Eurographics Symposium on Rendering 2014
Wojciech Jarosz and Pieter Peers
(Guest Editors)

Volume 33 (2014), Number 4

Vectorising Bitmaps into Semi-Transparent Gradient Layers

C. Richardt [1,2] J. Lopez-Moreno [1,3] A. Bousseau [1] M. Agrawala [4] G. Drettakis [1]

[1] Inria† [2] MPI Informatik [3] URJC, Madrid [4] University of California, Berkeley

(a) Input photograph (b) Multi-layer vector representation (c) Editing result

Figure 1: *Our interactive vectorisation technique lets users vectorise an input bitmap (a) into a stack of opaque and semi-transparent vector layers composed of linear or radial colour gradients (b). Users can manipulate the resulting layers using standard tools to quickly produce new looks (c). We outline semi-transparent layers for visualisation; these edges are not part of our result. We rasterised figures to avoid problems with transparency in some PDF viewers. See supplemental material for vector graphics.*

Abstract

We present an interactive approach for decompositing bitmap drawings and studio photographs into opaque and semi-transparent vector layers. Semi-transparent layers are especially challenging to extract, since they require the inversion of the non-linear compositing equation. We make this problem tractable by exploiting the parametric nature of vector gradients, jointly separating and vectorising semi-transparent regions. Specifically, we constrain the foreground colours to vary according to linear or radial parametric gradients, restricting the number of unknowns and allowing our system to efficiently solve for an editable semi-transparent foreground. We propose a progressive workflow, where the user successively selects a semi-transparent or opaque region in the bitmap, which our algorithm separates as a foreground vector gradient and a background bitmap layer. The user can choose to decompose the background further or vectorise it as an opaque layer. The resulting layered vector representation allows a variety of edits, such as modifying the shape of highlights, adding texture to an object or changing its diffuse colour.

1. Introduction

Vector graphics enjoy great popularity in graphic design for their compactness, editability and scalability. Skilled vector artists commonly blend multiple layers, each composed of simple colour and transparency gradients, to represent the appearance of an object. Each layer typically corresponds to a single aspect of the shading, such as diffuse shading, specular highlights, shadows or Fresnel reflections. The challenge is to fully capture complex shading effects while also maintaining a small number of layers and gradient parameters so that the vector representation remains compact and easy to edit.

Automatic image vectorisation techniques convert bitmaps into a set of editable vector paths and gradients. (We use the term *gradient* to refer to the smooth colour variations encountered in vector graphics; this should not be confused with the gradient of a function, i.e. the vector of its partial derivatives.) We believe that the use of semi-transparent layers has largely been overlooked by existing algorithms which focus on the generation of opaque vector graphics [LL06,SLWS07,LHM09,XLY09,LHFY12]. The resulting vector representation is often very complex and difficult to edit because it combines all aspects of the shading into a single opaque layer.

† Inria authors are at REVES/Inria Sophia-Antipolis Méditerranée.

We present the first interactive *decompositing* tool that can generate a simple, editable set of opaque and semi-transparent vector layers directly from a bitmap. Users progressively select smooth colour regions in the image, which our system separates into a foreground region filled with a vector gradient and a background bitmap layer. Users can repeat the process to further separate the background bitmap into additional gradient layers. The final step is to vectorise the remaining background layer. Figure 1 shows how the resulting vector representation separates the transparent highlights, shadows and liquid for a photograph of a wine glass. Users can edit each layer independently and then re-composite them to quickly generate a variety of looks and appearances.

Semi-transparent layers are especially challenging to extract as they require us to invert the compositing equation [PD84], a non-linear mixture of foreground and background colours. While this problem has been considered in the context of alpha matting [SB96, WC07] and reflection separation [LZW04], existing methods estimate bitmap layers rather than vector gradients and therefore do not provide a small set of parameters suitable for editing the matted layers. Our main technical contribution is a decompositing algorithm that exploits the parametric nature of vector gradients to jointly separate and vectorise semi-transparent layers. In particular, we constrain the foreground colours to vary according to linear or radial gradients, allowing us to solve for a small set of parameters per layer instead of the thousands of unknowns over all pixels in a region. The few degrees of freedom of linear and radial gradients also offer a good trade-off between goodness of fit and editability, while simplifying colour variations in the image to reproduce the clean and sharp look of hand-made vector art. We have integrated our decompositing algorithm in Adobe Photoshop and we export our gradients as Illustrator layers, allowing vector artists to create and edit semi-transparent layers with the tools they are familiar with.

Our approach is limited to foreground layers composed of linear or radial gradients, and therefore cannot capture complex semi-transparent textures or reflections. However, vector images are often not meant to have the complexity of natural images. Instead, vector artists create images that represent objects with stylised lighting. We facilitate the creation of similar vector art from studio photographs, with lighting designed to be simple and effective. We demonstrate that our approach can successfully vectorise a diverse set of studio photographs and drawings into a layered representation (Figures 1 and 6). We also show how the resulting representation supports a variety of editing operations such as modifying the shape of highlights, altering the diffuse colour or texture of a region, or changing the gradient parameters to significantly alter the surface appearance of an object.

2. Related Work

Image vectorisation The goal of vectorisation algorithms is to facilitate or automate the creation of vector graphics from bitmap images. Most methods segment the input image into smooth colour regions that are then represented by vector gradients. Lecot and Lévy [LL06] fit linear and radial gradients to generate vector images in Art Deco style. Gradient meshes represent complex gradients by interpolating colours over the faces of a quad mesh, making them a powerful primitive to capture the smooth colour variations of natural images [SLWS07]. However, this smooth interpolation is less effective in representing sharp colour discontinuities, which require the introduction of holes [LHM09] or closely-spaced mesh lines on either side of image edges [SLWS07]. This limitation motivates Xia et al. [XLY09] and Liao et al. [LHFY12] to represent the image as a piecewise-smooth surface whose control mesh aligns with strong image edges, using a multiresolution representation for the latter. These approaches also share similarities with diffusion curves [OBW*08], which vectorise images by storing colours on each side of strong edges while computing smooth colour variations in-between the edges using a diffusion process. All of these algorithms are designed to vectorise opaque objects into a single layer.

Our approach complements these techniques by decompositing the image into multiple transparent and opaque layers. We represent transparent layers with linear or radial gradients because artists often use such simple gradients to represent reflections and shadows in vector art. We also found that linear and radial gradients are sufficient to represent the highlights, shadows and transparency encountered in stock photographs, which are typically shot using large area light sources. In contrast, opaque layers often contain more complex shading variations, which we express with any suitable representation, such as gradient meshes [SLWS07]. The main advantage of our layered representation is that each layer is simpler than the composed image.

A few vectorisation algorithms consider the creation of layered representations for opaque objects, for example relying on motion detection to separate dynamic foreground objects from a static background when vectorising cartoon animations [ZCZ*09]. Price and Barrett [PB06] decompose objects into several gradient meshes using a hierarchical segmentation algorithm. In contrast, we focus on semi-transparent layers, which require inversion of the compositing equation.

The versatility of vector layers motivates Eisemann and colleagues [EWHS08, EPD09] to convert 3D models into vector images, separating cartoon shading, highlights, shadows and occlusions. Lopez-Moreno et al. [LMPB*13] generate editable shading effects as vector layers by first building normal fields from line drawings. We share a similar goal of generating and editing vector layers, but our input consists of bitmap images. Thus, we must estimate layers by inverting the compositing equation rather than rendering them from a 3D model or a normal field.

Image decomposition The key observation that guides our approach is that complex images often can be explained with just a few simple layers. In fact, researchers have shown

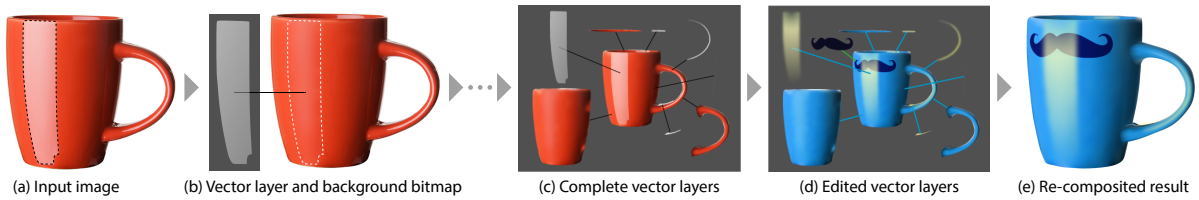


Figure 2: *Interactive workflow. A user selects a region in an image (a), which our algorithm decomposites into a vector foreground layer and a background bitmap (b). This process is repeated and opaque layers are handled with existing tools to create a complete set of vector layers (c). Layers can be edited easily (shown as blue lines in d) and added (green lines), and re-composited to enable powerful editing applications (e).*

that the human visual system infers transparent layers if they result in a simpler explanation of the visual stimuli [Ade93, LZW02]. Vector graphics represent transparency using *alpha compositing* [PD84], which composites the foreground colour $\mathbf{F}$ *over* the background $\mathbf{B}$ using the foreground's opacity channel α:

$$\mathbf{I} = \mathbf{F} \circ \mathbf{B} = \alpha \cdot \mathbf{F} + (1-\alpha) \cdot \mathbf{B}. \tag{1}$$

Matting algorithms aim to invert this equation in order to recover the RGB colours $\mathbf{F}$ and $\mathbf{B}$ and the scalar α at each pixel of an input image $\mathbf{I}$. Existing algorithms solve this problem using assumptions on the colour distributions of the foreground and background layers [SB96, CCSS01, WC07, LLW08]. While we adopt a similar strategy, we adapt it to our context by constraining smooth regions to be filled with linear or radial gradients, allowing us to jointly separate and vectorise the foreground layer into commonly used vector gradients.

Reflection-separation methods handle semi-transparent layers by estimating colour channels pre-multiplied by the alpha channel, reducing the image formation model to a sum of two layers $\mathbf{I} = \mathbf{F} + \mathbf{B}$ [LZW04, LB14]. While this approach linearises the problem and facilitates its resolution, it prevents the subsequent independent manipulation of colour and transparency as is common in vector graphics applications. Yeung et al. [YTBK11] describe a user-assisted method to extract semi-transparent objects from photographs and re-composite them. They introduce a three-layer bitmap representation to capture bright highlights, smooth colour transmission and refractive deformations. While we do not support refractive deformations, we unify highlights and colour transmission by expressing both effects as parametric gradients.

3. Overview

A layered image consists of multiple layers $\mathbf{L}_i$ that are composited from back to front onto the bottom-most layer $\mathbf{L}_1$. Thus, each pixel $\mathbf{x}$ of an image $\mathbf{I}_n$ composed of n layers is formed recursively as

$$\mathbf{I}_n(\mathbf{x}) = \begin{cases} \mathbf{L}_n(\mathbf{x}) \circ \mathbf{I}_{n-1}(\mathbf{x}) & n > 1 \\ \mathbf{L}_1(\mathbf{x}) & n = 1. \end{cases} \tag{2}$$

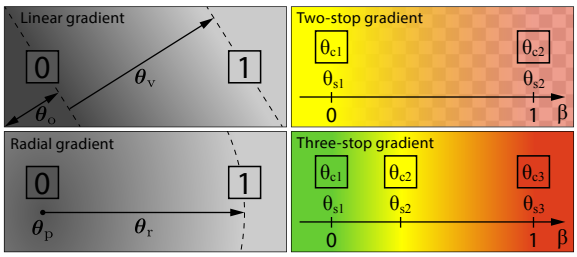


Figure 3: *We define parametric gradients by composing a gradient function g (left) with a colour function c (right), that encodes colour and transparency as an (r,g,b,α) tuple.*

To decomposite an image into its constituent layers, we must invert this equation. However, the inverse problem is underdetermined as n layers result in n unknown RGB colours and $n-1$ unknown alphas to explain one RGB observation – per pixel. We adopt a greedy approach, removing one layer at a time in their reverse compositing order, from front to back. That is, at each step, we separate the image $\mathbf{I}_k$ into a foreground vector layer $\mathbf{L}_k$ with its alpha channel and the remaining background bitmap $\mathbf{I}_{k-1}$, which we further decompose in subsequent steps.

We additionally constrain each semi-transparent foreground layer by modelling it as a parametric gradient function $f = c \circ g = c(g(\mathbf{x},\theta),\theta)$, composed of a gradient function $g(\mathbf{x},\theta)$ that maps a pixel coordinate $\mathbf{x} \in \mathbb{N}^2$ to a parametric value $\beta \in \mathbb{R}$, and a colour function $c(\beta,\theta)$ that maps this parametric value to an (r,g,b,α) colour, all based on a set of parameters θ. As shown in Figure 3, these parameters encode the colours of a gradient as well as their transition, through the colour and gradient functions, respectively. The gradient function g defines linear gradients by a direction vector θ_v and signed offset distance θ_o from the origin, and radial gradients are defined by centre θ_p and radius θ_r:

$$g_{\text{linear}}(\mathbf{x},\theta) = \frac{\mathbf{x} \cdot \theta_v}{\|\theta_v\|^2} + \theta_o \text{ and} \tag{3}$$

$$g_{\text{radial}}(\mathbf{x},\theta) = \frac{\|\mathbf{x} - \theta_p\|}{\theta_r}. \tag{4}$$

The colour function c maps the parametric gradient to colours using piecewise-linear interpolation between *gradient stops* with colours θ_{ci} $(i = 1,\ldots,k)$ at offsets $\theta_{si} \in [0,1]$, such that

$\theta_{s1} := 0$, $\theta_{sk} := 1$. We express gradients with 2 and 3 stops as

$$c_2(\beta,\theta) = \mathrm{mix}(\theta_{c1},\theta_{c2},\beta) \text{ and} \tag{5}$$

$$c_3(\beta,\theta) = \begin{cases} \mathrm{mix}\left(\theta_{c1},\theta_{c2},\frac{\beta}{\theta_{s2}}\right) & \beta \le \theta_{s2} \\ \mathrm{mix}\left(\theta_{c2},\theta_{c3},\frac{\beta-\theta_{s2}}{1-\theta_{s2}}\right) & \beta > \theta_{s2} \end{cases}, \tag{6}$$

with $\mathrm{mix}(\mathbf{a},\mathbf{b},t) = (1-t)\cdot\mathbf{a} + t\cdot\mathbf{b}$ for linear interpolation. The colour functions can easily be extended to more stops.

We represent opaque layers with arbitrary colour fills, which include constant colour, linear and radial gradients of multiple colours, and gradient meshes.

Our interactive method proceeds as follows. The user first selects a smooth image region for decompositing using Adobe Photoshop's selection tools. The user additionally indicates if the region is opaque or semi-transparent, if it is a linear or radial gradient, and how many stops it has (we have presets for two and three). The user can optionally indicate that a region is a highlight or a shadow, which further constrains the decomposition. Our algorithm then separates the selected region from its background and vectorises it. It first converts the boundary to a Bézier spline using the open-source vectorisation tool Potrace [Sel03]. It then separates the region into a foreground gradient layer, by fitting a gradient to the colours inside the region (Section 4.1), and a background bitmap layer (Section 4.2). The user can choose to further decomposite the background bitmap into additional gradient layers by repeating this procedure, or vectorise the background as a one or more opaque layers.

4. Decompositing

Extracting a semi-transparent gradient layer requires us to estimate the parameters θ of the foreground gradient f as well as the bitmap background $\mathbf{B}$ that together best explain the input image. We cast this estimation as a least-squares minimisation over the selected region R:

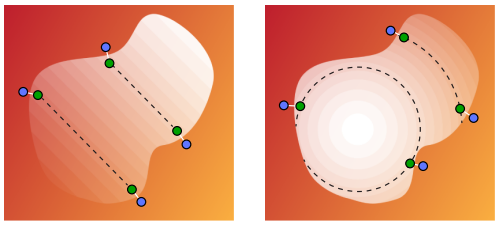
$$\arg\min_{\theta,\mathbf{B}} \sum_{\mathbf{x}\in R} \left(\mathbf{I}(\mathbf{x}) - f(\mathbf{x},\theta)\circ\mathbf{B}(\mathbf{x})\right)^2, \tag{7}$$

where the compositing implicitly involves the foreground alpha:

$$f(\mathbf{x},\theta)\circ\mathbf{B}(\mathbf{x}) = f_\alpha(\mathbf{x},\theta)\cdot f_{\mathrm{rgb}}(\mathbf{x},\theta) + (1-f_\alpha(\mathbf{x},\theta))\cdot\mathbf{B}(\mathbf{x}), \tag{8}$$

and subscripts select (r,g,b,α) channels: $f_\alpha(\mathbf{x},\theta)$ is the alpha component of the foreground gradient, while $f_{\mathrm{rgb}}(\mathbf{x},\theta)$ is its RGB colour vector $\left(f_r(\mathbf{x},\theta), f_g(\mathbf{x},\theta), f_b(\mathbf{x},\theta)\right)$.

Finding the foreground gradient parameters θ and background image $\mathbf{B}$ that minimise Equation 7 requires solving an under-determined system. Each input pixel in $\mathbf{I}$ provides one colour and we must recover one colour for each background pixel as well as the set of foreground gradient parameters for the region. Our algorithm tackles this problem in two steps, by first estimating the foreground gradient (Section 4.1) and then the background bitmap (Section 4.2).

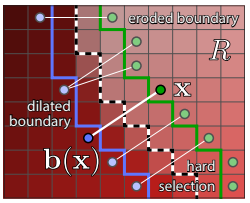


(a) Linear gradient (b) Radial gradient

Figure 4: *We can separate the foreground 2-stop gradient if at least two iso-contours (dashed lines) are observed over two different background colours. This condition extends to each linear piece of multi-stop gradients. We assume the background is continuous across the region boundary, so blue locations approximate the background at the green locations.*

4.1. Solving for a semi-transparent foreground

Equation 7 corresponds to the matting problem, which Smith and Blinn [SB96] proved to have a unique solution in the context of bitmap images if each pixel of the foreground is observed over at least two different known background colours. In our context, the foreground pixel values are the same along each iso-contour of the gradient, which implies that we can solve for the matting problem along an iso-contour if we observe at least two of its pixels over different background colours. Furthermore, two different iso-contours are sufficient to constrain the location and colours of two-stop gradients (Figure 4). Our problem is hence well-posed if we observe at least two different iso-contours, each with at least two pixels on different backgrounds, for each linear segment of a multi-stop gradient.



While in theory the background colour is unknown over the entire region, we make the additional assumption that it is continuous across the boundary of the region, which allows us to approximate the background colour of each pixel $\mathbf{x}$ along the boundary with the image colour at the closest location $\mathbf{b}(\mathbf{x})$ outside the region boundary (see inset). In practice, we erode and dilate the hard selection by a few pixels to account for anti-aliasing or blurry edges in the image, sampling $\mathbf{x}$ along the boundary ∂R of the eroded region (shown in green in the inset) and $\mathbf{b}(\mathbf{x})$ outside the dilated boundary (shown in blue). We then solve for the optimal parameters Θ of f solely based on the boundary pixels as

$$\Theta = \arg\min_{\theta} \sum_{\mathbf{x}\in\partial R} \left(\mathbf{I}(\mathbf{x}) - f(\mathbf{x},\theta)\circ\mathbf{I}(\mathbf{b}(\mathbf{x}))\right)^2. \tag{9}$$

We minimise Equation 9 using a standard non-linear least-squares solver. We found that the IPOPT solver [WB06], wrapped by the CoMISo package [BZK12] with automatically differentiated Jacobians and Hessians, achieves good convergence. We constrain variables expressing colours, opacities and stop offsets (i.e. θ_{ci} and θ_{si}) to the unit range to produce valid vector gradients.

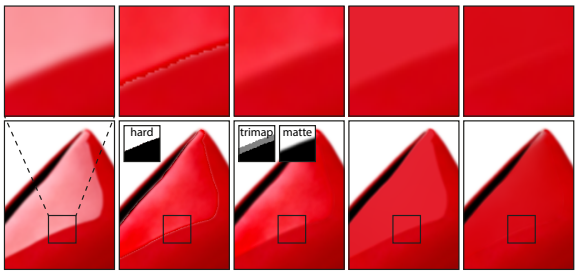


Figure 5: *Background estimation using Equation 10 for image (a): a hard region boundary causes halo artefacts (b) that are removed with a matted boundary (c). A smoothness term suppresses noise (d) and combined with Poisson blending produces the final result (e). To clearly see the differences, please view this figure on a monitor.*

When the background is uniformly coloured, we cannot obtain two observations of the foreground on different background colours and therefore cannot uniquely solve Equation 9. In this case, we offer users two additional constraints. Users may choose to fix the foreground to have a known constant colour so that only the alpha channel is unknown. We provide this constraint to users in the form of two presets: a white colour for highlights and a black colour for shadows. Users can alternatively choose to favour a semi-transparent solution, which we express as an additional weak regularisation term $\varepsilon|f_\alpha(\mathbf{x},\theta)-0.5|^2$ in Equation 9.

Limitation Our assumption of a continuous background across boundaries may not hold when the background contains strong edges that are coincident with the region boundary. We later account for incorrect background pixel estimates in a subsequent step (Section 4.3).

4.2. Recovering the background

Given the foreground gradient $f(\mathbf{x},\Theta)$, we estimate the background bitmap $\mathbf{B}$ from the image $\mathbf{I}$ by inverting the composition equation (1) at each pixel of the selected region as the background estimate

$$\widetilde{\mathbf{B}}(\mathbf{x}) = \frac{\mathbf{I}(\mathbf{x}) - f_\alpha(\mathbf{x},\Theta)\cdot f_{\mathrm{rgb}}(\mathbf{x},\Theta)}{1-f_\alpha(\mathbf{x},\Theta)+\varepsilon}, \tag{10}$$

with $\varepsilon = 10^{-20}$ for robustness to opaque pixels.

However, the resulting background layer can be polluted with three types of artefacts (Figure 5). First, while the user specifies a hard selection boundary, the region often has a blurry boundary in the original bitmap, which produces halo artefacts after foreground removal (Figure 5b). We address this problem by computing the alpha matte $M(\mathbf{x})$ of the soft boundary in the image, and then replacing $f_\alpha(\mathbf{x},\Theta)$ with $M(\mathbf{x})f_\alpha(\mathbf{x},\Theta)$ in Equation 10. We use the Laplacian matting algorithm to compute the matte [LLW08], dilating and eroding the user-specified region boundary to generate a suitable trimap (see Figure 5c).

The two other artefacts we observe are noise from the input bitmap, which is amplified by the inversion of the compositing equation, and a low-frequency residual due to smooth variations in the foreground that are not modelled by a linear or radial gradient, producing a colour mismatch on the border of the region. We treat both artefacts using smoothing (Figure 5d) and Poisson blending [PGB03] (Figure 5e), solving for the background that minimises

$$\begin{aligned}\arg\min_{\mathbf{B}}\ & \lambda_c\cdot\textstyle\sum_{\mathbf{x}\in\partial R}\big(\mathbf{B}(\mathbf{x})-\mathbf{I}(\mathbf{b}(\mathbf{x}))\big)^2 \\ &+\lambda_p\cdot\textstyle\sum_{\mathbf{x}\in R}\big(1-f_\alpha(\mathbf{x},\Theta)\big)\cdot\Big(\nabla\mathbf{B}(\mathbf{x})-\nabla\widetilde{\mathbf{B}}(\mathbf{x})\Big)^2 \\ &+\lambda_v\cdot\textstyle\sum_{\mathbf{x}\in R}\|\nabla\mathbf{B}(\mathbf{x})\|^2,\end{aligned} \tag{11}$$

where the first term favours continuous colours across the region boundary, the second term preserves variations of the initial estimate $\widetilde{\mathbf{B}}$ within the region, and the third term penalises variation overall. We additionally weight the second term by $(1-f_\alpha)$ because the division by the same factor in Equation 10 amplifies noise in the image, such as sensor noise or quantisation, particularly for more opaque foregrounds. The optimisation can be solved using a standard linear least-squares solver, such as the backslash operator in MATLAB. We use weights $\lambda_c=10$, $\lambda_p=1$ and $\lambda_v=0.1$ for all results.

We also experimented with total-variation (TV) denoising $|\nabla\mathbf{B}|$ that better preserves sharp edges but is non-linear [ROF92]. We use $\|\nabla\mathbf{B}\|^2$ by default and allow users to apply total variation if they are not satisfied by the diffused result. We solve the optimisation with total-variation denoising using iteratively reweighed least-squares [RW09].

4.3. Iterative gradient refitting

Image pixels outside a region are not always similar to the background colour behind the region, for example if the region boundary aligns with an edge in the background. These incorrect background samples are outliers that bias the estimation process and therefore need to be excluded. We iteratively exclude these outliers and refit the gradient until convergence. Specifically, we compare the sampled image colour $\mathbf{I}(\mathbf{b}(\mathbf{x}))$ to the initial background estimate obtained from Equation 10 and remove any samples that differ significantly. The complete inlier criterion is

$$\big(1-f_\alpha(\mathbf{x},\Theta)\big)\cdot\Big\|\mathbf{I}(\mathbf{b}(\mathbf{x}))-\widetilde{\mathbf{B}}(\mathbf{x})\Big\| < \tau, \tag{12}$$

where we weight the Euclidean colour difference with a $(1-f_\alpha)$ term as the background estimate is less accurate for more opaque pixels. We use ten iterations and a default value of $\tau=0.2$. Users can alternatively refine this outlier rejection by interactively erasing samples along boundaries that should not contribute to the estimation. We also provide controls for users to directly constrain the background to a specific constant colour where necessary.

Figure 6: *Decompositing and editing results. For each result, we show the input photo or drawing on the left, and on the right an exploded view of all layers as well as editing results for some of the images. We added outlines to semi-transparent layers for visualisation; these lines are not part of our results. We use blue lines to show edited layers, green lines for added layers, and the re-composited result in the centre. Most gradients have two stops, except for the camera lens where three-stops gradients capture saturated and coloured highlights.*

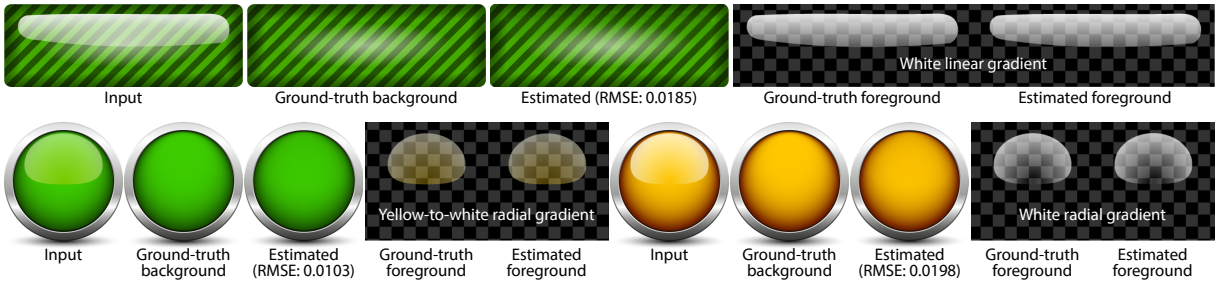


Figure 7: *Ground-truth comparison on vector art: our decompositing results are close to the ground-truth decompositions as indicated by the low root-mean-squared errors (RMSE) of the background bitmaps. The vector gradients are correctly estimated despite the textured background and the subtle colour gradients in the two radial gradient examples.*

4.4. Separating opaque layers

Opaque layers can be captured by any existing vectorisation algorithm. In our implementation, we provide users with the ability to vectorise an opaque region as either a linear or radial gradient, or a gradient mesh [SLWS07]. Users can also use the image tracing tools in Illustrator or Inkscape for this task. As opaque regions fully obscure the background, they leave a hole that we fill with a colour diffusion [OBW*08].

4.5. Implementation

We use Adobe's publicly available ExtendScript Toolkit to communicate with Photoshop. Using a script, we export the current image and selection (as a binary mask), and pass them to our program that implements the decompositing steps. The resulting foreground vector and background bitmap layers are loaded back into Photoshop, and control is returned to the user. We use Scalable Vector Graphics (SVG) as an interchange format for all vector layers, which we finally load into Adobe Illustrator to create the final vectorised result.

5. Results

Figures 1, 2, 6 and 7 show layers extracted from input bitmaps with our decompositing algorithm. All results have been produced with 2-stop gradients, except the camera lens and traffic cone that contain multiple 3-stop gradients. Most layers are linear gradients; some highlights in the glass (Figure 1) and the shadow of the vacuum cleaner (Figure 6) are radial gradients. While shadows and highlights are mainly black and white, the liquid in the glass and reflections in the lens, among others, are colour gradients. Our layers capture effects like Fresnel reflections on the side of the wine glass, but we do not recover and store the relationship between the colours of the reflection and the background. However, users can manually adjust the colours of gradient stops in the reflection layers to match the appearance of a new background.

Decompositing a single semi-transparent layer takes a few seconds to a minute, depending on the size of region and background estimation approach, which makes our interactive approach practical. The examples we show were created from scratch in a few minutes to an hour, depending on the number of layers and complexity of regions and images. Typical times for each region are: <10 s for selection in Photoshop, <1 s outline vectorisation using Potrace, $1-10$ s foreground fitting (Equation 9), $1-5$ s Laplacian matting and $5-30$ s background estimation (Equation 11). The current bottleneck is the background estimation (Poisson and denoising) which could be accelerated with a multi-grid GPU solver [MP08].

Our current implementation vectorises each region independently. As a result, boundaries of regions do not always match up perfectly. We manually adjusted the boundaries between regions of the shoe, and the shadow and body of the vacuum cleaner in Figure 6. Automatic snapping could automate this task. To decomposite a hand-drawn bitmap, we first applied morphological closing to remove the contour lines (vacuum cleaner and purifier in Figure 6). The corresponding layers are provided in the supplemental material.

Evaluation For ground-truth evaluation, we process a bitmap generated from vector art. We show two examples in Figure 7: a linear gradient on a textured button and two radial highlight on a round button. The results of decompositing are very close to the original layers used to create the illustration.

Limitations To make the decompositing problem tractable, our approach makes some assumptions (Section 4.1). While these assumptions are often satisfied for two-stop linear gradients, failure cases are more frequent for radial and multi-stop gradients as more iso-contours are susceptible to be missed. If the assumptions are violated, our approach may produce undesirable decompositions that nonetheless are often still valid, as shown in Figures 8 and 9. Adding additional constraints, such as a white gradient colour or medium opacity, often resolves existing ambiguities.

Vector editing applications The layered vector art created using our approach can be edited with conventional vector graphics tools such as Illustrator or Inkscape. In Figure 6, we change the shape of the large highlight on the left of the bottle, and the texture of the shoe. We change the colour of the liquid layer in Figure 1 and introduce a layer for the straw.

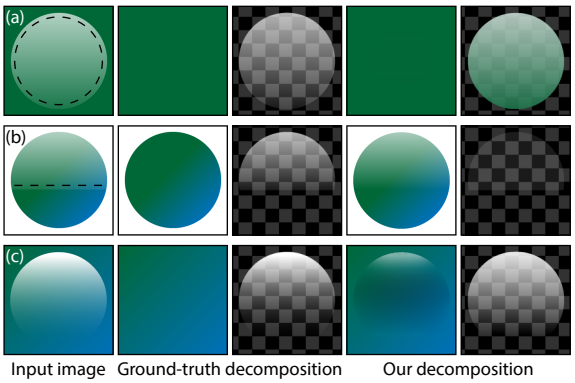


Figure 8: *Examples of failure cases, for which our approach produces valid but undesirable decompositions. If gradient iso-contours (dashed lines) are not observed over two distinct backgrounds (a), the gradient colours and opacities are underconstrained. If only one iso-contour is observed (b), we obtain a constant-opacity gradient. Quadratic gradients (c) can be approximated as piecewise linear, but leave a residual in the background.*

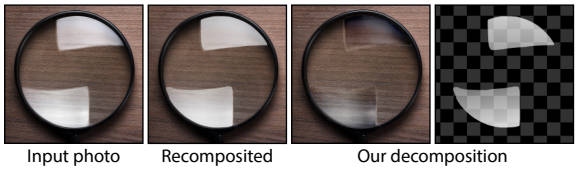


Figure 9: *Example of a real-world failure case. If an image layer cannot be represented well by a colour gradient, our method tends to leave residual foreground colours in the background layer. However, we observed that these residuals are often smoothed away by subsequent vectorisation steps.*

Similarly, a layer is added in Figure 2 for the moustache. All of these edits are difficult or impossible to achieve starting from just a bitmap, but using our layered image vectorisation approach, they become much easier.

Potential automation Users of our tool currently need to manually select the gradient type (linear or radial), number of stops (two or three) and any other constraints (black, white or constant colour) of each region. Our framework can in principle allow for an automatic selection of these parameters, for example by selecting the gradient that minimises Equation 9. However, we found the choice of gradient primitive fairly easy; it also leaves artistic freedom to the user, for instance to constrain a highlight to be white even if it is slightly coloured in the input.

6. Conclusion

Vector artists create complex artworks by stacking simple layers. We have demonstrated the benefit of this strategy for image vectorisation, both to capture and edit transparency effects in drawings and photographs. Our approach facilitates the creation of such layered vector graphics from bitmaps, and we thus see our method as a valuable tool for professional artists and novice users alike.

While we focused on alpha compositing, our framework should easily extend to other popular blending modes, such as *screen* and *overlay*. Another extension of our work would consist in casting the decompositing problem as a global optimisation across multiple layers to better capture residual colour variations, potentially using our greedy solution as an initial solution. Finally, our layered representation could facilitate the creation of *vector shade trees* [LMPB*13] from existing drawings and photographs, which would represent a first step towards the capture and transfer of material appearance in illustrations.

Acknowledgements We thank Julia Chatain for her work on an early version of this project, Robert Carroll and David Bommes for help with optimisation, and Pascal Barla and Frédo Durand for feedback on an early draft. This work was supported by the Inria CRISP associate team, ANR-12-JS02-003-01 DRAO, and a research donation from Adobe.

Image credits Images from Shutterstock: the wine glass (Givaga) and bottle (Dmitri Gristsenko), the red cup (George Dolgikh) and shoe (Picsfive), the magnifying glass (Ruslan Grumble) and the glossy round button (Roman Sotola). Lens image by Flickr user squinza (CC BY-SA 2.0). Vacuum cleaner and purifier drawn by Spencer Nugent on sketch-a-day.com. Striped green button by openclipart.org user inky2010.

References

[Ade93] ADELSON E. H.: Perceptual organization and the judgment of brightness. *Science 262*, 5142 (1993), 2042–2044. 3

[BZK12] BOMMES D., ZIMMER H., KOBBELT L.: Practical mixed-integer optimization for geometry processing. In *Curves and Surfaces* (2012), pp. 193–206. 4

[CCSS01] CHUANG Y.-Y., CURLESS B., SALESIN D. H., SZELISKI R.: A Bayesian approach to digital matting. In *CVPR* (2001), p. 264. 3

[EPD09] EISEMANN E., PARIS S., DURAND F.: A visibility algorithm for converting 3D meshes into editable 2D vector graphics. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009), 83:1–8. 2

[EWHS08] EISEMANN E., WINNEMÖLLER H., HART J. C., SALESIN D.: Stylized vector art from 3D models with region support. *Computer Graphics Forum (Proc. EGSR) 27*, 4 (2008), 1199–1207. 2

[LB14] LI Y., BROWN M. S.: Single image layer separation using relative smoothness. In *CVPR* (2014). 3

[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. 1, 2

[LHM09] LAI Y.-K., HU S.-M., MARTIN R. R.: Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009), 85:1–8. 1, 2

[LL06] LECOT G., LÉVY B.: Ardeco: automatic region detection and conversion. In *EGSR* (2006), pp. 349–360. 1, 2

[LLW08] LEVIN A., LISCHINSKI D., WEISS Y.: A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 2 (2008), 228–242. 3, 5

[LMPB*13] LOPEZ-MORENO J., POPOV S., BOUSSEAU A., AGRAWALA M., DRETTAKIS G.: Depicting stylized materials with vector shade trees. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 118:1–10. 2, 8

[LZW02] LEVIN A., ZOMET A., WEISS Y.: Learning to perceive transparency from the statistics of natural scenes. In *NIPS* (2002). 3

[LZW04] LEVIN A., ZOMET A., WEISS Y.: Separating reflections from a single image using local features. In *CVPR* (2004), vol. 1, pp. 306–313. 2, 3

[MP08] MCCANN J., POLLARD N. S.: Real-time gradient-domain painting. *ACM Transactions on Graphics (Proc. SIGGRAPH) 27*, 3 (2008), 93:1–7. 7

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (Proc. SIGGRAPH) 27*, 3 (2008), 92:1–8. 2, 7

[PB06] PRICE B., BARRETT W.: Object-based vectorization for interactive image editing. *The Visual Computer 22*, 9 (2006), 661–670. 2

[PD84] PORTER T., DUFF T.: Compositing digital images. *Computer Graphics (Proc. SIGGRAPH) 18*, 3 (1984), 253–259. 2, 3

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (2003), 313–318. 5

[ROF92] RUDIN L. I., OSHER S., FATEMI E.: Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena 60*, 1–4 (1992), 259–268. 5

[RW09] RODRÍGUEZ P., WOHLBERG B.: Efficient minimization method for a generalized total variation functional. *IEEE Transactions on Image Processing 18*, 2 (2009), 322–332. 5

[SB96] SMITH A. R., BLINN J. F.: Blue screen matting. In *SIGGRAPH* (1996), pp. 259–268. 2, 3, 4

[Sel03] SELINGER P.: Potrace: a polygon-based tracing algorithm, 2003. 4

[SLWS07] SUN J., LIANG L., WEN F., SHUM H.-Y.: Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (2007), 11. 1, 2, 7

[WB06] WÄCHTER A., BIEGLER L. T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*, 1 (2006), 25–57. 4

[WC07] WANG J., COHEN M.: Image and video matting: A survey. *Foundations and Trends in Computer Graphics and Vision 3*, 2 (2007), 97–175. 2, 3

[XLY09] XIA T., LIAO B., YU Y.: Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009), 115:1–10. 1, 2

[YTBK11] YEUNG S.-K., TANG C.-K., BROWN M. S., KANG S. B.: Matting and compositing of transparent and refractive objects. *ACM Transactions on Graphics 30*, 1 (2011), 2:1–13. 3

[ZCZ*09] ZHANG S.-H., CHEN T., ZHANG Y.-F., HU S.-M., MARTIN R. R.: Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (2009), 618–629. 2