify the boundary rules of our method to give a result that matches cubic Bézier curves. However, such modification would require more complicated, non-uniform subdivision rules at boundary vertices and edges, and preclude direct use of existing subdivision schemes and libraries such as Pixar's.

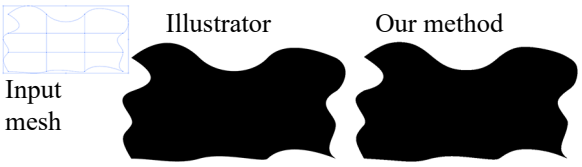

Figure 8: *Illustrator's boundary curves are represented by C^1 Bézier cubics whereas our method produces C^2 uniform cubic B-splines. The magnitudes of colour gradient constraints in our method were adjusted to closely follow the boundary curve produced by Illustrator.*

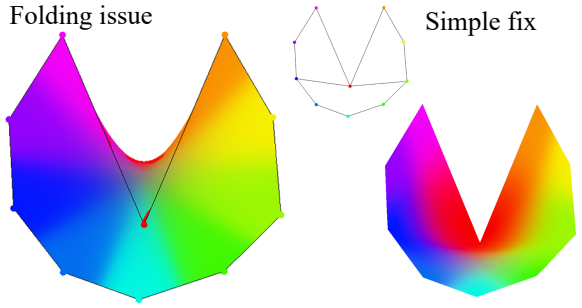

Figure 9: *Concave mesh elements might lead to folds (left), which can be fixed by increasing convexity (right).*

Flat spots Owing to the zero derivatives that occur at every control point, our solution produces flat spots in colour space at the control points. This is identical to the behaviour of Ferguson patches at the control points. In 3D mesh modelling, such flat spots would be undesirable and, indeed, this is the reason that Ferguson patches are not used in a 3D mesh modelling context [RA90]. However, for this application of interpolating colour on a 2D plane, both Ferguson patches and our method are acceptable. This is because the human eye's response to colour gradient is dramatically different to its response to geometric gradient. Our method does offer the potential for the colour gradient at the control points to be set to a value other than zero: it would be a trivial extension to the current method. But, while possible, it is difficult to see how such a facility could be presented easily to the user or whether it would be useful at all.

Folds Folds can be produced in concave mesh regions, as illustrated in Figure 9(left). Such folds are created by the user, who is responsible for designing the geometry of the mesh. Nevertheless, certain decisions in the design of the algorithms do influence the chances of producing folds.

For example, by interpolating geometric co-ordinates, as discussed above, folds are more likely to be produced. The software could assist by indicating potential problems at high concave regions. A simple fix to such folds is to increase the convexity of the region, as illustrated in Figure 9(right).

5. Results and discussion

In this section, we discuss visual results, performance, and the advantages of multi-resolution gradient mesh editing.

Visual results Results are shown in the various figures in this paper and in the supplementary material. Control mesh visualisations after the initial ternary subdivision step are shown in Figure 4 and in the supplementary web page and video.

Our experimental results are generally on meshes with large spatial extents and with primary colours. This is because these give the best view of any infelicities in the method: a large spatial extent between adjacent points of dramatically different colour allows us to see best the effect the colour propagation of each control point and therefore to analyse most clearly the behaviour of our method. Colours are interpolated in *Lab* colour space, except where we are comparing against other methods that use *RGB* colour space, in which case we also use *RGB* for fair comparison.

Performance Our solution is computationally efficient. The initial ternary subdivision of the input gradient mesh is efficient because the subdivision procedure requires only local linear combinations of control points to compute the co-ordinates of the new control points. Using our naïve C++ single-core CPU implementation, tested with an Intel SU4100 1.3 GHz CPU, our software produced the refined control mesh in the range of <1 millisecond to 20 milliseconds in the gradient meshes we tested. Additionally, our solution is GPU-friendly because subdivision using local linear combinations of control points can be performed using shaders. Finally, current methods for producing Catmull-Clark surfaces (e.g. [NLMD12]) can be employed to produce the final surfaces efficiently.

Multi-resolution editing A particular advantage of subdivision, as demonstrated in surface modelling for animation [DKT98], is the native support for multi-resolution editing [Zor06]. That is, meshes can be edited after each step of subdivision as the mesh is progressively refined (Figure 11). Thus, meshes at the coarsest level can be relatively sparse, while still supporting complex colourings in the interior of the object. Multi-resolution editing is challenging when attempted with Ferguson patches and is not available in Illustrator's tool.

For the purpose of artistic design, we implemented a version that constrains our multi-resolution mechanism so that only colour can be manipulated at finer levels of detail (Figure 10). Geometry can be edited only on the sparse, original, mesh. This dramatically reduces the complexity of interaction for the user.

One could also follow the approach taken in [FB88] and support local editing via a hierarchical construction. However, hierarchical refinement does not offer topological freedom and cannot be used to change the flow of a mesh. By contrast, extraordinary elements (vertices and faces of arbitrary valency) supported by our method offer full topological freedom via subdivision.

6. Comparisons and visual evaluation

In this section, we compare our method with Illustrator's gradient mesh tool and discuss how our solution behaves, visually, compared to it. CorelDRAW has practically identical functionality to Illustrator, so comparisons with Illustrator are sufficient to demonstrate comparisons between our method and existing methods.

Mathematical comparison In the regular setting, our method produces C^2 bi-cubic Bézier patches with gradient control. The gradient control associated with an original control point is related to the 1-ring neighbourhood of the point after the ternary subdivision step. The new control points in this neighbourhood lie on a plane, which relate to the tangent plane of the surface point associated with the original control point. The magnitude of the gradient is controlled by the geometric co-ordinates of the control points in the 1-ring neighbourhood, which are dictated via the colour gradient constraints. This gradient control is analogous to Ferguson patches' gradient control, where the derivative constraints give rise to the tangent plane of the surface point associated with the original control point. Thus, both methods lead to bi-cubic patches over regular elements and offer similar user control over shape and colour. Additionally, both methods yield (piece-wise) cubic boundary curves.

In the irregular setting, our solution, which produces Catmull-Clark surfaces, provides sensible behaviour according to the colour gradient constraints. In this setting, the 1-ring neighbourhood of an original control point, defined after the ternary step, models the related local surface neighbourhood according to the colour gradient constraints. This local surface patch is defined as a regular spline surface, surrounding an isolated point related to the original irregular control point [PR08]. The continuity at this isolated point is C^1 and the continuity everywhere else is C^2. Figure 13(bottom) demonstrates the visual behaviour of the solution at an irregular control point with different configurations of the colour gradient constraints.

Visual evaluation In the regular setting, our method produces visually indistinguishable results compared with Illustrator's tool (see Figures 12(top) and 13 and the supplementary web page for visual comparisons). This behaviour is extended to the irregular setting as demonstrated in Figure 13(bottom), where the red colour propagates naturally

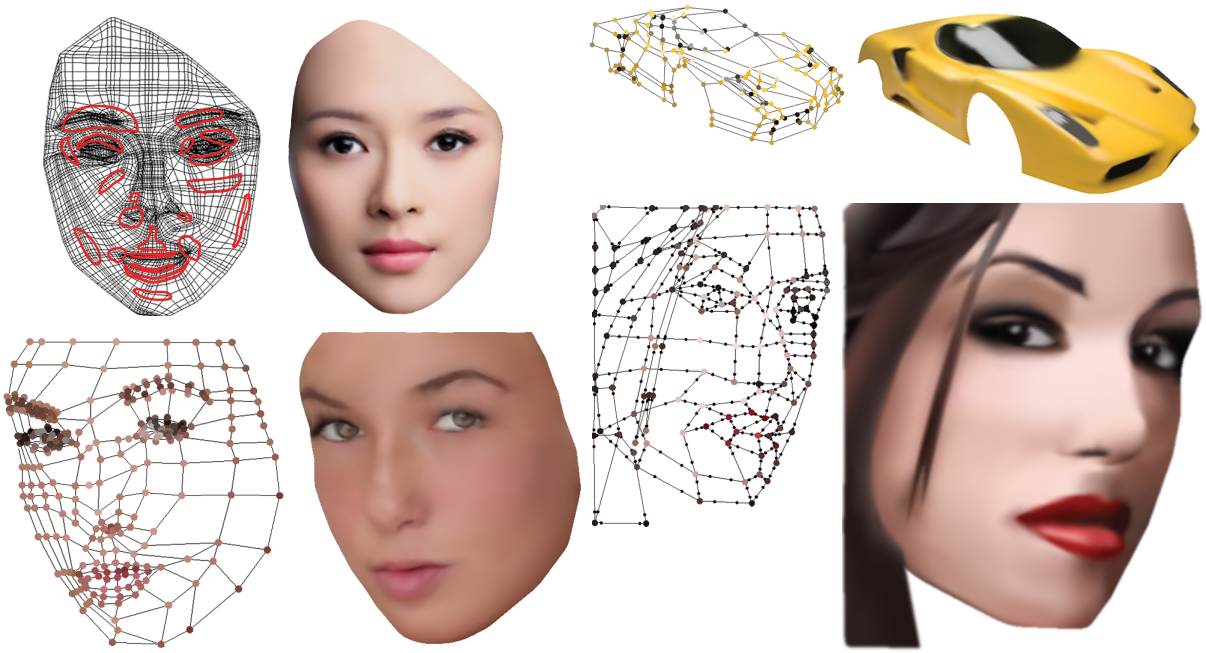

Figure 10: *Gradient meshes created by the authors, inspired by gradient mesh art found on the Internet (see the supplementary web page for details). The results were created using relatively sparse gradient meshes as input. The top-left image shows an example of how details can be added via multi-resolution editing after two subdivision steps on the example from Figure 2. The regions outlined in red have vertices whose* colour *has been edited after subdivision. Any* geometric *changes are edited only at the sparse, original, level. Similar edits were made to the other three results.*

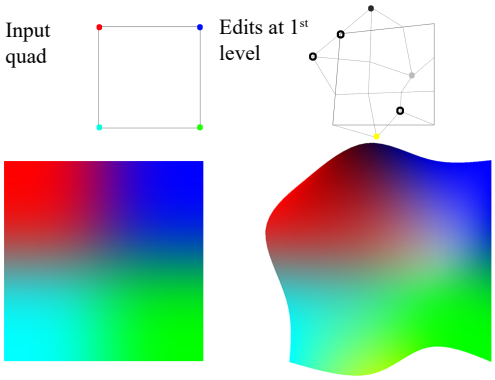

Figure 11: *Multi-resolution editing after the first subdivision step. Top right: coloured disks represent vertices that have been moved and recoloured; black rings represent vertices that have been moved but not recoloured.*

according to the colour gradient constraints. Notice how these constraints give rise to a circular profile of the colour propagation around the original point (blue curves in the figure). This profile is indirectly modelled via the colour gradient constraints. It can be optionally modelled directly via manual multi-resolution edits after the first subdivision step.

The added capability of supporting non-rectangular mesh elements can have practical advantages. One practical advantage of meshes of arbitrary topology lies in the fact that such meshes are not *unnecessarily* dense. Nevertheless, dense meshes *are* required to capture finer image details. We suggest our subdivision-based approach can be advantageous over Illustrator's tool in such settings, since the meshes can be initially sparsely defined. Then, the finer details can be manipulated by multi-resolution colour adjustments (Figure 10) or by insertion of detailed geometry only where needed (Figure 12).

We note that we have addressed the interpolation problem and not the problem of creating an efficient user interface for gradient mesh editing. However, we suggest the points mentioned in this section can be exploited to either improve current features of the gradient mesh tool or to add new features to it.

7. Future work

Subdivision technology has enabled software developers to create efficient and easy-to-learn tools for 3D object manipulation and animation. One prominent example is Pixar's

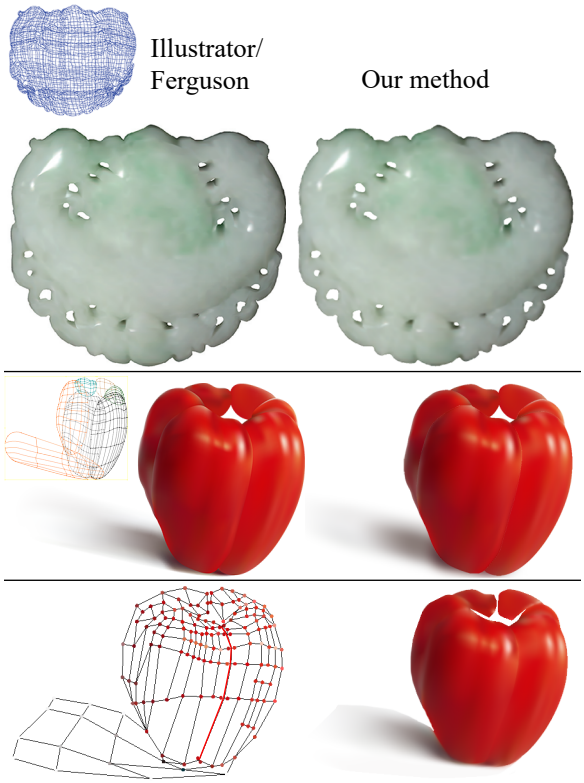

Figure 12: Top two rows: *Two rectangular gradient mesh examples rendered with Illustrator's gradient mesh tool (left) and our method (right). The top example is a single mesh; the pepper comprises seven overlapping gradient meshes. The meshes used for our method were manually traced from gradient meshes provided by lifeinvector.com©. The colour gradient constraints were adjusted to best match Illustrator's results.* Bottom row: *A similar pepper rendered with our method (right) from a single, less dense, mesh (left) manually defined with SketchUp.*

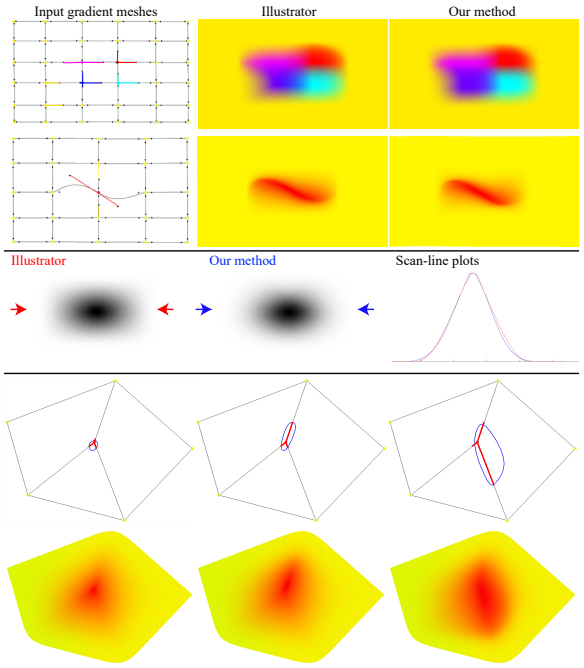

Figure 13: Top: *Colour gradient constraints (top left) applied using Illustrator's gradient mesh tool (top centre) and our method (top right). Manipulated constraints are highlighted with coloured lines.* Middle: *Simplest-case comparison with scan-line plots applied across the middle row of the image. This comparison indicates that the rendering methods employed vary between the two methods, where our rendering method produces a marginally smoother surface approximation.* Bottom: *Behaviour of colour gradients at an irregular vertex. The propagation of the red colour relates to its 1-ring neighbourhood, which is influenced by the colour gradient constraints. The blue curves are cubic B-splines defined by the vertices of the 1-ring neighbourhoods and they thus relate to the shape of the red colour in that local neighbourhood. [See the supplementary web page for more visual (high-resolution) comparisons and further data.]*

RenderMan. One hypothesis is that similar subdivision-tuned modelling features, such as arbitrary-topology meshes and multi-resolution edits, can make the gradient mesh tool easier to learn and easier to work with, compared with Illustrator's tool based on Ferguson patches. We aim to create such an alternative gradient mesh tool to test this hypothesis in future work.

The vectorisation problem is closely linked to our problem because any vectorisation method must employ some chosen colour interpolation method, like Ferguson patches [SLWS07, LHM09, XLY09, LJH13] and Loop subdivision [LHFY12]. One hypothesis is that our interpolation method can give rise to practical advantages compared with previous approaches. We believe it would be interesting to study this hypothesis in future work. Figure 14 shows an initial experiment of ours that indicates potential improvements using our method.

8. Conclusion

We have introduced a viable gradient mesh primitive for vector graphics in the setting of control meshes of arbitrary manifold topologies. Our solution, using Catmull-Clark subdivision, emulates the previous gradient mesh tool restricted to rectangular grids and improves on it by allowing arbitrary mesh topology. By contrast to standard subdivision schemes, our approach guarantees that input colours are interpolated, that the colour interpolant is defined inside the given col-

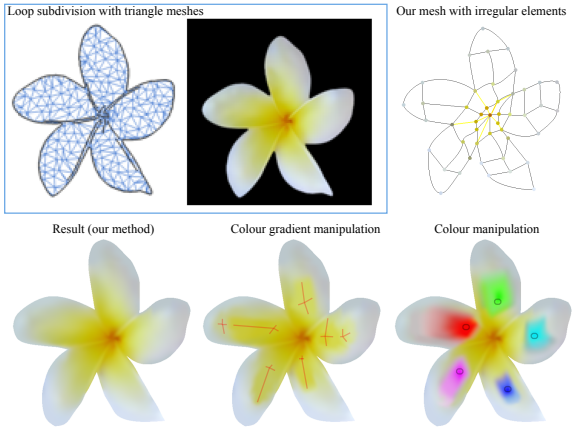

Figure 14: *An initial experiment illustrates that our method could potentially be used as an underlying interpolation method for the vectorisation problem. Note that the gradient mesh created as input for our method was manually created by the authors, whereas the input for the Loop subdivision method was automatically created by the method of Liao et al. [LHFY12]. An interesting avenue for future work is to create an automatic vectorisation method that takes advantage of our method (such as topological freedom and multi-resolution refinements).*

our space, that the interpolant is smooth across the domain, and that the interpolant behaves according to colour gradient constraints.

References

[BB13] BARLA P., BOUSSEAU A.: Gradient art: creation and vectorization. In *Image and Video-Based Artistic Stylisation*, Rosin P., Colomosse J., (Eds.). Springer, 2013, ch. 8, pp. 149–166. 2, 4

[CC78] CATMULL E., CLARK J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (1978), 350–355. 2, 6

[DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *Proc. SIGGRAPH* (1998), ACM, pp. 85–94. 2, 6, 8

[DLG87] DYN N., LEVIN D., GREGORY J. A.: A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design 4*, 4 (1987), 257–268. 5

[FB88] FORSEY D. R., BARTELS R. H.: Hierarchical B-spline refinement. *Proc. SIGGRAPH* (1988), 205–212. 8

[Fer64] FERGUSON J.: Multivariable curve interpolation. *J. ACM 11*, 2 (1964), 221–228. 1

[FS08] FLOATER M. S., SCHULZ C.: Pointwise radial minimization: Hermite interpolation on arbitrary domains. In *Proc. Symposium Geometry Processing* (2008), Eurographics, pp. 1505–1512. 3

[LHFY12] LIAO Z., HOPPE H., FORSYTH D., YU Z.: A subdivision-based representation for vector image editing. *IEEE Trans. Vis. Graph. 18*, 11 (2012), 1858–1867. 3, 4, 10, 11

[LHM09] LAI Y.-K., HU S.-M., MARTIN R. R.: Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Trans. Graph. 28*, 3 (2009), 85:1–8. 1, 4, 10

[LJH13] LI X.-Y., JU T., HU S.-M.: Cubic mean value coordinates. *ACM Trans. Graph. 32*, 4 (2013), 126:1–10. 3, 10

[LSNC09] LOOP C., SCHAEFER S., NI T., CASTAÑO I.: Approximating subdivision surfaces with Gregory patches for hardware tessellation. *ACM Trans. Graph. 28*, 5 (2009), 151:1–9. 7

[NLMD12] NIESSNER M., LOOP C., MEYER M., DEROSE T.: Feature-adaptive GPU rendering of Catmull-Clark subdivision surfaces. *ACM Trans. Graph. 31*, 1 (2012), 6:1–11. 6, 8

[OBB*13] ORZAN A., BOUSSEAU A., BARLA P., WINNEMÖLLER H., THOLLOT J., SALESIN D.: Diffusion curves: A vector representation for smooth-shaded images. *Commun. ACM 56*, 7 (2013), 101–108. 2

[PR08] PETERS J., REIF U.: *Subdivision Surfaces*. Springer, New York City, USA, 2008. 8

[RA90] ROGERS D. F., ADAMS J. A.: *Mathematical Elements for Computer Graphics*, 2 ed. McGraw-Hill, 1990. 1, 7

[SLWS07] SUN J., LIANG L., WEN F., SHUM H.-Y.: Image vectorization using optimized gradient meshes. *ACM Trans. Graph. 26*, 3 (2007), 11. 1, 10

[XLY09] XIA T., LIAO B., YU Y.: Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph. 28*, 5 (2009), 115:1–10. 2, 4, 10

[Zor06] ZORIN D.: Modeling with multiresolution subdivision surfaces. In *SIGGRAPH Courses* (2006), ACM, pp. 30–50. 8