# Local and Hierarchical Refinement for Subdivision Gradient Meshes

Verstraaten, T. W.; Kosinka, J.

*Published in:*
COMPUTER GRAPHICS FORUM

*DOI:*
10.1111/cgf.13575



*Publication date:*
2018

DOI: 10.1111/cgf.13575
Pacific Graphics 2018
H. Fu, A. Ghosh, and J. Kopf
(Guest Editors)

*Volume 37 (2018), Number 7*

# Local and Hierarchical Refinement for Subdivision Gradient Meshes

T. W. Verstraaten[1] and J. Kosinka[1]

[1]Bernoulli Institute, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands

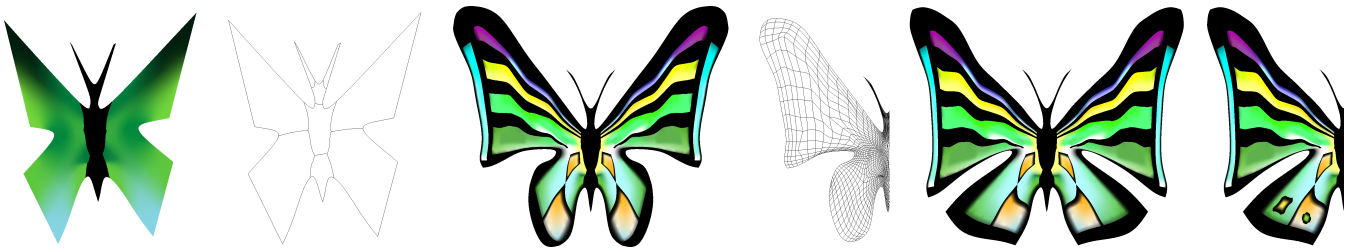

**Figure 1:** ***From the left:*** *An initial, level 0 design of a butterfly model. Its mesh consists of only nine polygons. The model edited at several levels (up to level 4) via local refinement of geometry and/or colour. Half of the butterfly mesh at level 4. The overall shape of the model is adjusted at level 0; the refinements are expressed in local frames tied to the mesh, and thus follow adjustments of the coarse mesh naturally. The mesh can then be again edited at finer levels, adding more detail (lower left wing). At any level, colour can be assigned to a vertex as a whole, or to any of its segments wedged between two incident edges to introduce sharp colour transitions (Method 2).*

**Abstract**

*Gradient mesh design tools allow users to create detailed scalable images, traditionally through the creation and manipulation of a (dense) mesh with regular rectangular topology. Through recent advances it is now possible to allow gradient meshes to have arbitrary manifold topology, using a modified Catmull-Clark subdivision scheme to define the resultant geometry and colour [LKSD17]. We present two novel methods to allow local and hierarchical refinement of both colour and geometry for such subdivision gradient meshes. Our methods leverage the mesh properties that the particular subdivision scheme ensures. In both methods, the artists enjoy all the standard capabilities of manipulating the mesh and the associated colour gradients at the coarsest level as well as locally at refined levels. Further novel features include interpolation of both position and colour of the vertices of the input meshes, local detail follows coarser-level edits, and support for sharp colour transitions, all at any level in the hierarchy offered by subdivision.*

**CCS Concepts**

•***Computing methodologies*** → *Graphics systems and interfaces; Parametric curve and surface models;*

## 1. Introduction

Gradient meshes are a powerful vector graphics primitive for creating scalable illustrations. *Traditional gradient meshes* are rendered as bicubic (Ferguson) patches over a regular rectangular mesh, the vertices of which are assigned colours by the user; see e.g. [SLWS07, BLHK18]. Clearly, the necessity of a regular rectangular mesh for bicubic patches serves as a severe restriction on the artist designing the mesh and thus also the final image.

Through recent advances it is now possible to create gradient meshes of arbitrary manifold topology [LKSD17, SL17]. The core idea behind these *subdivision gradient meshes* is the use of subdivision surfaces to define the emergent colour surface. Starting from a user defined mesh, a colour surface is produced by applying a single modified ternary subdivision step and then proceeding with Catmull-Clark subdivision [CC78]. This ensures $\mathcal{C}^2$ colour surfaces almost everywhere. This method allows artists to include faces and vertices of arbitrary valency and produces a well-defined colour surface provided the faces are convex. The same convexity condition applies also to traditional gradient meshes.

The subdivision gradient mesh tool allows for more flexibility than the traditional gradient meshes. However, when the user wants to locally add more detail to their mesh, this has so far been possible only in the strict setting of regular rectangular meshes [BLHK18]. Although the authors of [LKSD17] mention hierarchical editing as

enabled via subdivision, their method provides only the naive approach offered by all subdivision schemes and it does not preserve detail when a coarser level is edited and does not, in general, interpolate colour at refined levels.

We focus on facilitating local editing at any level in the hierarchy that ensures preservation of local detail; see Figure 1. The main contributions of this paper are:

- We present two novel methods that allow for local and hierarchical refinement in subdivision gradient meshes. Both methods ensure interpolation of both position and colour.
- We design a method for storing the refinement of geometry in local coordinate frames, thus preserving details when coarser levels are (re)edited.
- We enrich subdivision gradient meshes by allowing sharp colour transitions at any level in the hierarchy.

We start by reviewing relevant related work (Section 2). To simplify the exposition of our methods, we first present our ideas in the more accessible univariate case (Section 3) and then move on to the full bivariate case (Section 4). We then present two methods for hierarchical subdivision gradient mesh editing: Method 1 (Section 5) is based on control vectors [KSD15], which allows for adding details via special blending functions that have to be constructed. Method 2 (Section 6) relies on direct colour adjustment at any level in the hierarchy, and in contrast to Method 1 supports sharp colour transitions. We then showcase and discuss our results (Section 7) and conclude the paper (Section 8).

## 2. Related work

The (traditional) gradient mesh tool first appeared in Adobe Illustrator as the 'Mesh object' [Ado98], and is now available, in slightly different forms [BLHK18], also in CorelDRAW and Inkscape. All these implementations rely on a regular grid of quadrilateral patches, rendered as Ferguson or Coons bicubic patches. Although very popular among artists, this traditional gradient mesh suffers from two major limitations that may hinder its usability and expressive power: the restriction to regular rectangular arrays and the lack of support for local refinement.

The first restriction, concerning topology, has been addressed and alleviated by employing either triangular Bézier patches [XLY09], or the concept of Loop subdivision surfaces [Loo87] which naturally supports meshes of arbitrary manifold topology [LHFY12, ZZW14]. Subdivision surfaces have also been used in the *shading curve* primitive [LTKD15]. More recently, [LKSD17] showed that colour interpolation can be achieved by using a modified ternary subdivision step before proceeding with Catmull-Clark subdivision to produce the limit colour surface. We build on this work by adding support for geometry interpolation and sharp creases at any level in the hierarchy arising via subdivision.

Mathematically exact and local refinement for traditional gradient meshes has been addressed recently in [BLHK18], lifting the limitation of global refinement. Our approach offers a similar type of local refinement, but based on subdivision. To ensure local editing and preservation of detail, pioneered in [FB88] in the context of hierarchical B-splines, we take advantage of the concept of *control vectors* [KSD15] and encode hierarchical edits in local frames.

Further details on gradient meshes can be found in the survey [BB13], which also discusses *diffusion curves* [OBW*08, FSH11], a different vector graphics primitive and a popular alternative to (extended) gradient meshes. Our contribution focuses on gradient meshes, but we borrow the idea of sharp colour transitions from diffusion curves and allow them in our flexible primitive.

## 3. Preliminaries: the univariate case

We start by presenting our approach to local refinement of geometry and colour in the simpler, univariate setting. Consider the simplified case of a parametrised (colour) curve $S(t) = \sum_{i=1}^{N} B_i(t)\mathbf{p}_i$, where $\mathbf{p}_i$ are (colour) control points. We assume for the moment that $S$ has only the red colour channel $r$, i.e., that $S(t) = [x(t), r(t)]^\top$, where $x$ stands for position. We make no assumptions on the shape of the $B_i(\cdot)$. We investigate the idea of basis enrichment by the use of control vectors as detailed in [KSD15]. To that end, we define a refined colour curve $\tilde{S}(t)$ via

$$\tilde{S}(t) = \sum_{i=1}^{N} B_i(t)\mathbf{p}_i + \sum_{j=1}^{M} f_j(t)\mathbf{c}_j. \tag{1}$$

In this context, the $\mathbf{c}_j$ are to be understood as *control vectors* in that they do not specify a position in space but rather a displacement. The $f_j$ are blending functions that we need to specify. In this manner, $\mathbf{c}_j = \mathbf{0}$ for all $j$ guarantees that $S(t) = \tilde{S}(t)$, i.e., the original curve is recovered if the control vectors are set to zero. This method is therefore useful for users as adding control vectors initialised to $\mathbf{0}$ leaves the curve undisturbed until the control vectors are explicitly altered. We take a closer look at the case $M = 1$ to see what this idea offers and how to pick the new basis function $f(t) := f_1(t)$.

Consider a colour curve $S(t)$ as shown in Figure 2, left, and suppose for ease of exposition that it is defined for $t \in [0,1]$. We want to create a new curve that has some additional detail around $t_0 \in (0,1)$ that can be set using a control vector $\mathbf{c} = [c_x, c_r]^\top$. In particular, we want to create a new curve $\tilde{S}(t)$ such that $\tilde{S}(t_0) = S(t_0) + \mathbf{c}$, so that $\mathbf{c}$ specifies exactly the displacement from the original curve at $t = t_0$, see Figure 2, right. We can do this via Equation (1) provided that we make an appropriate choice for $f(t)$. It is clear that the treatment of colour and position should be different as the latter is not bounded while the former has to stay within the gamut $[0,1]$. Thus, Equation (1) only makes sense if we set $f$ to be a matrix of the form

$$f(t) = \begin{bmatrix} f_x(t) & 0 \\ 0 & f_r(t) \end{bmatrix}.$$

### 3.1. Refinement of geometry

The expression for the refined $x$-component is given by

$$\tilde{x}(t) = x(t) + f_x(t)c_x.$$

As is discussed in [KSD15], $f_x$ can be chosen from a wide selection of possible functions that satisfy $f_x(t_0) = 1$ and $f_x(0) = f_x(1) = 0$. In our context, the following set of conditions gives pleasing results. Let $b(\cdot)$ be any smooth basis function $b : [0,1] \to [0,1]$ that attains its only maximum at $b(t_0) = 1$ and its only two minima at

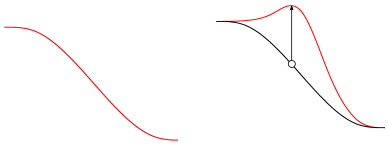

**Figure 2:** *The influence of a control vector on a colour curve S. **Left:** A univariate colour curve $S = [x(t), r(t)]^\top$. **Right:** The colour curve $\tilde{S}$ after dragging a point to a new position. The original curve is shown in black. Notice that the refinement via a control vector has local support and is interpolated.*

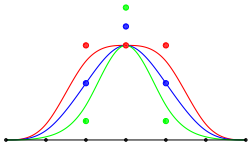

**Figure 3:** *The different cubic B-spline curves (generated by subdivision) over $\mathbb{T}$ using the non-zero control values $[1-s, 1+s/2, 1-s]$. Red: $s = 0$; blue: $s = 0.4$; green: $s = 0.8$.*

$b(0) = b(1) = 0$. Many such functions exist and we will discuss appropriate choices later on. Then the refinement of $x$ is done via

$$\tilde{x}(t) = x(t) + b(t)c_x$$

by simply using $f_x(t) = b(t)$. This small set of requirements on $b(\cdot)$ ensures that $\tilde{x}(t_0) = x(t_0) + c_x$, and that $\tilde{x}(0) = x(0)$ and $\tilde{x}(1) = x(1)$.

One of the main advantages of the control vector approach is that the refinement of position can be expressed in a local frame, as we show in Section 4.2. That is, when the user alters the original (coarse) geometry, the refinement in position is dragged along with it as it is expressed via the displacement $c_x$.

### 3.2. Refinement of colour

We now consider the $r$-component of Equation (1), i.e.,

$$\tilde{r}(t) = r(t) + f_r(t)c_r.$$

In our setting, we set the following conditions on $\tilde{r}(t)$:

1. $\tilde{r}(t_0) = r(t_0) + c_r$. This amounts to requiring that $f_r(t_0) = 1$.
2. The support of $f_r$ is $(0,1)$. This ensures that $\tilde{r}(0) = r(0)$ and $\tilde{r}(1) = r(1)$ and that the refinement is local.
3. $0 \leq \tilde{r}(t) \leq 1$. This is exactly the requirement that the newly formed colour curve does not leave the gamut.

The value of $c_r$ is constrained to lie in the interval $[-r(t_0), 1 - r(t_0)]$, so that Conditions 1 and 3 do not conflict. Unlike in the case of geometry, we cannot use $f_r(t) = b(t)$ for colour refinement. Indeed, as $b'(t_0) = 0$, we would have that $\tilde{r}'(t_0) = r'(t_0)$, which could leave the gamut in the reasonable scenario with $r'(t_0) \neq 0$ and $c_r = 1 - r(t_0)$. A suitable function $\tilde{r}(t)$ satisfying all three conditions is given by

$$\tilde{r}(t) = \begin{cases} r(t) + \frac{1-r(t)}{1-r(t_0)} b(t) c_r & \text{if } c_r \geq 0, \\ r(t) + \frac{r(t)}{r(t_0)} b(t) c_r & \text{if } c_r < 0. \end{cases} \tag{2}$$

The appearance of $1 - r(t)$ and $r(t)$ multiplying $b(t)$ is not surprising, as these functions measure the admissible amount that we can deviate from $r(t)$ in the cases $c_r \geq 0$ and $c_r < 0$, respectively.

### 3.3. Choosing the blending function

The choice of $b(t)$ is important as it lies at the heart of the refinement of both colour and position. We choose a $b(t)$ that is linked to the underlying mesh. Following [LKSD17], each knot interval is uniformly subdivided into three by a ternary step by inserting two knots into it, and these are then uniformly subdivided by a binary step, resulting in seven knots in total, see Figure 3, collected into a knot-vector $\mathbb{T}$.

It is now simple to define a function $b(t)$ satisfying the requirements from the previous section via uniform cubic subdivision, i.e., the univariate version of Catmull-Clark subdivision. The local knot vector $\mathbb{T}$ supports three uniform cubic B-splines, and their sum is exactly the $b(t)$ we are after. Expressed in terms of subdivision, we associate the middle three knots in $\mathbb{T}$ with the value of one, all other control values are set to zero; see the red graph in Figure 3. Generating $b(t)$ can, after the two initial steps, be done using binary uniform cubic subdivision with the mask $[1,4,6,4,1]/8$. As these blending functions are generated in exactly the same manner as the original colour curve, their implementation is straightforward.

The shape of $b(t)$ determines how large the effective influence of $\mathbf{c}$ is around $t = t_0$, i.e., the middle knot of $\mathbb{T}$. As $b(t)$ may be quite 'blunt' at its maximum, we offer the user control over its shape by changing the control values that generate it. Any triple of values $[1-s, 1+s/2, 1-s]$ produces a $b(t)$ that attains a maximum of 1 at the middle knot and is supported on $\mathbb{T}$. This follows from the limit stencil for uniform cubic subdivision, which reads $[1,4,1]/6$. We have to ensure that $s \in [0,1)$ to avoid several maxima or negative values. Examples are shown in Figure 3. Clearly, increasing $s$ creates a more pronounced peak so that finer detail can be added.

We remark that increasing $s$ has the effect of producing colour values that may be outside the gamut for finitely many subdivision steps, but in the limit, the gamut will be respected at all vertices. In practice, this means that colours should be projected to their limit values using the limit stencil, before rendering a pixel-dense mesh.

### 3.4. Hierarchical refinement

It is possible to extend the presented approach in a hierarchical manner, i.e., to be able to add finer detail on top of previously added detail. As binary subdivision proceeds, the six knot-intervals in $\mathbb{T}$ are split into twelve. The first or second six of these can be used to produce a basis function $b_1(t)$ in a manner completely analogous to $b_0(t) := b(t)$ to provide finer detail than could be achieved using $b_0(t)$. This can be continued as desired to produce $b_i(t)$, where $b_{i+1}(t)$ has half the support of $b_i(t)$. The final colour curve $\tilde{S}$ is then

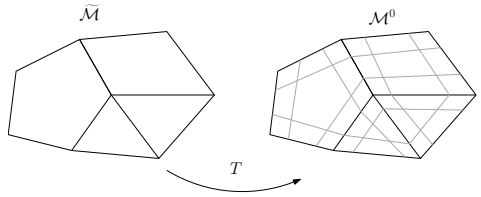


**Figure 4:** *The ternary subdivision step $T : \widetilde{\mathcal{M}} \mapsto \mathcal{M}^0$. Two new vertices are placed along each edge, and $2n+1$ new faces are created for each face of valency $n$. All newly created faces have valency 4, except for the centre face which inherits its valency from the original face.*

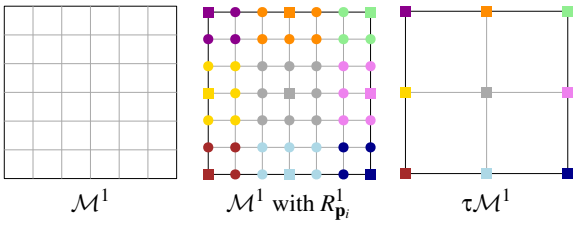


**Figure 5:** ***Left:*** *A part of $\mathcal{M}^1$ arising from a quadrilateral face in $\widetilde{\mathcal{M}}$.* ***Middle:*** *Division of the vertices into disjoint one-ring neighbourhoods: the vertices $\tau(C\widetilde{\mathcal{M}})$ are shown as coloured squares and the sets $R^1_{\mathbf{p}_i}$ are shown as disks of the appropriate colours.* ***Right:*** *The vertices $\tau(C\widetilde{\mathcal{M}})$ are connected into the mesh $\tau\mathcal{M}^1$.*

defined by the recurrence relation:

$$\begin{aligned} S_0 &:= S, \\ S_{i+1} &:= S_i + \sum_j f_{i,j}\mathbf{c}_{i,j}, \qquad \text{for } i = 0,\dots l-1, \\ \tilde{S} &:= S_l. \end{aligned}$$

Here, the $f_{i,j}$ are the basis functions at level $i$, defined as in Section 3.1 or in Section 3.2 for geometry or colour, respectively. After each step, $S_i$ is a colour curve that stays within the gamut. Thus, $\tilde{S}$ is a well defined colour curve. In this formulation, the coarse detail is added first and finer detail is added later.

We are now ready to generalise these ideas to the bivariate case of subdivision gradient meshes of arbitrary manifold topology.

## 4. Preliminaries: the bivariate case

Before presenting our two methods for local refinement of subdivision gradient meshes, we lay out the necessary terminology and mathematics common to both methods.

A mesh $\mathcal{M} = (V,E,F)$ is determined by a set of vertices $V$, a set of edges $E \subset \{(\mathbf{v}_i,\mathbf{v}_j) \in V \times V \mid i \neq j\}$, and a set of faces $F$. We assume it is planar and of manifold topology (also called connectivity or combinatorics). Additionally, for a gradient mesh, denoted $\widetilde{\mathcal{M}}$, we have a set of *colour gradient vectors* $G$. These gradient vectors are assigned to each pair $(\mathbf{v}_i,(\mathbf{v}_i,\mathbf{v}_j)) \in V \times E$. That is, for each vertex, a colour gradient vector is associated to each edge incident with that vertex. We denote by $R^m_{\mathbf{v},\mathcal{M}}$ the $m$-ring neighbourhood of a vertex $\mathbf{v}$ in the mesh $\mathcal{M}$, and we drop the subscript $\mathcal{M}$ when no confusion is likely to arise.

We outline here the method put forward in [LKSD17] to allow for arbitrary manifold topology gradient meshes. As noted earlier, a ternary subdivision step is initially performed on the gradient mesh $\widetilde{\mathcal{M}}$, which also consumes the colour gradients and produces a finer mesh, as illustrated in Figure 4. We denote this operator as $T : \widetilde{\mathcal{M}} \mapsto \mathcal{M}^0$. Then we define a sequence of increasingly finer and finer meshes $\mathcal{M}^0, \mathcal{M}^1, \dots$, where $\mathcal{M}^{i+1} = C\mathcal{M}^i$ for $i = 0,1,2,\dots$ and $C$ is the Catmull-Clark subdivision operator. We then create a colour surface over $\widetilde{\mathcal{M}}$ by computing $\mathcal{M}^l = C^l(T\widetilde{\mathcal{M}})$ and using e.g. bilinear interpolation inside all faces or subdividing until the mesh is pixel dense for rendering.

### 4.1. Topology of the involved meshes

The topology of the mesh that arises after a number of subdivision steps turns out to be important. We will make extensive use of:

**Lemma 1** Consider a topologically manifold gradient mesh $\widetilde{\mathcal{M}}$. Then the two meshes

$$\mathcal{M}^1 := CT\widetilde{\mathcal{M}} \quad \text{and} \quad TC\widetilde{\mathcal{M}}$$

have identical topology. That is, as far topology is concerned, $T$ and $C$ commute. Notice that for the ternary step we need information on the colour gradients. We are however only interested in the arising topology so here we neglect the colour gradient information.

A proof of Lemma 1 is given in the supplementary material. We denote topological equivalence via

$$\mathcal{M}^1 \overset{\text{top}}{\simeq} TC\widetilde{\mathcal{M}}.$$

Note, however, that the two operators do not commute in terms of the resultant colour surface as the $T$ operator creates flat colour spots at all vertices.

The following corollary follows immediately:

**Corollary 2** For all $k \geq 0$ and $0 \leq p \leq k$, it holds that

$$\mathcal{M}^k := C^kT\widetilde{\mathcal{M}} \overset{\text{top}}{\simeq} C^pTC^{k-p}\widetilde{\mathcal{M}}.$$

In particular, $\mathcal{M}^k \overset{\text{top}}{\simeq} TC^k\widetilde{\mathcal{M}}$.

As the topology of $\mathcal{M}^k$ is the same as that of $TC^k\widetilde{\mathcal{M}}$, there exists a natural injection $\tau$ mapping the vertices in $C^k\widetilde{\mathcal{M}}$ to the vertices in $\mathcal{M}^k$. This injection is defined by associating each vertex in $C^k\widetilde{\mathcal{M}}$ to the vertex in $\mathcal{M}^k$ created by the vertex subdivision stencil of $T$. Observe that $T$ creates disjoint one-ring neighbourhoods around each vertex in a mesh. Thus we have the following:

**Lemma 3**

$$R^1_{\mathbf{p}_i,\mathcal{M}^k} \cap R^1_{\mathbf{p}_j,\mathcal{M}^k} = \begin{cases} \emptyset & \text{if } i \neq j, \\ R^1_{\mathbf{p}_i,\mathcal{M}^k} & \text{otherwise,} \end{cases} \tag{3}$$

for all $\mathbf{p}_i$ and $\mathbf{p}_j$ in $\tau(C^k\widetilde{\mathcal{M}})$.

This is illustrated in Figure 5, and leads us to the following:

**Definition 4** For $k \geq 0$, we define the set of $R^1$*-separated vertices at level* $k$ to be the set of all vertices in $\tau(C^k\widetilde{\mathcal{M}})$. We call these

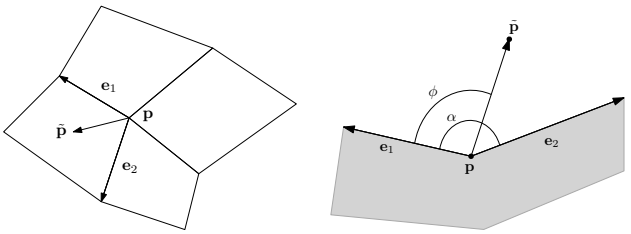


**Figure 6:** *Expressing refinement of geometry in a local frame. A vertex* $\mathbf{p}$ *is displaced to* $\tilde{\mathbf{p}}$*, lying in an incident sector defined by* $\mathbf{e}_1$ *and* $\mathbf{e}_2$*. **Left:** The edges* $\mathbf{e}_1$ *and* $\mathbf{e}_2$ *can be used to store the refinement* $\Delta\mathbf{p} = \tilde{\mathbf{p}} - \mathbf{p}$ *as* $a\mathbf{e}_1 + b\mathbf{e}_2$*. **Right:** The displacement can be locally stored as the angle fraction* $\phi/\alpha$ *and the scaled length* $\|\tilde{\mathbf{p}} - \mathbf{p}\|_2 / \sqrt{\|\mathbf{e}_1\|_2 \|\mathbf{e}_2\|_2}$ *of the displacement.*

vertices simply $R^1$*-separated vertices* when there is no confusion about the level of these vertices.

Note that the set of $R^1$-separated vertices at level 0 is the set of images of the vertices in $\widetilde{\mathcal{M}}$ under the vertex subdivision stencil of $T$. We can connect the vertices in $\tau(C^k\widetilde{\mathcal{M}})$ into a mesh by simply preserving the connectivity of $C^k\widetilde{\mathcal{M}}$. We denote the resultant mesh by $\tau\mathcal{M}^k$. Notice, crucially, that $\tau\mathcal{M}^k$ consists of a subset of the vertices in $\mathcal{M}^k = C^k T\widetilde{\mathcal{M}}$.

We have described the initial ternary step in terms of topology. Regarding geometry, it is a modified linear ternary step: the positions of the original vertices are unchanged, the new edge points play the role of colour gradient handles, and the inner face points are positioned using bilinear interpolation [LKSD17, Section 4.3].

**Editable vertices:** We must make a decision on which vertices we allow the user to edit, in terms of both position and colour, at each refinement level. That is, we must define how to construct $\mathcal{R}_k$, the set of *editable vertices* at level $k$. We let $\mathcal{R}_k$ be the set of $R^1$-separated vertices at level $k$. That is, the set of vertices in $\tau(\mathcal{M}^k)$. Consequently, at level $k \geq 0$, editable vertices have pairwise disjoint one-ring neighbourhoods by Lemma 3.

### 4.2. Local representation of geometry refinement

When a colour is assigned to a vertex (of any level), this colour is to be interpolated in the limit. Therefore it makes sense to store the colour component simply as the colour chosen by the user.

The geometry component of a vertex at level zero is simply stored in global coordinates. However, when the position of a higher-level vertex is adjusted, we would like to store it as a displacement. In this manner, translating the level 0 mesh translates the refined positions also. Furthermore, it is natural to store the displacement not in global coordinates but rather in some local frame. For this we need a method to map refinements from global coordinates to refinements in some local coordinate frame, as well as the inverse map. A suitable representation should be invariant under translation, rotation and uniform scaling of the level 0 mesh. We discuss two approaches to achieve this.

**Approach A:** For a vertex $\mathbf{p}$ of valency $n$, the parametric domain of the colour surface is logically divided into $n$ sectors, each bounded by the infinite rays given by two consecutive edges incident with $\mathbf{p}$. The refinement of geometry updates the position of $\mathbf{p}$ to a point $\tilde{\mathbf{p}} := \mathbf{p} + \Delta\mathbf{p}$, lying in one of these sectors. We denote the edges bounding this sector by $\mathbf{e}_1$ and $\mathbf{e}_2$. We can express the displacement of $\mathbf{p}$ locally via

$$\Delta\mathbf{p} = a\mathbf{e}_1 + b\mathbf{e}_2; \tag{4}$$

see Figure 6, left. By taking inner products with $\mathbf{e}_1$ and $\mathbf{e}_2$ we obtain

$$\begin{bmatrix} \Delta\mathbf{p}\cdot\mathbf{e}_1 \\ \Delta\mathbf{p}\cdot\mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} \|\mathbf{e}_1\|_2^2 & \mathbf{e}_1\cdot\mathbf{e}_2 \\ \mathbf{e}_2\cdot\mathbf{e}_1 & \|\mathbf{e}_2\|_2^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}.$$

This can readily be solved for $a$ and $b$ provided that $\mathbf{e}_1$ and $\mathbf{e}_2$ are linearly independent. We can also easily convert back from local to global coordinates by using Equation (4).

**Approach B:** We let $\mathbf{e}_1$ and $\mathbf{e}_2$ be as in Approach A. They subtend an angle $\alpha$ with each other. The vector $\Delta\mathbf{p}$ makes an angle $\phi$ with $\mathbf{e}_1$. We then store the pair

$$(\tilde{\phi}, \tilde{\rho}) := \left(\phi/\alpha, \|\Delta\mathbf{p}\|_2 / \sqrt{\|\mathbf{e}_1\|_2 \|\mathbf{e}_2\|_2}\right)$$

as the local coordinates of the displaced vertex. We can again easily invert this operation to find the global coordinates of $\Delta\mathbf{p}$. This is illustrated for a boundary vertex in Figure 6, right.

**Comparison**: When the displaced vertex $\mathbf{p}$ lies on the boundary of the mesh, Approach A is less natural than Approach B. Indeed, at the boundary, it is likely that after some subdivision steps the two outgoing boundary edges of a vertex have nearly opposite directions, making the linear system in Equation (4) numerically unstable. On the other hand, Approach A offers full affine invariance.

Both approaches require us to store an index designating the sector around the vertex where the refinement is defined, and two additional floating point numbers, so the storage cost is constant and small. This constitutes a significant improvement when compared to the exponential blow up of storage space required when storing the subdivided mesh to gain more detail. The local representations may be archived by using for instance double indices, indicating the chosen refinement level and the vertex that is to be refined.

In our implementation, we used Approach B for boundary vertices and Approach A for non-boundary vertices. It should be noted that there is no natural ordering for the sectors around a vertex, rather, the ordering is based on the particular implementation of the ternary and Catmull-Clark subdivision steps. The same holds for the choice of which bounding edge to call $\mathbf{e}_1$ and which to call $\mathbf{e}_2$. To store these refinements, a convention needs to be established which fixes the orderings.

## 5. Method 1: control vectors

The extension of the univariate technique for local refinement (Section 3) to the full bivariate setting is mostly straightforward. The colour *surface* $S$ is now of the form

$$S : D \to \mathbb{R}^2 \times [0,1]^3,$$

where $D$ is the parameter space, a connected two-dimensional subset of $\mathbb{R}^2$, and $[0,1]^3$ represents the full RGB colour space. The local knot vector $\mathbb{T}$ now becomes a local knot mesh and thus lacks

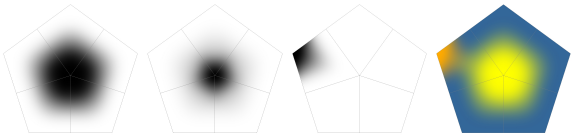

**Figure 7:** *Examples of the basis function $b(\mathbf{u})$ over a pentagonal face in $\mathcal{M}^0$. The grey lines show the mesh $\tau(\mathcal{M}^1)$.* ***Far left:*** *$s = 0$.* ***Left:*** *$s = 0.5$.* ***Right:*** *The basis function $b(\mathbf{u})$ at a boundary vertex.* ***Far right:*** *A refined colour surface.*

the natural ordering that we used in the univariate setting, but we can proceed using the same ideas as before to define the blending functions associated with control vectors.

### 5.1. Bivariate blending functions

On top of editable vertices, we need to define bivariate basis functions $b(\mathbf{u})$, again by subdivision. In other words, for refinement at a vertex $\mathbf{p} \in \mathcal{R}_k$ we must decide on a mesh $\mathcal{T}_\mathbf{p} \subset \mathcal{M}^k$ over which to define the limit function $b(\mathbf{u})$. We must also decide on a choice of control values for the vertices in $\mathcal{T}_\mathbf{p}$. We gather these values in a vector $\mathbf{b}_0$ and denote the value of a vertex $\mathbf{q} \in \mathcal{T}_\mathbf{p}$ as $\mathbf{b}_0(\mathbf{q})$.

The following setup proves convenient and natural. Following closely the univariate setting, we let $\mathbf{b}_0(\mathbf{q})$ be non-zero only for $\mathbf{q} \in R^1_\mathbf{p}$. The properties of Catmull-Clark subdivision then ensure that the resultant basis function is supported over the three-ring neighbourhood of $\mathbf{p}$. By the choice of editable vertices and by Lemma 3, we conclude that $b|_\mathbf{q} = 0$ for all $\mathbf{q} \in \mathcal{R}_k \setminus \{\mathbf{p}\}$. The small support of the basis function generated in this manner allows us to define $\mathcal{T}_\mathbf{p} = R^3_\mathbf{p}$, where $\mathbf{b}_0(\mathbf{q}) = 0$ for $\mathbf{q} \in \mathcal{T}_\mathbf{p} \setminus R^1_\mathbf{p}$. This should be understood to mean that $\mathcal{T}_\mathbf{p}$ is the mesh obtained by taking the vertices in $R^3_\mathbf{p}$ inheriting all edges and faces.

It remains to set the non-zero control values in $\mathbf{b}_0$. To allow control over the shape of $b(\mathbf{u})$, we follow the approach taken in the univariate case; see Section 3.3. In order to achieve the same effect, we first need the limit stencil of Catmull-Clark subdivision. For a vertex $\mathbf{p}$ of valency $n$ in the mesh obtained after the ternary subdivision step, its limit position $\mathbf{p}^\infty$ is given by [HKD93, Appendix A]

$$\mathbf{p}^\infty = \frac{n}{n+5}\mathbf{p} + \frac{4}{n(n+5)}\sum_{j=0}^{n-1}\mathbf{e}_j + \frac{1}{n(n+5)}\sum_{j=0}^{n-1}\mathbf{f}_j, \tag{5}$$

where $\mathbf{e}_j$ and $\mathbf{f}_j$ are the edge- and face-connected neighbours of $\mathbf{p}$, respectively. And so to ensure that $\mathbf{p}^\infty$ gets the value of 1, we set all vertices in the one-ring neighbourhood of $\mathbf{p}$ to the value

$$e = e(s) = 1 - \frac{ns}{5}.$$

And $e \geq 0$ gives us a bound on the choice of $s$, namely $s \leq \frac{5}{n}$, with default value $s = 0$. The full specification of $\mathbf{b}_0$ over $\mathcal{T}_\mathbf{p} = R^3_\mathbf{p}$ is

$$\mathbf{b}_0(\mathbf{q}) = \begin{cases} 1+s & \text{if } \mathbf{q} = \mathbf{p}, \\ 1 - \frac{ns}{5} & \text{if } \mathbf{q} \in R^1_\mathbf{p} \setminus \{\mathbf{p}\}, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 7 shows examples of the resultant basis functions.

### 5.2. Refinement of colour

In full, the blending function $f$ is now a $5 \times 5$ matrix function and $\mathbf{c}$ is a vector with five components:

$$f = \begin{bmatrix} f_x & 0 & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 & 0 \\ 0 & 0 & f_r & 0 & 0 \\ 0 & 0 & 0 & f_g & 0 \\ 0 & 0 & 0 & 0 & f_b \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_x \\ c_y \\ c_r \\ c_g \\ c_b \end{bmatrix}.$$

Armed with a basis function $b(\mathbf{u})$ defined in the previous section, we now look at the bivariate counterpart of Equation (2). Focusing on the red channel, this reads

$$\tilde{r}(\mathbf{u}) = \begin{cases} r(\mathbf{u}) + \frac{1-r(\mathbf{u})}{1-r(\mathbf{u}_0)} b(\mathbf{u}) c_r & \text{if } c_r \geq 0, \\ r(\mathbf{u}) + \frac{r(\mathbf{u})}{r(\mathbf{u}_0)} b(\mathbf{u}) c_r & \text{if } c_r < 0. \end{cases} \tag{6}$$

Analogous update relations hold for the green and blue channel. Figure 7, far right, shows an example of a refined colour surface with local colour refinements obtained by using Equation (6).

The control vector $\mathbf{c}$ must respect the bounds

$$\begin{aligned} -r(\mathbf{u}_0) \leq c_r \leq 1 - r(\mathbf{u}_0), \\ -g(\mathbf{u}_0) \leq c_g \leq 1 - g(\mathbf{u}_0), \\ -b(\mathbf{u}_0) \leq c_b \leq 1 - b(\mathbf{u}_0), \end{aligned}$$

to ensure that the limit surface does not leave the gamut. This is trivial to achieve in the user interface as the user defines the final (interpolated) colour, and not its 'displacement'. Thus, the user is never bothered with these bounds.

### 5.3. Refinement of geometry

The refinement equation for the geometry reads:

$$\tilde{\mathbf{x}}(\mathbf{u}) = \mathbf{x}(\mathbf{u}) + b(\mathbf{u})\mathbf{c}_\mathbf{x}, \tag{7}$$

where

$$\mathbf{c}_\mathbf{x} = [c_x, c_y]^\top \quad \text{and} \quad \mathbf{x}(\mathbf{u}) = [x(\mathbf{u}), y(\mathbf{u})]^\top.$$

The basis function $b(\mathbf{u})$ associated to a vertex $\mathbf{p}$ is defined through Catmull-Clark subdivision using some initial value set $\mathbf{b}_0$ defined over a mesh $\mathcal{T}_\mathbf{p}$; see Section 5.1. We define the *extension of* $\mathbf{b}_0$ *to the full mesh* $\mathcal{M}^k$, denoted by $\mathbf{p}_0^{\text{ext}}$, as

$$\mathbf{p}_0^{\text{ext}}(\mathbf{q}) = \begin{cases} \mathbf{b}_0(\mathbf{q}) & \text{if } \mathbf{q} \in \mathcal{T}_\mathbf{p}, \\ 0 & \text{otherwise.} \end{cases}$$

There are only finitely many values in $\mathbf{p}_0^{\text{ext}}$, so that we may choose some ordering of them and view $\mathbf{p}_0^{\text{ext}}$ as a vector. With this viewpoint, we obtain $\mathbf{p}_i^{\text{ext}}$ by Catmull-Clark subdivision of $\mathbf{p}_0^{\text{ext}}$ via

$$\mathbf{p}_i^{\text{ext}} = S_{k+i}\mathbf{p}_{i-1}^{\text{ext}} \qquad \text{for } i = 1, 2, \ldots, \tag{8}$$

where $S_{k+i}$ is the subdivision matrix mapping $\mathcal{M}^{k+i-1}$ to $\mathcal{M}^{k+i}$.

We can apply the same reasoning to the positions of the vertices in $\mathcal{M}^k$: we collect them in a vector $\mathbf{x}_k$ and proceed with Catmull-Clark subdivision

$$\mathbf{x}_{k+1} = S_{k+1}\mathbf{x}_k \qquad \text{for } k \geq 0.$$

The refinement matrix for $\mathbf{x}_{k+i}$ is the same as that for $\mathbf{p}_i^{\text{ext}}$ in Equation (8) as $\mathbf{x}_{k+i}$ and $\mathbf{p}_i^{\text{ext}}$ both arise from the topology of $\mathcal{M}^{k+i}$.

We can now state the following:

**Lemma 5** Consider the refinement of geometry, using Equation (7), of the intermediate mesh $\mathcal{M}^k$ through some $\mathbf{c_x}$ and the associated $\mathbf{b}_0$ at an editable vertex. Then, the resultant colour surface is the same as the colour surface obtained by altering the position of the vertices in $\mathcal{M}^k$ via

$$\mathbf{x}_k \leftarrow \mathbf{x}_k + \mathbf{p}_0^{\text{ext}}\mathbf{c}_\mathbf{x}, \tag{9}$$

and proceeding with Catmull-Clark subdivision.

That is, for the purposes of refinement of geometry, we do not need to explicitly build $b(\mathbf{u})$ but can instead alter the position values in some neighbourhood of the refinement in the intermediate mesh. Note that the result holds for the theoretical limit surface case as well as for the practical case where a finite number of subdivision steps is performed. A proof of the lemma is given in the supplementary material.

Lemma 5 offers a significant simplification of the implementation of the refinement of geometry. Instead of creating several intermediate meshes that need to be subdivided, we can simply change the geometry of $\mathcal{M}^k$ via Equation (9) and proceed with Catmull-Clark subdivision.

The result analogous to Lemma 5 does *not* hold for the refinement of colour using control vectors. This is due to the appearance of $1-r(\mathbf{u})$ (resp. $r(\mathbf{u})$) in the refinement equations. These functions are not known at intermediate refinement stages. An alternative approach is explored in Section 6.

### 5.4. Position and colour interpolation

In the initial ternary step, the colour of all editable vertices $\mathbf{v}$, whose colour has been set or adjusted by the user, is copied to all the corresponding vertices in $R^1_{\mathbf{v},\mathcal{M}^0}$ before proceeding with subdivision. The subsequent Catmull-Clark subdivision then ensures that the colour of these vertices is interpolated, as observed and exploited already in [LKSD17]. When finer detail is added at any level $k \geq 0$ via control vectors as detailed in Section 5.2, the construction of $b(\mathbf{u})$ and Equation (6) ensure that the new colour of the edited vertex is again interpolated. For the interpolation of the positions of control points set at level $k = 0$, we exploit Equation (5). The user specifies the position of $\mathbf{p}^\infty$ (corresponding to $\mathbf{p}$), and also the positions of $\mathbf{e}_j$ which are effectively the colour gradient handles associated to $\mathbf{p}$. The positions of $\mathbf{f}_j$ are determined using bilinear interpolation. To ensure that the surface interpolates $\mathbf{p}^\infty$, we internally adjust $\mathbf{p}$ via Equation (5) to

$$\mathbf{p} = \frac{n+5}{n}\mathbf{p}^\infty - \frac{4}{n^2}\sum_{j=0}^{n-1}\mathbf{e}_j - \frac{1}{n^2}\sum_{j=0}^{n-1}\mathbf{f}_j.$$

Boundary vertices are treated similarly but using the univariate limit stencil to determine the position of $\mathbf{p}$. We can use the same approach at any higher level $k > 0$. As we simply edit intermediate meshes which are not rendered and project vertices to their limit positions before rendering, the user sees the control points at their desired limit positions. Effectively, as far as the user is concerned, the subdivision scheme is interpolatory in both colour and position.

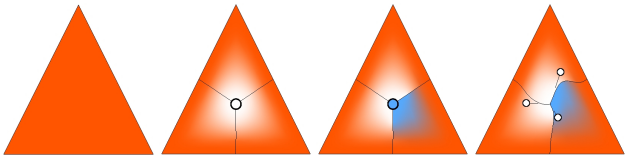

**Figure 8:** ***From the left:*** *A triangular face. Colour refinement is applied. A sharp colour refinement is added on top of the smooth colour refinement. Colour gradients can be altered at refined levels.*

## 6. Method 2: intermediate adjustment

In this section we present an alternative method of changing the emergent colour surface, this time by directly editing the colours of vertices at specific subdivision levels. We make use of the fact that colour is interpolated at a vertex if the surrounding one-ring neighbourhood of this vertex has the same colour value. The refinement of geometry stays the same as in Method 1 (Section 5), including geometry refinement representation in local frames (Section 4.2) and geometry interpolation (Section 5.4).

### 6.1. Colour refinement

We again use the main result of Section 4.1: In the mesh $\mathcal{M}^k$ there exists a subset of vertices, logically associated to the topology of $C^k\widetilde{\mathcal{M}}$, with mutually disjoint one-ring neighbourhoods of vertices.

More precisely, the colour of a vertex $\mathbf{p}$ in $\mathcal{M}^k$ is interpolated if all vertices in $R^1_\mathbf{p}$ have the same colour. Therefore, if we allow the user to assign a colour to an arbitrary number of vertices in $\tau(C^k\widetilde{\mathcal{M}})$, we can ensure interpolation of all set colours by setting the $R^1$ neighbourhoods of these to the same colour values. The condition in Equation (3) ensures that this can be done independently; see Figure 5. This amounts to applying the result obtained in Lemma 5 for geometry also to the refinement of colour.

Altering the position or colour of vertices in $\widetilde{\mathcal{M}}$ does not alter its topology, nor does it alter the topology of $\mathcal{M}^k$. As the colour refinement method just presented is concerned with (local) topology of intermittent meshes, the added detail is expressed in terms of a local frame. That is, the colour refinements are simply deformed along with any deformations in $\widetilde{\mathcal{M}}$; see Figure 1.

**Colour gradients:** We exploited the freedom to choose the colours of $R^1_\mathbf{p}$ to allow for colour interpolation at $\mathbf{p}$. Similarly, we can change the position of the points in $R^1_\mathbf{p}$ to control the extent of the propagation of colour in global space. This is exactly analogous to how the ternary step controls the propagation of colour through colour gradients. The colour gradients around a refined point $\mathbf{p} \in \tau(C^{k-1}\widetilde{\mathcal{M}})$ determine the position of the points in $R^1_\mathbf{p} \subseteq \mathcal{M}^k \setminus \tau(C^{k-1}\widetilde{\mathcal{M}})$; see Figure 8.

### 6.2. Sharp colour transitions

We can create colour surfaces having $\mathcal{C}^0$-continuous colour transitions at desired edges and vertices using the methods developed in [DKT98], by using a different set of subdivision weights for elements tagged as *sharp*. The particular set of subdivision weights

**Figure 9:** *An example of hierarchical refinement applied to a model of a leaf.* ***Far left:*** *A level 0 mesh consisting of valency 3 and valency 4 faces.* ***Left:*** *Level 1 refinement of colour.* ***Right:*** *Further refinement of colour and geometry at level 2.* ***Far right:*** *The level 2 editable mesh $\tau(\mathcal{M}^2)$. Colour refinement was performed using Method 1; see Section 5.*

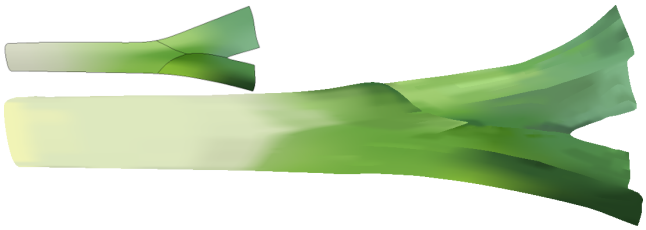

**Figure 10:** ***Left:*** *A mesh of arbitrary manifold topology that represents a leek.* ***Right:*** *Level 1 and 2 refinement of both colour and geometry has been applied to the original mesh using Method 2. Sharp colour transitions are used both in the level 0 mesh and for the refined colours, which avoids the need for layering.*

for each vertex and edge are then determined by the number of adjacent sharp edges and vertices. We can extend upon these ideas to allow $\mathcal{C}^{-1}$ colour transitions across edges.

We allow the user to assign a specific colour to a vertex $\mathbf{p}$ for each face incident with $\mathbf{p}$. Smooth vertices are then designated by vertices having the same colour for all incident faces. The subdivision rules treat edges that separate two different colours of a vertex as sharp edges, and similarly treat vertices having more than one colour as sharp, using the update rules of [DKT98] defined for sharp edges and vertices.

The updated subdivision rules only change the weights used to determine the new position and colour of vertices and do not affect the topology of the created meshes. Therefore, the methods of refinement of geometry and colour presented here still work naturally with added sharpness capabilities; see Figure 8. Sharp colour transitions are currently limited to Method 2, although it should be possible to modify Method 1 to support this feature as well.

## 7. Results and discussion

In Figure 9, we show an example of the refinement of colour and geometry at several levels using Method 1 (detailed in Section 5) on a leaf model. Figure 10 shows an example of refinement of geometry and colour using Method 2 (detailed in Section 6) on a leek model, showcasing sharp colour transitions, which were also used on the maritime flag model in Figure 11. A football model, naturally relying on pentagons and hexagons, is depicted in Figure 12.

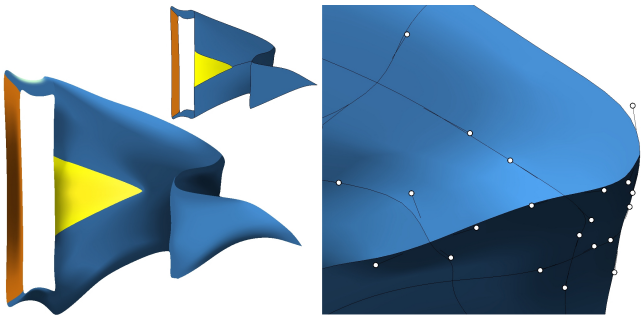

**Figure 11:** *A rendering of a maritime flag (Method 2).* ***Top:*** *The level 0 mesh, consisting of faces of valencies 3, 4, 5 and 6 as well as a hole.* ***Left:*** *Level 1 refinement is applied to add detail.* ***Right:*** *A close-up of the level 2 mesh. All level 0 capabilities (setting colour, position and gradients, and adding sharp colour transitions) are available at all refinement levels.*

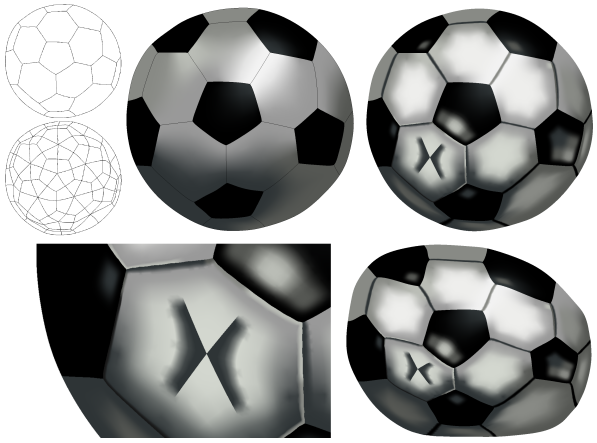

**Figure 12:** *A rendering of a football created with Method 2.* ***Top left:*** *The top mesh is the level 0 mesh used in our tool; the bottom mesh is at level 1, which would have to be used as the base mesh in traditional gradient mesh tools based on quads only.* ***Top middle:*** *The football at level 0, consisting of quads, pentagons, and hexagons.* ***Top right:*** *The final rendering after edits at levels up to level 4.* ***Bottom left:*** *An inset of the part of the football with the logo.* ***Bottom right:*** *A deflated ball modelled by editing a few vertices of level 0.*

The final model is also shown edited at the coarsest level to achieve a deflation effect. In Figure 13 we show an example of a complex design of a blackberry being locally refined. Our experimental implementation can be seen in action in the supplementary videos.

### 7.1. Comparing Method 1 and Method 2

Both of the presented refinement methods achieve the goal of allowing local hierarchical refinement of colour and geometry for subdivision gradient meshes. Additionally, both of them support interpolation of both position and colour at any level in the (subdivision) hierarchy.

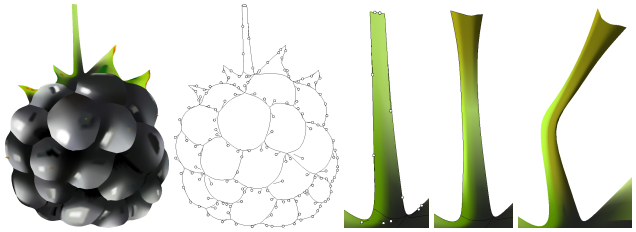

**Figure 13:** ***From the left:*** *A blackberry created in our gradient mesh tool. The arbitrary manifold topology mesh consists of faces and vertices of several different valencies. A zoom on the stem (level 0). Level 1 and 2 refinement of colour and geometry. Further level 1 refinement of geometry; level 2 details follow suit naturally.*

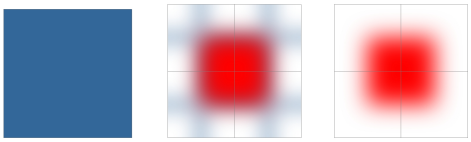

**Figure 14:** ***Left:*** *A quadrilateral gradient mesh.* ***Middle:*** *Refinement using Method 1. All editable vertices are set to white, except for the middle vertex which is set to red. The colour settings of the level 0 mesh are still visible in regions where the blending functions take on small values.* ***Right:*** *Refinement using Method 2, with the same colour settings. All colour information of the level 0 mesh is overwritten on level 1.*

Method 1, detailed in Section 5, is more general in that any basis function satisfying a small set of conditions may be used to perform the refinement. Although basis functions obtained through subdivision over the mesh are natural in some sense, other basis functions may be used based on other considerations.

Method 2, presented in Section 6, allows us to modify the colour at different refinement levels without having to store any basis functions. It should be noted that this refinement method is in some sense more invasive than Method 1, in the following way: if a user decides to explicitly set the colours of all editable points at some refinement level, effectively, the colour of all points in the mesh at that step are altered. This has the effect of overriding all colour information from previous refinement level, as well as the colours set in the coarse mesh. In contrast, Method 1 always respects the unrefined limit surface via Equation (6); see Figure 14. It is thus up to the user to decide which method best suits their needs. In terms of user interface and control, the methods are indistinguishable.

### 7.2. Improvements over previous methods

Due to the restrictions on traditional gradient mesh topologies, layering is often used to connect and overlay several gradient meshes, and it is typically non-trivial to hide these transitions. The need for such layering is greatly reduced by the increased flexibility inherent in subdivision gradient meshes. Our novel refinement methods further reduce the need for artists to use layering to circumvent the restrictions of traditional gradient mesh tools by making the primitive more flexible; see e.g. Figure 10 and Figure 15, left.

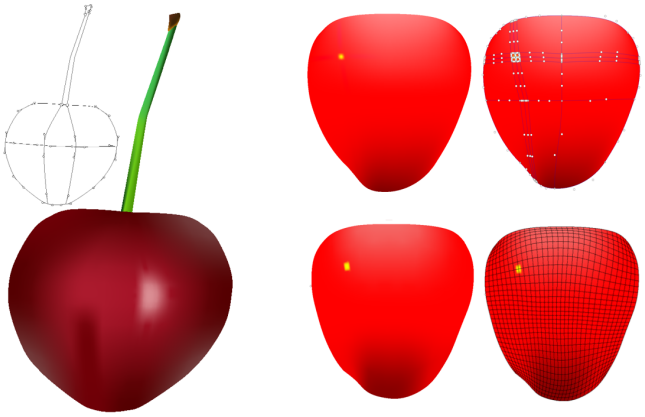

**Figure 15:** ***Left:*** *A simple cherry model, featuring sharp colour transitions. Traditional gradient mesh tools typically require modelling this using at least two meshes/layers. Our Method 2 facilitates a single-mesh representation.* ***Top middle:*** *A simple strawberry model made in Inkscape with a single yellow seed added.* ***Top right:*** *The face splitting required to add the detail in Inkscape (or any traditional gradient mesh tool).* ***Bottom middle:*** *The same model made in our tool.* ***Bottom right:*** *The level 4 editable mesh where the colour refinement was performed. The entire editable mesh is available to the user for editing, but no extra storage or computation costs are incurred until a refinement is explicitly set.*

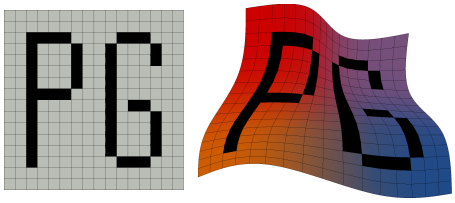

**Figure 16:** *A logo created from a single quad using Method 2. The thin black lines are the edges of the level 4 mesh.* ***Left:*** *The letters were created at level 4.* ***Right:*** *The logo has been deformed and recoloured at level 0. Note how the level 4 details follow the overall geometry adjustment and that colour refinements are kept when colours are edited at level 0 (or any other lower level than that of the details).*

Traditional gradient mesh tools do not provide a hierarchy of editable levels. Thus, after some detail is added to the mesh by splitting faces, there is no way to conveniently alter the coarse mesh aspects that would automatically drag the detail along. In contrast, the local representation of refinements of geometry at fine levels offered by our novel methods allows the user to go back to a coarser level and manipulate the mesh in a way that preserves the relative position of the refinements. This is shown in Figures 1 and 13. Another clear example of the advantages of having a hierarchy of refinement levels is showcased in Figure 16.

The fact that subdivision gradient meshes match, and go beyond, the capabilities and expressiveness of traditional gradient mesh tools, like that of Adobe Illustrator, was shown already in [KSD15]. But our methods go even further. The addition of a small detail re-

quires splitting several faces in traditional gradient mesh tools, increasing the mesh complexity. This splitting is not exact: split faces do not leave the colour surface unchanged, with the exception of the method of [BLHK18], which is, however, restricted to quadrilateral faces.

In contrast, moving to the refined mesh in our tool is exact and comes at no additional storage cost as the new vertices and connectivity are determined simply by Catmull-Clark subdivision. Refinement of the colour of an editable vertex requires the storage of only the index of the vertex, the refinement level, and the new colour. One can also easily undo such refinements (by deleting the stored triple), instead of having to merge a large set of faces back together into the original mesh. Furthermore, due to the simultaneous availability of all editable vertices at a chosen refinement level, adding several details at the same refinement level is simple and time-efficient for the user; see Figure 1. This is shown also conceptually in Figure 15, right.

### 7.3. Limitations

There are some limitations to the methods presented here. First, both methods do not work unless Catmull-Clark subdivision is performed at least until the highest refinement level at which the user altered the mesh. This means that the rendering of the colour surface is likely more expensive than when using traditional, regular rectangular, gradient meshes, which can be rendered very efficiently on the GPU. Nevertheless, even in the case of subdivision, most of the surface is still bicubic and can be rendered efficiently as such. And indeed, a commercial implementation could be expected to rely on dedicated libraries, such as opensubdiv [NLMD12], as opposed to our experimental CPU implementation.

Additionally, non-convex faces are not well suited for the methods outlined here. These give rise to colour surfaces that 'fold over'. The user should be aware of this and divide parts of the intended non-convex design into several convex faces. This is possible due to the topological freedom of our tool.

Yet another limitation is the use of a uniform subdivision scheme, which only allows for refinement at (logically) dyadic points. This has been identified already in [BLHK18]. The use of a non-uniform subdivision scheme might alleviate this.

## 8. Conclusion

We have presented two methods of local hierarchical refinement of the colour surface, one based on bivariate blending functions associated to control vectors, and the other on explicitly altering colour values at arbitrary subdivision levels. Both methods can be used at any desired refinement level and ensure interpolation of the chosen colour as well as position. Further, due to the use of control vectors, finer edits follow changes performed at coarser levels.

Subdivision gradient meshes enjoy flexibility in terms of topology. Our contribution in this area offers true hierarchical editing as well as support for sharp colour transitions, thus offering greater flexibility for vector graphics designers than offered by traditional and recent subdivision gradient meshes.

### 8.1. Future work

The results obtained in this paper may prove useful in related areas or even other fields. Here we give some examples.

**Image vectorization:** In the area of image vectorization, developments have been made that use gradient meshes as a vector primitive (see e.g. [LHM09, SLWS07, PB06]), but these methods may suffer from the same problems that designers of gradient mesh surfaces suffer from: The prohibitive costs and mesh complexity as a result of local details. Using our methods one can imagine having a level 0 layer that captures the coarse details of an image, and capturing finer details at higher refinement levels, storing them as refinements of the coarse mesh. Careful metrics will need to be established, however, in order to determine which detail needs to be stored at which refinement level.

**Time-varying topology:** In [DRVdP15], a novel data structure is introduced that supports time-continuous topological events, such as the merging and splitting of control points. Similar time-varying topological capabilities may be feasible with the refinement techniques proposed in this paper. A key property of our methods is that the emergent colour surface stays unchanged when adding zero control vectors to control points. These can then be smoothly changed with respect to time. However, as these meshes are dynamic, one needs not only adaptive refinement, but also adaptive coarsening, which presents an interesting future challenge.

**Isogeometric analysis:** Isogeometric analysis aims at integrating CAD representations of geometries and finite element type methods to numerically approximate the solutions of PDEs over these geometries. When simulating such PDEs, there is often a large discrepancy in the amount of detail needed at various points of the domain. A priori methods for determining the appropriate mesh resolution at each location in the mesh are not always available.

In [WZHS16, KLCD16] methods to allow variable resolution at different positions are presented based on truncated hierarchical Catmull-Clark (THCC) subdivision surfaces. Such methods allow a finer resolution by subdividing faces where necessary, thus creating new basis functions, and subsequently truncating these from the surrounding coarser basis functions. Thereby we only refine the mesh where it counts. Our method of hierarchical refinement may provide a viable alternative to this technique. Refining a mesh through our procedure provides locally more degrees of freedom to accurately simulate the PDE while avoiding the need for complicated and unpredictably shaped basis functions that may arise from THCC splines. For this to be successful we will require a detailed analysis to see how this affects error bounds and approximations at refined levels, as well as linear independence of added functions.

## Acknowledgements

This paper is based on the first author's MSc thesis at the University of Groningen. We would like to thank Pieter Barendrecht for his useful insights and feedback during the development of some of the methods presented in this paper. Also, we would like to thank Gert Vegter and Fred Wubs for their comments and suggestions on the original MSc manuscript. And we thank Fanna Lautenbach for her help with the leek model in Figure 10.

## References


[Ado98] ADOBE SYSTEMS: *Adobe Illustrator 8.0 Classroom in a Book*. Adobe Press, 1998. 2

[BB13] BARLA P., BOUSSEAU A.: Gradient art: Creation and vectorization. In *Image and Video-Based Artistic Stylisation*. Springer, 2013, pp. 149–166. 2

[BLHK18] BARENDRECHT P. J., LUINSTRA M., HOGERVORST J., KOSINKA J.: Locally refinable gradient meshes supporting branching and sharp colour transitions. *The Visual Computer 34*, 6 (Jun 2018), 949–960. doi:10.1007/s00371-018-1547-1. 1, 2, 10

[CC78] CATMULL E., CLARK J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (1978), 350–355. doi:10.10010-4485(78)90110-0. 1

[DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 85–94. doi:10.1145/280814.280826. 7, 8

[DRVdP15] DALSTEIN B., RONFARD R., VAN DE PANNE M.: Vector graphics animation with time-varying topology. *ACM Trans. Graph. 34*, 4 (2015), 145. doi:10.1145/2766913. 10

[FB88] FORSEY D. R., BARTELS R. H.: Hierarchical B-spline refinement. *ACM Siggraph Computer Graphics 22*, 4 (1988), 205–212. doi:10.1145/378456.378512. 2

[FSH11] FINCH M., SNYDER J., HOPPE H.: Freeform vector graphics with controlled thin-plate splines. In *ACM Trans. Graph.* (2011), vol. 30, ACM, p. 166. doi:10.1145/2024156.2024200. 2

[HKD93] HALSTEAD M., KASS M., DEROSE T.: Efficient, fair interpolation using Catmull-Clark surfaces. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 35–44. doi:10.1145/166117.166121. 6

[KLCD16] KANG H., LI X., CHEN F., DENG J.: Truncated hierarchical Loop subdivision surfaces and application in isogeometric analysis. *Comput. Math. Appl. 72*, 8 (Oct. 2016), 2041–2055. doi:10.1016/j.camwa.2016.06.045. 10

[KSD15] KOSINKA J., SABIN M. A., DODGSON N. A.: Control vectors for splines. *Computer-Aided Design 58* (2015), 173–178. Solid and Physical Modeling 2014. doi:10.1016/j.cad.2014.08.028. 2, 9

[LHFY12] LIAO Z., HOPPE H., FORSYTH D., YU Y.: A subdivision-based representation for vector image editing. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 1858–1867. doi:10.1109/TVCG.2012.76. 2

[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 (July 2009), 85:1–85:8. doi:10.1145/1531326.1531391. 10

[LKSD17] LIENG H., KOSINKA J., SHEN J., DODGSON N. A.: A colour interpolation scheme for topologically unrestricted gradient meshes. 112–121. doi:10.1111/cgf.12862. 1, 2, 3, 4, 5, 7

[Loo87] LOOP C.: Smooth subdivision surfaces based on triangles, January 1987. Department of Mathematics, The University of Utah, Master's Thesis. 2

[LTKD15] LIENG H., TASSE F., KOSINKA J., DODGSON N. A.: Shading curves: Vector-based drawing with explicit gradient control. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 228–239. doi:10.1111/cgf.12532. 2

[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. doi:10.1145/2077341.2077347. 10

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: A vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 92:1–92:8. doi:10.1145/1360612.1360691. 2

[PB06] PRICE B., BARRETT W.: Object-based vectorization for interactive image editing. *The Visual Computer 22*, 9 (Sep 2006), 661–670. doi:10.1007/s00371-006-0051-1. 10

[SL17] SVERGJA J. K., LIENG H.: A gradient mesh tool for non-rectangular gradient meshes. In *ACM SIGGRAPH 2017 Posters* (New York, NY, USA, 2017), SIGGRAPH '17, ACM, pp. 59:1–59:2. doi:10.1145/3102163.3102172. 1

[SLWS07] SUN J., LIANG L., WEN F., SHUM H.-Y.: Image vectorization using optimized gradient meshes. In *ACM Trans. Graph.* (2007), vol. 26, ACM, p. 11. doi:10.1145/1275808.1276391. 1, 10

[WZHS16] WEI X., ZHANG Y. J., HUGHES T. J., SCOTT M. A.: Extended truncated hierarchical Catmull-Clark subdivision. *Computer Methods in Applied Mechanics and Engineering 299* (2016), 316–336. doi:10.1016/j.cma.2015.10.024. 10

[XLY09] XIA T., LIAO B., YU Y.: Patch-based image vectorization with automatic curvilinear feature alignment. In *ACM Trans. Graph.* (2009), vol. 28, ACM, p. 115. doi:10.1145/1661412.1618461. 2

[ZZW14] ZHOU H., ZHENG J., WEI L.: Representing images using curvilinear feature driven subdivision surfaces. *IEEE Transactions on Image Processing 23*, 8 (2014), 3268–3280. doi:10.1109/TIP.2014.2327807. 2