Image Decomposition using Geometric Region Colour Unmixing

Mairéad Grogan
Trinity College Dublin
mgrogan@tcd.ie

Aljosa Smolic
Trinity College Dublin
smolica@scss.tcd.ie

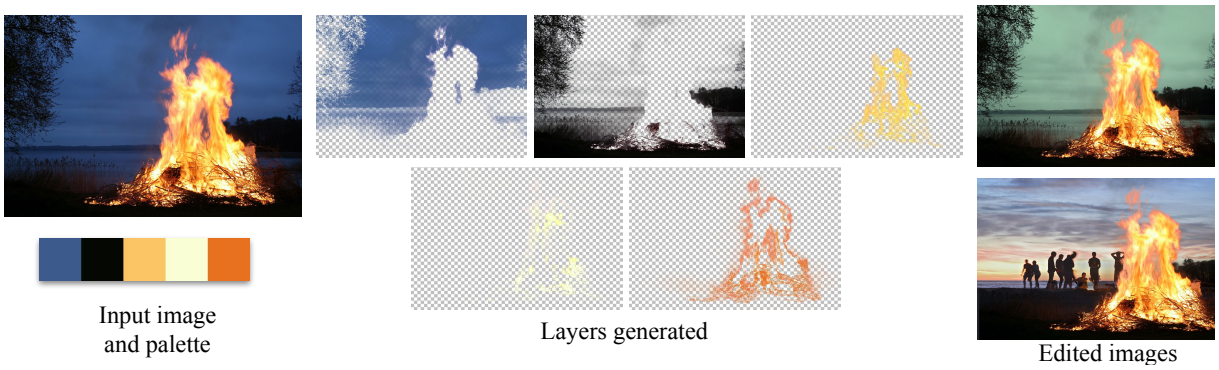


Figure 1: Left: input image with palette generated using our palette estimation step; Middle: the layers generated using our image decomposition method; Right: Some recolouring and compositing results achieved using our layers.

ABSTRACT

In this paper, we propose a new geometric approach for image decomposition which aims to combine the advantages of two state of the art techniques. Given an input image, we first compute a palette of colours from the image and use it to split the RGB space into a number of different regions. Depending on which region a given pixel lies in, different geometric methods are used to unmix the pixel's colour into a number of colours, where each colour is associated with a different layer. The layers created are smooth and homogeneous in colour, and have no reconstruction error when recombined. Our layer decomposition technique is fast to compute and the layers created can be used successfully in several applications, including layer compositing and recolouring.

CCS CONCEPTS

• **Computing methodologies** → **Image processing**; **Image manipulation**; **Image processing**.

KEYWORDS

Layer decomposition, compositing, colour unmixing, segmentation.

ACM Reference Format:
Mairéad Grogan and Aljosa Smolic. 2020. Image Decomposition using Geometric Region Colour Unmixing. In *European Conference on Visual Media Production (CVMP '20), December 7–8, 2020, Virtual Event, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3429341.3429354

CVMP '20, December 7–8, 2020, Virtual Event, United Kingdom
2020. ACM ISBN 978-1-4503-8198-7/20/12.
https://doi.org/10.1145/3429341.3429354

1 INTRODUCTION

To create complex image edits, artists often use layer based editing tools to composite or remove image layers or change the colours of objects in a scene. Layers can be created by analysing different image aspects including colour [Aksoy et al. 2017; Tan et al. 2018], shading and reflection [Bell et al. 2014], or foreground and background objects [Levin et al. 2008]. In this paper, we propose a method that decomposes an image into several semi transparent layers based on colour, with each layer associated with one of the dominant colours in the image. When recombined, layers reconstruct the input image without error, are homogeneous in colour and spatially smooth. Similar decomposition methods use per pixel non-linear optimisation, which can be very computationally demanding [Aksoy et al. 2017; Koyama and Goto 2018]. Recently, a geometric approach was proposed which unmixes a pixel based on its position in RGBXY space [Tan et al. 2018]. Even in higher dimensional spaces their geometric approach is computationally efficient, a clear advantage over non-linear optimisation approaches. However, Tan et al's method can create layers with colours that are very different from those in the input image. and introduce a reconstruction error when recombined. Taking inspiration from this recent geometric approach, we propose a new geometric strategy for layer decomposition that creates layers that are homogeneous in colour and faithful to the colours in the input image, with no reconstruction error when recombined. We also propose a new palette extraction method with colours representative of the image that lie at strategically chosen positions in RGB space to ensure high quality layers are created using our geometric decomposition. When colour changes in the input image create small discontinuities in the layers, we propose an optional postprocessing filtering step for further smoothing.

After the state of the art (Sec. 2), we describe our layer decomposition strategy (Sec. 3) followed by our palette extraction (Sec. 4)

and filtering steps (Sec. 5). Finally, we give an in depth evaluation of our method in comparison to other state of the art methods (Sec. 6).

2 STATE OF THE ART

2.1 Palette Extraction

Palette extraction involves estimating a small sample of colours that represent the colours in an image. Early works trained models on user rated palettes but were slow to compute and only generated palettes with 5 colours [Lin and Hanrahan 2013; O'Donovan et al. 2011]. Models based on Gaussian Mixture Models were also proposed, but again were slow [Shapira et al. 2009]. More recently, k-means variants were proposed [Chang et al. 2015; Phan et al. 2018; Zhang et al. 2017], but require a predefined palette size and often miss more vibrant image colours since they favour colours at the centre of the RGB space. The opposite is true of Tan et al. [2018, 2016], who use the image colours convex hull in RGB space to compute the palette. Vibrant colours dominate their palettes even though they may not be present in the input image. Their later work proposed a way to automatically estimate the optimal palette size [Tan et al. 2018]. Aksoy et al. [2017] also proposed a way to automatically detects the size of the palette, while also capturing colours similar to those in the input image. Unlike k-means based approaches, they succeed in capturing the more vibrant colours in the image. In this paper, we propose a palette extraction method based on that of Aksoy et al, and refined so that it will create high quality layers with our image decomposition strategy. We have also reduced the computation time of this step, which is quite computationally intensive.

2.2 Computing Layers

Unmixing based layer decomposition attempts to unmix pixel colours into several layers using a colour model for each layer [Aksoy et al. 2017; Lin et al. 2017; Tai et al. 2005; Tan et al. 2018]. Tai et al. [2005] alternately estimate layer colour, alpha values, and colour model parameters but their layer alpha values can have unnatural gradients. Tan et al. [2018, 2016] create layers that consist of only one colour, meaning there is no colour variation across layers. In their earlier work [Tan et al. 2016], a non-linear optimisation scheme is used to estimate layers with regularisation terms enforcing alpha sparsity and spatial coherence across layers, which is quite slow. This was extended in [Tan et al. 2018] to a purely geometric method using a pixel's location in RGBXY space to determine how it will be unmixed. This extension greatly decreases the computational complexity and ensures that spatial coherence is enforced without complex optimisations in 5D space. However, the layers estimated can still have colours that are very different to those on the input image, and suffer from reconstruction errors when combined.

In their earlier work, Aksoy et al. [2016] used a local colour model to estimate layers but layer gradients were not always smooth. This method was extended to region-based layer decomposition [Aksoy et al. 2017], with a sparsity term ensuring layers capture local regions of similar colour. Layers contain colours similar to the input image and are smoothed to remove harsh edges. However, computation is quite slow. This method was recently extended to allow for the inclusion of advanced alpha blend modes [Koyama and Goto 2018]. In this paper, we propose a layer decomposition approach that aims to combine the advantages of Tan et al. and Aksoy et al.'s most recent approaches, creating layers that are homogeneous in colour, smooth and create no reconstruction error, using a geometric unmixing approach that is much faster than non-linear optimisation approaches [Aksoy et al. 2017].

3 IMAGE DECOMPOSITION

In this section we describe how, given a palette of n colours $\{\mathbf{x}_i\}_{i=1,..n}$, $n \geq 4$, we decompose an image into n semi-transparent layers $\{L_i\}_{i=1,..n}$, with each layer having homogeneous colours similar to one of the colours in the palette. For a pixel p in the image, we will define opacity values α_i^p and layer colours $\mathbf{u}_i^p$ such that

$$\sum_i \alpha_i^p \mathbf{u}_i^p = \mathbf{c}^p \quad \forall p \tag{1}$$

where $\mathbf{c}^p$ is the colour of the pixel p in RGB space. We also assume that the input image is opaque, and impose the constraint:

$$\sum_i \alpha_i^p = 1 \quad \forall p. \tag{2}$$

Finally, we constrain the range of layer colours and alpha values:

$$\alpha_i^p, \mathbf{u}_i^p \in [0,1] \quad \forall i, p. \tag{3}$$

For convenience, we will drop the superscript p throughout the rest of the paper and present our approach at a pixel level unless specified otherwise.

Equation 1 defines the *alpha-add* representation for overlaying layers. While other overlay representations are available, similar to Aksoy et al. [2017], Tan et al. [2018] and Chen et al. [2013] we chose the *alpha-add* layer representation but it can be converted to the *overlay* [Porter and Duff 1984] representation using the method described in [Aksoy et al. 2017].

Next, we describe how we decompose an image into layers given an input palette of n colours.

3.1 Delaunay Triangulation

Given an input image and palette of n colours $\{\mathbf{x}_i\}_{i=1,..n}$, with $n \geq 4$, the Delaunay Triangulation of the palette colours $\{\mathbf{x}_i\}$ is first computed. This returns a list of n_T tetrahedra connecting the palette colours $\{\mathbf{x}_i\}$. The n_T tetrahedra have 4 vertices each and are defined by the $n_T \times 4$ matrix of vertex indices $\mathbf{T}$. This triangulation is used to define several different types of regions in RGB space. For each region, we define a method to decompose the pixel colours that lie within it, in a way that ensures layer colours and alpha values stay smooth across different regions in RGB space.

3.2 Region 1: Inside the Convex Hull

The first region that we consider is the area within the convex hull of the colours $\{\mathbf{x}_i\}$ in RGB space. All colours lying within the convex hull of $\{\mathbf{x}_i\}$ fall within one of the tetrahedra t described by $\mathbf{T}$, and for a given tetrahedron t we denote these colours as $\mathbf{c}^t$. Each tetrahedron t has four vertices $\{\mathbf{v}_i^t\}_{i=1,..4} \in \{\mathbf{x}_i\}_{i=1..n}$. For a colour $\mathbf{c}^t$ lying within a tetrahedron t, its unique normalised barycentric coordinates $\{w_i\}_{i=1,..4}$ can be computed with respect to the four vertices $\{\mathbf{v}_i^t\}_{i=1,..4}$ of t such that:

$$\mathbf{c}^t = \sum_{i=1}^{4} w_i \mathbf{v}_i^t. \tag{4}$$

These normalised barycentric coordinates satisfy the following constraints:

$$\sum_{i=1}^{4} w_i = 1, \quad w_i \in [0,1]. \tag{5}$$

For the points $\mathbf{c}^t$ we define the alpha values and layer colours as $\{\alpha_i\}_{i=1,..4} = \{w_i\}_{i=1,..4}$ and $\{\mathbf{u}_i\}_{i=1,..4} = \{\mathbf{v}_i^t\}_{i=1,..4}$, since they satisfy the layer and alpha constraints given in Equations 1-3. This decomposition is used for all colours that lie within the convex hull of $\{\mathbf{x}_i\}$ and therefore all of these colours are unmixed into 4 layers.

3.3 Region 2: Perpendicular to a Face

The second type of regions we consider are those that lie external to the convex hull of the colours $\{\mathbf{x}_i\}$, and perpendicular to one of the faces on the boundary of the Delaunay Triangulation described by $\mathbf{T}$ (see Fig. 2). Each face f on the boundary of the Delaunay Triangulation has three vertices $\{\mathbf{v}_i^f\}_{i=1,..3} \in \{\mathbf{x}_i\}_{i=1..n}$ and a normal vector $\mathbf{n}^f$, where $\mathbf{n}^f$ points in the direction away from the convex hull (Fig. 2 (a)). For each face f, we consider the region bounded by the following four planes:

$$p_1 := \mathbf{n}^f \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_1^f) = 0 \tag{6}$$

$$p_2 := \big((\mathbf{v}_1^f + \mathbf{n}^f) \times (\mathbf{v}_3^f - \mathbf{v}_1^f)\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_1^f) = 0 \tag{7}$$

$$p_3 := \big((\mathbf{v}_2^f + \mathbf{n}^f) \times (\mathbf{v}_1^f - \mathbf{v}_2^f)\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_2^f) = 0 \tag{8}$$

$$p_4 := \big((\mathbf{v}_3^f + \mathbf{n}^f) \times (\mathbf{v}_2^f - \mathbf{v}_3^f)\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_3^f) = 0 \tag{9}$$

where $\overrightarrow{\mathbf{x}} = (x, y, z)$, the variable used to represent any point on the given plane. The first plane p_1 contains the face f, while the other three planes are orthogonal to f and contain one of its edges (see Fig. 2 (b) and (c)). Simple planar checks are used to find all of the colours $\{\mathbf{c}^f\}$ that lie between these four planes.

For every colour $\mathbf{c}^f$ that lies orthogonal to the boundary face f, the following equation is used to project $\mathbf{c}^f$ onto f (Fig. 3 (a)):

$$\hat{\mathbf{c}}^f = \mathbf{c}^f - \frac{(\mathbf{c}^f - \mathbf{v}_1^f) \cdot \mathbf{n}^f}{\mathbf{n}^f \cdot \mathbf{n}^f} \mathbf{n}^f \tag{10}$$

The projected point $\hat{\mathbf{c}}^f$ now lies on the face f and since $\hat{\mathbf{c}}^f$ and the vertices $\{\mathbf{v}_i^f\}_{i=1,..3}$ are coplanar, unique normalised barycentric coordinates $\{w_i\}_{i=1,..3}$ can be computed for the point $\hat{\mathbf{c}}^f$ with respect to the vertices $\{\mathbf{v}_i^f\}_{i=1,..3}$, as in Fig. 3 (b). These coordinates satisfy the equation:

$$\hat{\mathbf{c}}^f = \sum_{i=1}^{3} w_i \mathbf{v}_i^f. \tag{11}$$

We then translate $\hat{\mathbf{c}}^f$ in the direction of n^f as follows:

$$\mathbf{c}^f = \hat{\mathbf{c}}^f + d\mathbf{n}^f \tag{12}$$

where

$$d = \left\| \frac{(\mathbf{c}^f - \mathbf{v}_1^f) \cdot \mathbf{n}^f}{\mathbf{n}^f \cdot \mathbf{n}^f} \right\|. \tag{13}$$

In the same way, we can translate the vertices $\{\mathbf{v}_i^f\}_{i=1,..3}$ to:

$$\bar{\mathbf{v}}_i^f = \mathbf{v}_i^f + d\mathbf{n}^f \quad \forall i \tag{14}$$

as shown in Fig. 3 (c). Then the following equation holds:

$$\mathbf{c}^f = \sum_{i=1}^{3} w^i \bar{\mathbf{v}}_i^f. \tag{15}$$

Therefore, for the points $\mathbf{c}^f$ that lie perpendicular to face f, we can define the alpha values and layer colours as $\{\alpha_i\}_{i=1,..3} = \{w_i\}_{i=1,..3}$ and $\{\mathbf{u}_i\}_{i=1,..3} = \{\bar{\mathbf{v}}_i^f\}_{i=1,..3}$. All of the colours in these regions are then unmixed into 3 layers.

3.4 Region 3: Above Edges

The third type of region we consider are the regions in RGB space that lie above an edge on the boundary of the Delaunay triangulation. Any edge on this boundary connects two faces of the triangulation, for example $e = [\mathbf{v}_1\ \mathbf{v}_2]$ is the edge connecting boundary faces f_1 and f_2 in Fig. 4. For the edge e, the region we consider is bounded by four planes as follows:

$$p_1 := \big((\mathbf{n}^{f_1}) \times (\mathbf{v}_2^f - \mathbf{v}_1^f)\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_1^f) = 0 \tag{16}$$

$$p_2 := \big((\mathbf{n}^{f_2}) \times (\mathbf{v}_2^f - \mathbf{v}_1^f)\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_1^f) = 0 \tag{17}$$

$$p_3 := \big((\mathbf{n}^{f_1}) \times (\mathbf{n}_2^f)\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_1^f) = 0 \tag{18}$$

$$p_4 := \big((\mathbf{n}^{f_1}) \times (\mathbf{n}^{f_2})\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_2^f) = 0 \tag{19}$$

Planes p_1 and p_2 contain the edge e while planes p_3 and p_4 are perpendicular to e, and are highlighted in Fig. 4 (a) and (b). We let $\mathbf{c}^e$ denote the colours that lie in this region in RGB space (4 (c)). To decompose the colours $\mathbf{c}^e$ we use the colour line assumption [Ruzon and Tomasi 2000] which states that the unmixed layer colours for a pixel and the pixel colour itself should lie on the same line in RGB space. Using a technique similar to that proposed by [Aksoy et al. 2017] in their projected colour unmixing step, for a given colour $\mathbf{c}^e$, we project it onto the planes p_3 and p_4 (Eqs. 18 and 19) as follows:

$$\{\hat{\mathbf{c}}^e_{\{1,2\}}\} = \mathbf{c}^e - \frac{(\mathbf{c}^e - \mathbf{v}_{\{1,2\}}) \cdot \mathbf{n}^p}{\mathbf{n}^p \cdot \mathbf{n}^p} \mathbf{n}^p \tag{20}$$

where $\mathbf{n}^p = (\mathbf{v}_1 - \mathbf{v}_2)/\|\mathbf{v}_1 - \mathbf{v}_2\|$ is the normal vector of planes p_3 and p_4. The colour value $\mathbf{c}^e$ then lies on the line connecting $\hat{\mathbf{c}}_1^e$ and $\hat{\mathbf{c}}_2^e$, as shown in Fig. 4 (d). Scalars w_1 and w_2 can be computed as:

$$w_1 = \frac{\|\mathbf{c}^e - \hat{\mathbf{c}}_2^e\|}{\|\hat{\mathbf{c}}_1^e - \hat{\mathbf{c}}_2^e\|}, \quad w_2 = 1 - w_1. \tag{21}$$

and satisfy the following equation:

$$\mathbf{c}^e = w_1 \hat{\mathbf{c}}_1^e + w_2 \hat{\mathbf{c}}_2^e. \tag{22}$$

Therefore we can define the alpha values and layer colours for $\mathbf{c}^e$ as $\alpha_{\{1,2\}} = w_{\{1,2\}}$ and $\mathbf{u}_{\{1,2\}} = \hat{\mathbf{c}}^e_{\{1,2\}}$. The same procedure is followed for all of the edges connecting two boundary faces of the triangulation. All of the colours in these regions are then unmixed into 2 layers.

3.5 Region 4: Near Points

The remaining colours in RGB space are associated with only one palette colour $\mathbf{x}_i$ lying on the convex hull of the Delaunay triangulation. A given palette colour $\mathbf{x}_i$ on the convex hull is connected to three faces, for example $\mathbf{x}_i = \mathbf{v}_2$ in Fig. 5 is connected to f_1, f_2 and f_3. Here, for the colour $\mathbf{x}_i = \mathbf{v}_2$ we define three planes bounding the region of interest as follows :

$$p_1 := \big((\mathbf{n}^{f_1}) \times (\mathbf{n}^{f_2})\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_2^f) = 0 \tag{23}$$

$$p_2 := \big((\mathbf{n}^{f_1}) \times (\mathbf{n}^{f_3})\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_2^f) = 0 \tag{24}$$

$$p_3 := \big((\mathbf{n}^{f_3}) \times (\mathbf{n}^{f_2})\big) \cdot (\overrightarrow{\mathbf{x}} - \mathbf{v}_2^f) = 0 \tag{25}$$

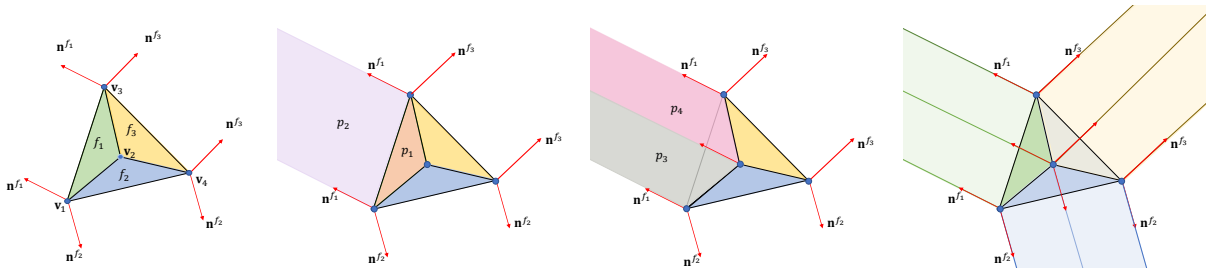


Figure 2: For simplicity, here we consider a palette with 4 colours only. (a) shows the tetrahedon t created by the Delaunay triangulation with vertices $\{\mathrm{v}_i\} = \{\mathrm{x}_i\}$ and boundary faces f_1, f_2, f_3, f_4 with normal vectors n^{f_i}. For the face f_1, (b) and (c) show the planes p_1, p_2, p_3 and p_4 that bound the region perpendicular to f_1. In (d), the regions we consider for faces f_1, f_2 and f_3 are marked in green yellow and blue.

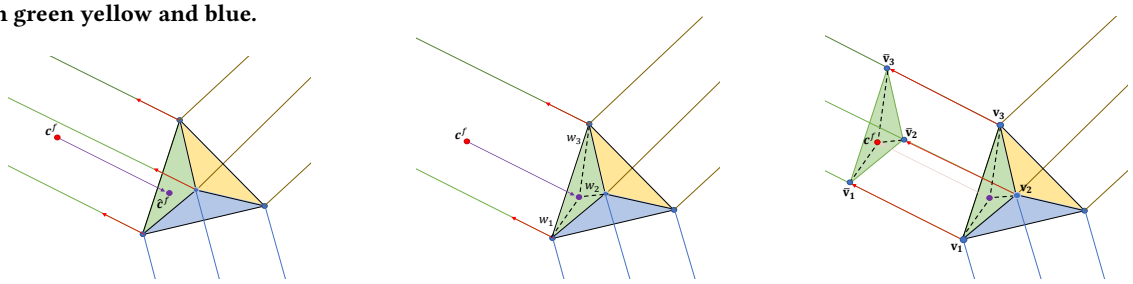


Figure 3: For a colour c^f that lies in the region perpendicular to f_1, (a) $\hat{\mathrm{c}}^f$ is the projection of c^f onto f_1. (b) Barycentric coordinates $\{w_i\}$ are then computed for $\hat{\mathrm{c}}^f$ with respect to $\{\mathrm{v}_i\}$. (c) The vertices $\{\mathrm{v}_i\}$ are translated to $\{\bar{\mathrm{v}}_i\}$ and the barycentric coordinates $\{w_i\}$ now relate to c^f and $\{\bar{\mathrm{v}}_i\}$.

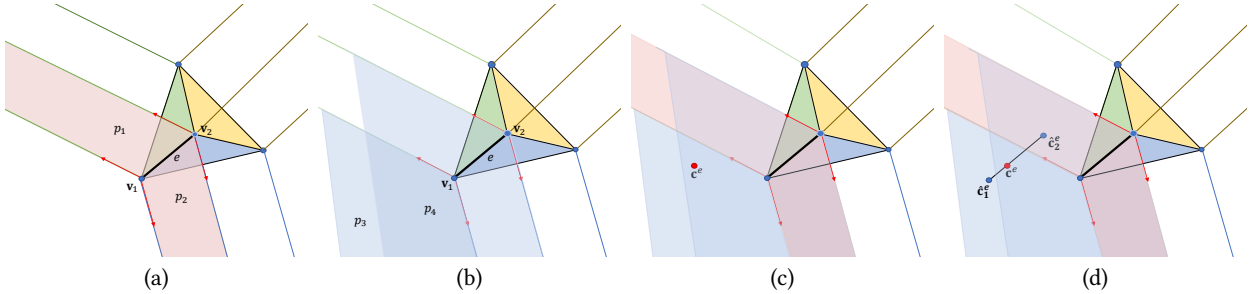


Figure 4: For the edge $e = [\mathrm{v}_1\mathrm{v}_2]$, (a) highlights the planes p_1 and p_2, and (b) highlights p_3 and p_4. (c) depicts the region bounded by these planes and any colours here are denoted c^e. In (d), the projection of c^e onto p_3 and p_4 create $\hat{\mathrm{c}}_1^e$ and $\hat{\mathrm{c}}_2^e$.

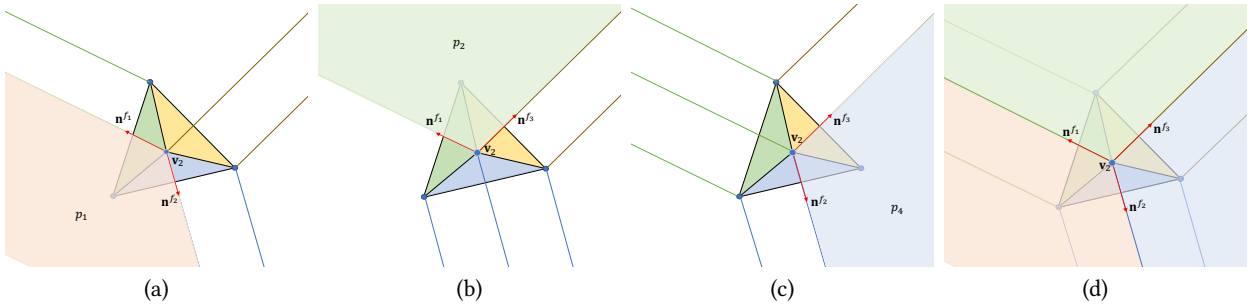


Figure 5: For the vertex v_2, (a) highlights the plane p_1, (b) highlight p_2, (c) highlights p_3, and (d) the region bounded by these planes.

These planes are highlighted in Figure 5. For all colours $\mathbf{c}^{x_i}$ that lie in this region, we define their alpha and colour values as $\alpha_1 = 1$ and $\mathbf{u}_1 = \mathbf{x}_i$. Therefore these colours remain fully opaque.

3.6 Overlap and Consistency between regions

Colours that lie on the boundary of two different regions can belong to either region type, but our use of barycentric coordinates when decomposing pixels ensures consistent pixel unmixing across region boundaries. For example, colours that lie on a face on the boundary of the Delaunay Triangulation can be considered within the convex hull of the palette colours (as in Section 3.2) as well as the region perpendicular to the face it lies on (as in Section 3.3). But both of these decomposition methods will result in the pixel being unmixed into 3 colours since only three non-zero barycentric coordinates exist for points lying on the face of a tetrahedron in 3D. This consistency across region boundaries ensures that there are no discontinuities in the decomposition of colours that lie close together but across different types of regions in RGB space.

3.7 Edges of the RGB cube

In some cases, when the pixel colour $\mathbf{c}^p$ lies near the edge of the RGB cube, equations 14 and 20 will create layer colours lying outside RGB space. Therefore, we add an additional step to overcome this issue. For estimated layer colours $\{\mathbf{u}_i\}_{i=1..m}$, when some $\mathbf{u}_i$ lie outside of the RGB cube we first crop the values of $\mathbf{u}_i$ above 1 or below 0 to 1 and 0. If $\mathbf{c}^p$ lies on the new line/triangle created by the cropped $\{\mathbf{u}_i\}$, we unmix $\mathbf{c}^p$ (see Fig. 6 (a)). Otherwise, we translate the scaled $\{\mathbf{u}_i\}$ so that the line/triangle formed by them contains the pixel colour $\mathbf{c}^p$. If any colour $\mathbf{u}_j$ is mapped outside of the RGB cube during this step, we refine $\mathbf{u}_j$ so that it becomes the point of intersection between the RGB cube boundary and the vector $\overrightarrow{\mathbf{c}^p\mathbf{u}_j}$. This ensures that the new line/triangle will still contain $\mathbf{c}^p$ (see Fig. 6 (b)). Finally, if $\{\mathbf{u}_i\}$ form a triangle, the alpha values $\{\alpha_i\}_{i=1..3}$ are the barycentric coordinates of $\mathbf{c}^p$ with respect to $\{\mathbf{u}_i\}$. If $\{\mathbf{u}_i\}$ form a line, equation 21 is used to compute $\{\alpha_i\}_{i=1,2}$.

We also found that in order to successfully unmix colours that lie on the boundary of the RGB cube using lines/triangles (Fig. 6 (c)), the lines/triangles must contain the colour $\mathbf{c}^p$ and be tangent to the boundary of the RGB space. Therefore, for colours $\mathbf{c}^p$ that lie very close to the RGB cube boundary (with values > 0.98 or < 0.02), rather than scaling out of range values to 1 and 0, we project those vertices $\{\mathbf{u}_i\}$ that lie close to $\mathbf{c}^p$ onto the plane parallel to the RGB boundary containing $\mathbf{c}^p$ (see Fig. 6 (c)). This ensures that these colours are unmixed successfully.

4 PALETTE ESTIMATION

When computing a suitable palette of colours for our layer decomposition, we began by testing Aksoy et al. [2017] palette extraction method since it both computes a suitable palette size and extracts good palette colours. While we found that in some cases our decomposition technique created nice layers with Aksoy et al's palette, in others, layers displayed blocky artifacts with sharp discontinuities in alpha values (see Fig. 7). This was due to the relationship between the position of the palette colours in RGB space and the types of tetrahedra that are created by the Delaunay triangulation step. Tetrahedra that are too flat, too large or too small can create undesirable changes in alpha values across layers (see Section 4.2 for more detail). We therefore split our palette estimation into two steps - a palette extraction step based on [Aksoy et al. 2017], followed by a palette refinement step, both of which are described below.

4.1 Initial Palette

When estimating a palette, Aksoy et al. [2017] associate each pixel p with a representation score r^p, where r^p indicates how well a pixel is represented by the colours in the palette (see [Aksoy et al. 2017] for details computing r^p). They then use a greedy iterative scheme to add colours to the palette until the majority of pixels in the image are well represented. To compute the representation score r^p, each palette colour is represented by a normal distribution with a unique covariance matrix. We follow the same procedure when computing our palette, but instead assign the same variance $0.0001 * I$ to each colour $\mathbf{x}^i$. This computes a larger palette of colours from the image which can be refined to ensure optimal results from our geometric layer decomposition. The elimination of unique covariance matrices also greatly reduces the computational cost of this step.

4.2 Palette Refinement

4.2.1 Removing nearby colours. We found that when palette colours lie too close together in RGB space, very small tetrahedra are created. This causes alpha values across layers to change too quickly from transparent to opaque, creating blocky edges in layers as seen in Fig 7). To avoid this, starting from the first palette colour, we remove any palette colours that are too close to it (within a distance $d = 0.28$ in RGB space, where all colour values are in the range $[0, 1]$). By starting with the first palette colour, we ensure that the most prevalent colours in the image take highest priority.

4.2.2 Removing flat tetrahedra. Flat tetrahedra with vertices that are practically coplanar can result in large discontinuities in alpha values across layers. Therefore, if three colinear palette colours are found (or nearly colinear, up to a given tolerance), we remove the middle colour along the line. This middle palette colour is somewhat redundant in any case since the colour line assumption [Ruzon and Tomasi 2000] states that if a colour is to be unmixed into two colours, it should be colinear to those two colours. Since the middle colour is colinear to two other palette colours, it can be easily unmixed using them and does not need to be represented in the palette. If, after this first step, a tetrahedron is still almost flat, we apply a small translation to one of its palette colours in the direction of the tetrahedron normal.

4.2.3 Dark/Bright colours. We found that if too many palette colours lie close to black/white, blocky artifacts such as those in Fig. 7 can be created, and so we add a constraint that ensures only one palette colour lies in the regions near white/black.

In Fig. 8 we show some palettes before and after refinement. We can see that these refinement steps do not remove any important colours from the palette, only those very similar to others. This step also ensures that the selected palette colours are strategically placed in RGB space for our layer decomposition method.

5 LAYER FILTERING

Although our layer colours change smoothly across the image, small spatial inconsistencies can be created by compression artifacts or

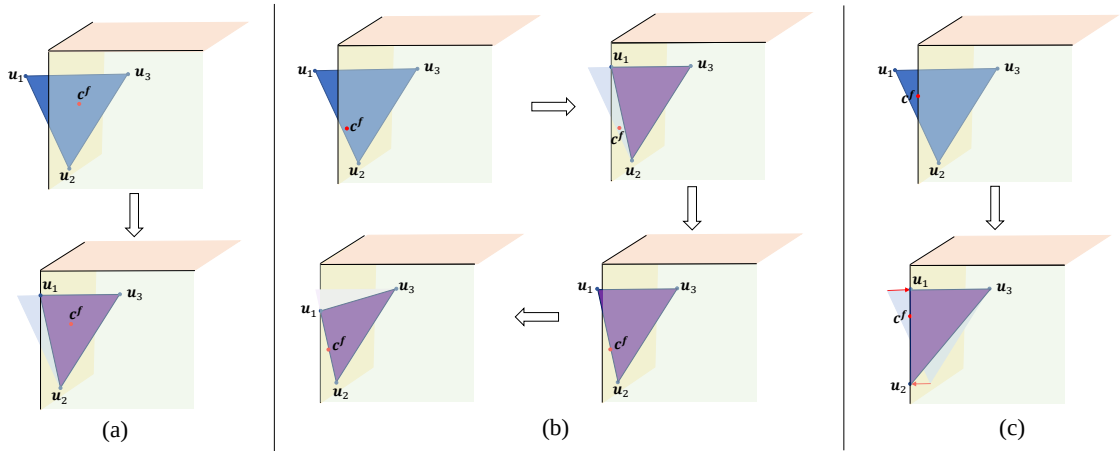


Figure 6: Steps followed when a pixel colour $\mathbf{c}^f$ lies close to the RGB cube boundary. (a) Any layer colours $\{\mathbf{u}_i\}$ outside the RGB cube are scaled to lie inside it. If $\mathbf{c}^f$ now lies on the new triangle, $\mathbf{c}^f$ is unmixed by computing its barycentric coordinates with respect to the new $\{\mathbf{u}_i\}$. (b) If $\mathbf{c}^f$ does not lie on the triangle after scaling, $\{u_i\}$ are translated so that they contain $\mathbf{c}^f$. If the translated points u_i are mapped outside of the RGB cube, they are shifted so that they are within it.(c) If the colour $\mathbf{c}^f$ is on the RGB boundary, nearby $\{\mathbf{u}_i\}$ are first translated so that they lie on the same plane as $\mathbf{c}^f$.

Figure 7: Rows 1-2: Input image and layers generated using our layer decomposition. The palette was extracted using the method in [Aksoy et al. 2017]. Row 3: Close up of layers show blocky discontinuities. Close ups are on white backgrounds for clarity.

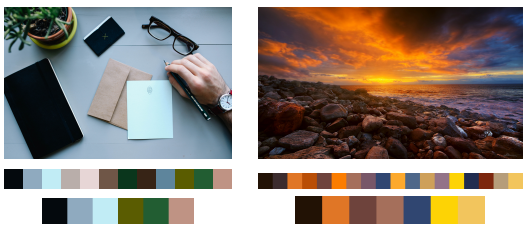


Figure 8: Images with palettes generated by our palette estimation method before (top) and after refinement (bottom).

colour variation along edges. To combat these spatial inconsistencies, we present an optional post-processing filtering step to smooth the alpha values of the layers. Similar to previous methods [Aksoy et al. 2017; Koyama and Goto 2018], we first apply a guided image filter to the alpha channel of each layer. Since filtering can cause alpha values of zero to become non-zero, some pixels may now have more than four non-zero alpha values across the layers. In this case, we keep only the four largest non-zero alpha values and discard the rest. We then normalise the sum of the alpha values for each pixel to ensure they sum to unity.

While previous methods proposed an optimisation framework to refine layer colours at this point [Aksoy et al. 2017], this can be very computationally intensive. To avoid this and maintain the advantages of our geometric approach, we instead follow the procedure outlined in Algorithm 1 to refine the colours of each layer given the new filtered alpha values. For each pixel, we iteratively recompute each of the layer colours using the remaining alpha values and layer colours, while ensuring that the new layer colours do not differ too much from the original layer colours before filtering. The alpha values are then recomputed given the new layer colours. We found that this process creates layers that are very similar in structure to the layers created by our initial decomposition step, but with any discontinuities created by quantisation artifacts or colour changes along edges removed (see Fig. 9).

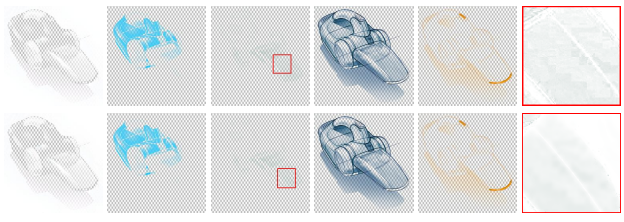


Figure 9: Our layers before (top) and after filtering (bottom). A close up of one of the layers is shown in red on the right. Zoom in to see quantisation artifacts.

For each pixel p in the image ;
input : Pixel p with colour $\mathbf{c}$
m non zero filtered alpha values $\{\tilde{\alpha}_i\}_{i=1..m}$
m layer colours $\{\mathbf{u}_i\}_{i=1..m}$
output : m refined alpha values $\{\alpha_i\}_{i=1..m}$
m refined layer colours $\{\mathbf{u}_i\}_{i=1..m}$
if $m = 1$ **then**
  $\alpha_1 = \tilde{\alpha}_1 = 1$;
  $\mathbf{u}_1 = \mathbf{c}$;
end
if $m = 2, 3, 4$ **then**
  Define $\{\tilde{\mathbf{u}}_i\}_{i=1..m} = \{\mathbf{u}_i\}_{i=1..m}$;
  /* Compute new colours using $\{\tilde{\alpha}_i\}$ */
  for $i = 1 : m$ **do**
    $\hat{\mathbf{u}}_i = \dfrac{\mathbf{c} - \sum_{j \neq i} \tilde{\alpha}_j \tilde{\mathbf{u}}_j}{\tilde{\alpha}_i}$;
    /* Ensure $\tilde{\mathbf{u}}_i$ not far from $\mathbf{u}_i$ */
    $\tilde{\mathbf{u}}_i = refine(\hat{\mathbf{u}}_i, \mathbf{u}_i)$;
  end
  if $m = 3, 4$ **then**
    /* Get bary. coord. of $\mathbf{c}$ wrt $\tilde{\mathbf{u}}_i$ */
    $\{w_i\}_{i=1..m} = computeBary(\mathbf{c}, \{\tilde{\mathbf{u}}_i\}_{i=1..m})$;
    $\{\alpha_i\}_{i=1..m} := \{w_i\}_{i=1..m}$;
    $\{\mathbf{u}_i\}_{i=1..m} := \{\tilde{\mathbf{u}}_i\}_{i=1..m}$;
  end
  if $m = 2$ **then**
    /* Get weights as in Eq 21 */
    $\{w_i\}_{i=1..m} = computeLinear(\mathbf{c}, \{\tilde{\mathbf{u}}_i\}_{i=1..m})$;
    $\{\alpha_i\}_{i=1..m} := \{w_i\}_{i=1..m}$;
    $\{\mathbf{u}_i\}_{i=1..m} := \{\tilde{\mathbf{u}}_i\}_{i=1..m}$;
  end
end

Algorithm 1: Method used to compute the new alpha values and layer colours for a pixel p taking as input the filtered alpha values and layer colours as computed in Section 3. Here, *refine*() is a function ensuring that the new layer colour $\tilde{\mathbf{u}}_i$ does not differ much (less than distance of 0.03) from the previously estimated colour $\mathbf{u}_i$.

6 RESULTS AND EVALUATION

6.1 Qualitative Assesment

6.1.1 Palette Estimation. In Fig. 10, we compare our palettes to those created by other state of the art techniques [Aksoy et al. 2017; Chang et al. 2015; Tan et al. 2018]. We found that our palettes are well aligned with those created by Aksoy et al. [2017][1], although they can differ in size (the hexagonal palette representation for Aksoy et al. indicates the colour variation estimated for each palette colour, as mentioned in Sec. 4.1). In contrast to our approach, the palettes generated by Tan et al. [2018] contain colours that are very different to those in the input images, such as the bright red that appears in the second image's palette. Similar to other k-means approaches, Chang et al. [2015] require the palette size n to be defined by the user, while our approach, similar to Aksoy et al. and Tan et al., estimates n automatically. It is also clear that Chang et al's k-means method captures 'duller' colours since the clustering energy favours colours lying further within the RGB cube (see the first and third image in Fig. 10). The voting scheme proposed by Aksoy et al. [2017] and implemented in our approach ensures that more vivid image colours are captured by our palette.

6.1.2 Layer Decomposition. In Fig. 16 we compare our layers to those of Tan et al. [2018] and Aksoy et al. [2017]. We present our layers without any filtering (Sec. 5) to highlight the smoothness of our layers in general. We found that our layers display similar qualities to both Tan et al. and Aksoy et al. When images contain colours that are vibrant and near the edge of the RGB cube, all three methods create similar layers, although Tan et al's layer/palette colours can still differ from those in the input image (see the red layer in Fig. 16, example 4). Our layers are similar in colour to those of Aksoy et al., with layer colours that are more faithful to the colours in the image (see Tan et al's bright red and yellow layers in Fig. 16 example 1, in comparison to Aksoy et al. and our orange layers). Our layers also display the same colour variation as those proposed by Aksoy et al, even though we do not explicitly estimate colour variances when estimating the colour palette (see Fig. 16 example 2, highlighted in blue). In fact, we found that our colour variation was smaller than that of Aksoy et al. (see Sec. 6.2). To highlight this, in Fig. 11 we present the colour values of three of the layers presented in example 2 of Fig. 16 without the alpha channel, highlighting that the colours captured by both methods are similar but with less variation in our layers. In comparison to Tan et al., both methods create colours more faithful to those in the original image.

We also found that layers by Aksoy et al. are more region based due to their sparsity constraint. When a palette colour appears in only a small portion of the image, only this section of the image will appear in their layer (see Fig. 16, example 3, circled in green). Our method is more global in nature and similar to that of Tan et al, pixels across the image are unmixed onto this red layer. We also applied our layer decomposition to a selection of digital paintings and again found that our method created smooth, homogeneous in colour layers (see example in Fig. 9). More layer results before and after filtering can be found in the supplementary material.

6.2 Quantitative assessment

6.2.1 Layer Analysis. Next we evaluate the three methods using three metrics proposed by Aksoy et al. [2017] to assess layer quality.

(1) Reconstruction error: Assesses the reconstruction error of the layers. The mean squared error between the input image and the sum of the alpha weighted colour layers is computed.
(2) Gradient Correlation: Assesses whether the texture content of the layers is similar to that of the input image. The correlation coefficient between the gradient of every colour channel of the input image with the gradient of the alpha channel of every layer is computed and averaged.
(3) Colour Variance: Assesses the colour homogeneity of layers. It returns the sum of the individual variances of RGB channels averaged over all layers of the input image.

We computed and averaged these metrics for 60 images using our methods, before and after filtering, and those of Tan et al. [2018] and Aksoy et al. [2017], with the results in Table 1. Colour and

[1] All results presented in this paper for [Aksoy et al. 2017] were provided by the authors. Our own implementation of [Aksoy et al. 2017] is also available at https://github.com/V-Sense/soft_segmentation

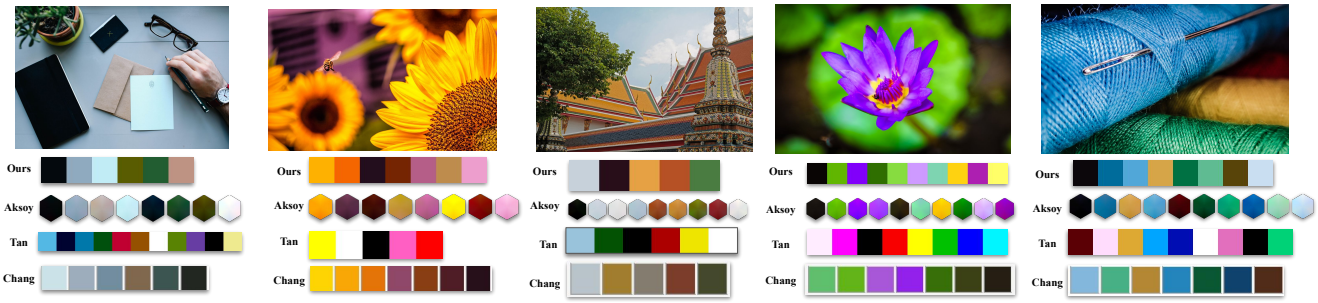


Figure 10: Palettes generated by our approach, Aksoy et al. [2017], Tan et al. [2018] and Chang et al. [2015].

Figure 11: Colours of the layers created for an image using Aksoy et al's method (top) and our method (bottom). Pixels with an alpha value of zeros are shown in black since their colours do not matter. The variance of colours in the layers created using our approach is very similar, if not more compact, than those created using Aksoy et al's method.

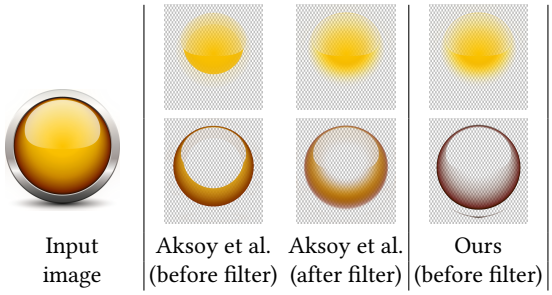


Figure 12: Here, we show the reliance of Aksoy et al. [2017] on heavy filtering to remove strong changes in the alpha channels of their initial layer estimation step. In contrast, our layers are smooth before filtering is applied.

alpha values were in the range $[0, 1]$. In terms of reconstruction error, Tan et al's layers have an average reconstruction error of 0.015, while our method and that of Aksoy et al. have errors below the quantisation limit. In terms of colour variance, our layers have a lower variance than those of Aksoy et al., as shown in Fig. 11. Since Tan et al's layers are all the same colour, their variance is zero. Finally, our layers score higher than Tan et al. in terms of gradient correlation both before and after filtering, and Aksoy et al. performs the best. While Aksoy et al's layers are very smooth, this is due to their heavy reliance on a filtering step to remove the strong alpha changes that often occur in their layers during the initial estimation step (see Fig. 12). In contrast, the layers generated by our approach before filtering, have alpha values that change smoothly across the image, with only small discontinuities appearing in some cases.

Table 1: Here we present the results of each method with respect to the three metrics used to assess layer quality.

	Tan et al.	Aksoy et al.	Ours (no filter)	Ours (with filter)
Rec. Error	0.0149	*0.0005*	*0.0019*	*0.0027*
Grad. Corr.	0.66	0.70	0.67	0.69
Col. Var.	0	0.005	0.0006	0.0007

6.2.2 Speed. Next, we compare the computation time of our method to [Aksoy et al. 2017; Tan et al. 2018]. Both ours and [Tan et al. 2018] were implemented on an Alienware Aurora-A5 PC with 4GHz Intel Core i7-6700K CPU and 16GB RAM. Tan et al's implementation is the non-parallelised Python version made available by the authors. Aksoy et al's performance is as reported in their paper, and their implementation is parallelised C++. Both our computation times (computed in MATLAB) and those of Tan et al. are on a single core. Since each pixel in our approach can be unmixed independently, a CPU or GPU parallelisation could speed up computation time. We downsize all input images before computing the palette extraction step, which takes 3 seconds. In Fig. 13 we analyse the computation time for our full palette extraction and image decomposition method (without the optional filtering step) across a variety of different image sizes. On the right, we analyse the computation time of our method when images with varying palette sizes are chosen. In both diagrams, our method is shown in red, Aksoy et al. [2017] in blue and Tan et al. [2018] in black. Overall, our method is as computationally efficient as Tan et al, and performs significantly better than Aksoy et al.

6.3 Applications

Recolouring and compositing results with our layers can be seen in Fig. 1 and 14 and show that our layers provide more flexibility to the user when editing images. Our layers successfully unmix semi-transparent objects such as fire and smoke, and blurred regions such as aesthetic bokeh. In Fig. 15, we also show some results on green screen keying examples. We found that in some cases, other scene objects can appear in our green layer, not just the background.

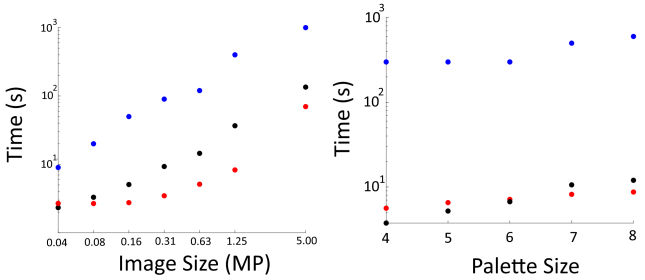
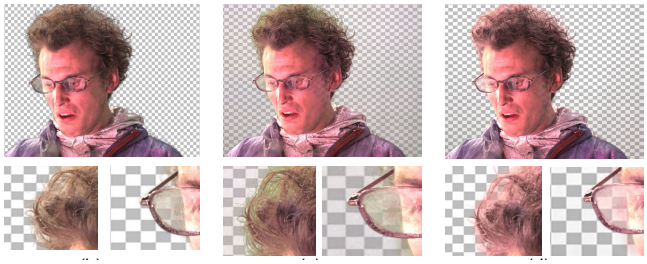


Figure 13: Computation times of our method (red), versus Tan et al. [2018] (black) and Aksoy et al. [2017](blue). Left: Time taken for each method using different input image sizes; Right: Time taken for each method decomposing images with different palette sizes.

When the green layer is removed, unexpected colour changes can then appear in other scene objects (see Fig 15 where green was found to be part of the hair in (b) and (c), which now appears to have a green or pink hue). While [Aksoy et al. 2016] was designed for keying applications and does not suffer from this issue, Tan et al's results show similar colour changes to ours (see Fig 15). In future, we will consider sparsity constraints to overcome this issue, similar to Aksoy et al. [2017].

Figure 14: Input images (top row) with recolouring (columns 1-2) and compositing results (column 3) using our layers.



(a) Aksoy et al. (b) Tan et al. (c) Ours

Figure 15: Keying results of our method, Aksoy et al. [2017] and Tan et al. [2018]. Input image is shown at top of Fig. 16.

7 CONCLUSIONS

We have presented a new geometric image decomposition method which estimates a palette of colours from an image and uses this palette to geometrically unmix the pixels in the image into a number of different layers. We have defined a technique to compute an optimal colour palette for our layer decomposition, which is used to split the RGB space into a number of different regions, with the unmixing technique for a given pixel colour defined by the region in which it lies. We have shown that our alpha values change smoothly across the layers, and in cases where quantisation errors or small discontinuities are present, we have proposed a filtering step for further smoothing. We have shown that our layers can be used for several applications including image recolouring and compositing, and have shown that we are the only layer decomposition method to efficiently create layers that are homogeneous in colour and create zero reconstruction error at a low computational cost. In future work, we would like to investigate how this method could be extended to higher dimensional space such as RGBXY.

ACKNOWLEDGMENTS

This publication emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 15/RP/2776.

REFERENCES

Yağız Aksoy, Tunç Ozan Aydın, Marc Pollefeys, and Aljoša Smolić. 2016. Interactive High-Quality Green-Screen Keying via Color Unmixing. *ACM Trans. Graph.* 35, 5 (2016), 152:1–152:12.

Yağız Aksoy, Tunç Ozan Aydın, Aljoša Smolić, and Marc Pollefeys. 2017. Unmixing-Based Soft Color Segmentation for Image Manipulation. *ACM Trans. Graph.* 36, 2 (2017), 19:1–19:19.

Sean Bell, Kavita Bala, and Noah Snavely. 2014. Intrinsic Images in the Wild. *ACM Trans. Graph.* 33, 4, Article 159 (July 2014), 12 pages.

Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. 2015. Palette-based Photo Recoloring. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34, 4 (July 2015).

Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. 2013. KNN Matting. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 9 (Sept. 2013), 2175–2188.

Yuki Koyama and Masataka Goto. 2018. Decomposing Images into Layers with Advanced Color Blending. *Computer Graphics Forum* 37, 7 (2018), 397–407. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13577

A. Levin, D. Lischinski, and Y. Weiss. 2008. A Closed-Form Solution to Natural Image Matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 2 (Feb 2008), 228–242.

Sharon Lin, Matthew Fisher, Angela Dai, and Pat Hanrahan. 2017. LayerBuilder: Layer Decomposition for Interactive Image and Video Color Editing. arXiv:1701.03754 [cs.GR]

Sharon Lin and Pat Hanrahan. 2013. Modeling How People Extract Color Themes from Images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 3101–3110.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2011. Color Compatibility from Large Datasets. *ACM Trans. Graph.* 30, 4, Article 63 (July 2011), 12 pages.

H. Q. Phan, H. Fu, and A. B. Chan. 2018. Color Orchestra: Ordering Color Palettes for Interpolation and Prediction. *IEEE Transactions on Visualization and Computer Graphics* 24, 6 (June 2018), 1942–1955.

Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 253–259.

M. A. Ruzon and C. Tomasi. 2000. Alpha estimation in natural images. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, Vol. 1. 18–25 vol.1.

L. Shapira, A. Shamir, and D. Cohen-Or. 2009. Image Appearance Exploration by Model-Based Navigation. *Computer Graphics Forum* 28, 2 (2009), 629–638.

Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. 2005. Local color transfer via probabilistic segmentation by expectation-maximization. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1. 747–754.

Jianchao Tan, Jose Echevarria, and Yotam Gingold. 2018. Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. *ACM Transactions on Graphics (TOG)* 37, 6, Article 262 (Dec. 2018), 10 pages.

Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. 2016. Decomposing Images into Layers via RGB-Space Geometry. *ACM Trans. Graph.* 36, 1, Article 61a (Nov. 2016).

Q. Zhang, C. Xiao, H. Sun, and F. Tang. 2017. Palette-Based Image Recoloring Using Color Decomposition Optimization. *IEEE Transactions on Image Processing* 26, 4 (April 2017), 1952–1964.

Figure 16: Palettes and layers generated using the approaches of Aksoy et al. [2017] (left), Tan et al. [2018] (middle) and our approach (right). Input images are shown at the top of the figure (including the input image used to generate results in Fig. 15).