Generating Digital Painting Lighting Effects via RGB-space Geometry

LVMIN ZHANG, Soochow University / Style2Paints, China
EDGAR SIMO-SERRA, Waseda University / JST PRESTO, Japan
YI JI, Soochow University, China
CHUNPING LIU, Soochow University, China

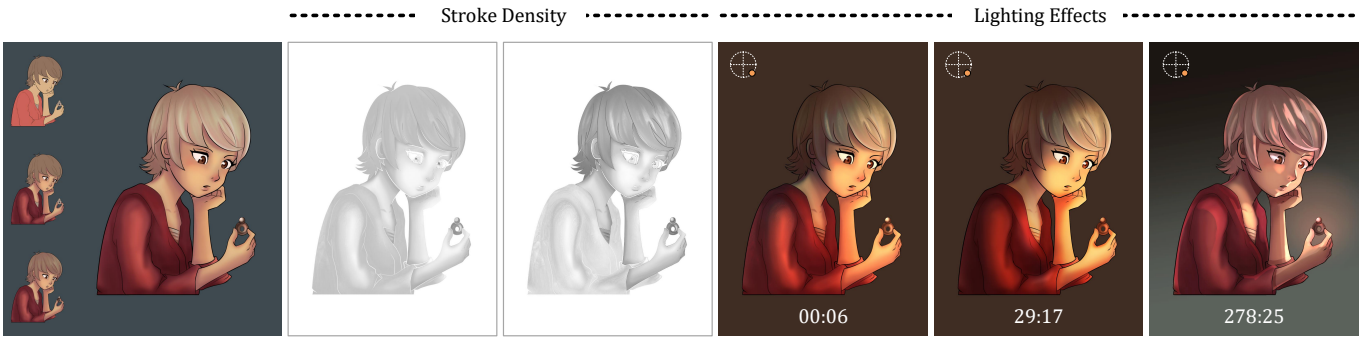


Fig. 1. **Generating digital painting lighting effects.** An artist is invited to draw a digital painting on the left. We then estimate a stroke density map using the canvas image, which is compared to the measured stroke density computed with the artist's real stroke history. The estimated stroke density is used to generate our lighting effects in real-time. We also invite artists to draw lighting effects for comparisons. One artist, namely conditioned artist, is invited to paint effects with similar style to ours, and another artist, namely unconditioned artist, is asked to try best to relight the original image according to artistic vision. The time taken for each approach is shown at the bottom of each image (in minutes), and the same with other figures in this paper. *© Artwork traced by the team of style2paints from an artwork by David Revoy under CC-BY license, www.peppercarrot.com, and the licenses in the Fig. 2-10 are the same.*

We present an algorithm to generate digital painting lighting effects from a single image. Our algorithm is based on a key observation: artists use many overlapping strokes to paint lighting effects, *i.e.*, pixels with dense stroke history tend to gather more illumination strokes. Based on this observation, we design an algorithm to both estimate the density of strokes in a digital painting using color geometry, and then generate novel lighting effects by mimicking artists' coarse-to-fine workflow. Coarse lighting effects are first generated using a wave transform, and then retouched according to the stroke density of the original illustrations into usable lighting effects.

Our algorithm is content-aware, with generated lighting effects naturally adapting to image structures, and can be used as an interactive tool to simplify current labor-intensive workflows for generating lighting effects for digital and matte paintings. In addition, our algorithm can also produce usable lighting effects for photographs or 3D rendered images. We evaluate our approach with both an in-depth qualitative and a quantitative analysis which includes a perceptual user study. Results show that our proposed approach is not only able to produce favorable lighting effects with respect to existing approaches, but also that it is able to significantly reduce the needed interaction time.

1 INTRODUCTION

Lighting plays an important role in digital and matte painting. Unlike physical illumination in real world or rendered scenes, the *painted* lighting effects in digital paintings are created by artists using heterogeneous strokes. The objective is artistic expression, and thus, more often than not, the illumination is not subject to any physical constraints. In current artistic workflows, artists manually paint these lighting effects and tediously modify them to find the best composition. In order to create usable lighting effect products, artists usually first draw some global or coarse illumination layers, and then retouch the details of these layers to naturally fit the original image content in a very time-consuming process. Some examples of lighting effects are shown in Fig. 1 and 2.

The problem of creating or manipulating digital painting lighting effects in illustrations or matte paintings has important differences with 3D relighting like [Debevec et al. 2000], which usually has emphasis on exploiting the 3D object structure or physical light geometry, whereas in this work we focus on the artistic stroke structure in digital paintings. We aim at simplifying the labor-intensive lighting effect painting workflow, where many objects are not even painted to have 3D appearance, making it hardly possible to solve the accurate physical geometry of these objects. Moreover, in real-life lighting effect creation, it is very important to ensure that details of artists' carefully painted illustration are not significantly modified when applying effects. Even small distortions can severely limit the applicability to real-world professional digital painting use cases. This makes it difficult for normal or proxy construction algorithms [Hudon et al. 2018; Xu et al. 2015] to meet artists' needs.

This paper presents an algorithm to synthesize digital painting lighting effects based on a key assumption: artists' newly painted strokes are related to their previous stroke history. In particular, when artists are asked to paint a novel lighting effect layer, their strokes share similarity to their previously painted strokes. To be specific, pixels with dense corresponding stroke history tend to capture more diversified lighting or shadow effects, whereas pixels

Fig. 2. **Examples of manually painted lighting effects.** We show several examples of lighting effects created in different styles by multiple artists. The reference lighting direction is shown in the top-left while the recorded painting time is shown in the top-right (in minutes).

with relatively sparse stroke history are likely to correspond to ambient or smooth color. Therefore, the density of stroke history is an important clue when generating digital painting lighting effects.

For most existing digital paintings, the stroke history is not available and thus the stroke density cannot be directly measured. Furthermore, recording and storing all strokes in a non-destructive manner is impractical due to the high resolution and sheer number of strokes used in most digital paintings. In our approach, we bypass this issue by designing an algorithm to directly estimate the density of stroke history from a single finished illustration. Our algorithm first extracts a virtual palette from the input image, and then makes use of the relationship between pixel colors and palette colors to estimate the stroke density. This estimation does not actually decompose images or extract strokes, whereas it is particularly effective to guide the synthesis of novel lighting effects.

We then generate the lighting effects by mimicking artists' workflow in a coarse-to-fine approach. Firstly, we generate coarse effect maps as an initial composition, and then proceed to refine the map according to the pixel-wise stroke density. The coarse effect map provides a rough and low-frequency effect of the highlights, shadows, and the color variations caused by the influence of nearby objects. This coarse effect map is then refined to fit the original image structure, allowing the generation of aesthetically pleasing lighting results. Our algorithm supports multiple types of light sources and can directly serve as a plug-in for professional digital painting tools.

We perform both an in-depth quantitative and qualitative evaluation of our approach, and validate our algorithm in a perceptual user study and compare with manually painted lighting effects. We find that the lighting effects generated by our approach are significantly better than existing approaches. Furthermore, our approach is found to be usable by artists and can be integrated in the matte painting workflow, which we find can speed up lighting effect composition by roughly 50%. Finally, results show our approach also has applicability to non-painted images such as photographs and 3D rendered ones.

In summary, our contributions are (a) an algorithm to generate and manipulate digital painting lighting effects, (b) an algorithm to estimate the density of artists' stroke history for digital paintings, (c) experiments on the connection between colors, strokes, and painted lighting effects based on real data collected from professional artists, and (d) an in-depth evaluation including a perceptual user study.

2 RELATED WORK

2.1 Sample-based Relighting

In order to manipulate the lighting condition in images, one classic approach is to use multiple samples with different lighting directions on a same scene, and then formulate the relighting objective as a light transport or interpolation problem. Some typical models are reflectance field [Debevec et al. 2000], progressive optimization [Matusik et al. 2004], wavelet [Peers and Dutre 2005], compressive transport [Peers et al. 2009], and deep neural networks [Chen et al. 2010; Peers et al. 2007]. Nevertheless, in digital painting use cases, samples with different lighting directions are labor-intensive to obtain, making it a chicken-and-egg problem to generate effects using manually created ones. This limits the applicability of these relighting approaches. Therefore, instead of requiring samples with multiple lighting directions, our approach generates digital painting lighting effects directly using the density of the artists' stroke history, which makes our tool well-suited for artistic creation scenarios.

2.2 Intrinsic Images

Many image relighting algorithms rely on decomposing images into reflectance (albedo) maps and illumination (irradiance) maps, known as intrinsic image decomposition [Barrow and Tenenbaum 1978]. Related techniques are usually based on optimization [Bell et al. 2014, 2013, 2015; Shen et al. 2011] or learning [Aksoy et al. 2018; Barron and Malik 2012; Gehler et al. 2011; Serra et al. 2012]. Although intrinsic decomposition is beneficial and effective to diversify relighting applications, in the case of digital paintings there are even more challenges. Digital illustrations are carefully painted by artists, and thus it is important to preserve as much of the original details as possible. Nevertheless, the perfect image reconstruction after layer editing is still a non-trivial open problem to intrinsic methods. For these reasons, instead of decomposing the illumination beyond input images, we base our pipeline on the structure of artists' painting strokes and try to conserve as much of the original details as possible.

2.3 Normal Estimation and Proxy Construction

Another common relighting approach is to estimate the albedo and surface normals, or proxy surfaces. This allows to efficiently compute the illumination for different directions [Kender and Smith 1984; Wu et al. 2008]. Fitting surfaces to artistic creations has also been extensively studied [Sykora et al. 2014; Wu et al. 2007; Xu et al. 2015]. These approaches typically depend on the consistency of the image albedo, as well as use strong assumptions such as Lambertian scenes. Recently, deep learning methods [Hudon et al. 2018; Su et al. 2018] have been also used to estimate surface normals. Sengupta *et al.* [2018] propose to estimate the normal map of faces. Kanamori and Endo [2018] extend this technique to full-body shots of individuals. Yu and Smith [2019] design an unsupervised training strategy to estimate normal maps for buildings and sculptures. Philip *et al.* [2019] build up a geometry relighting proxy to help neural networks, and Sun *et al.* [2019] propose to relight facial images using neural networks from a small but high-quality omnidirectional dataset. However, in general, learning based approaches may suffer from a lack of generalization caused by dependency on training data, which contains many implicit and explicit biases. Instead of relying on the estimation of surface normal and albedo, we make use of artists' stroke density to generate effects that are visually similar to those created manually by artists. Our approach exhibits stable generalization to a wide variety of images and can be directly used in digital illustration workflows.

2.4 Color Geometry

Besides the albedo-shading decomposition, images can also be decomposed using color geometry. Such color-based decomposition is often used to analyze the layers or strokes in images. The mixture or variance of image color has been extensively studied [Aksoy et al. 2017; Finlayson and amd Robert B. Fisher 2017; Koyama and Goto 2018; Lin et al. 2017]. It is also possible for images to be recolored using decomposed color layers [Aharoni-Mack et al. 2017; Chang et al. 2015; Zhang et al. 2017]. Tan *et al.* [2016] is the seminal work to use convex-hull based color palettes for image recoloring. Shih *et al.* [2013] generate night-time landscapes using a data-driven color transform. Although our approach is based on analyzing the density of stroke history, our algorithm does not explicitly decompose the image nor extract strokes, unlike image layer or stroke decomposition methods [Aksoy et al. 2017; Lin et al. 2017; Tan et al. 2018, 2016; Zhang et al. 2017]. On the contrary, our stroke density is directly estimated from pixel colors and image palettes, which reduces the inaccuracy caused by the hyper-parameters in image decomposition processing.

3 APPROACH

Our algorithm is based on a key assumption: when painting lighting effects, the artists' newly painted strokes are related to the stroke history of the painting. Pixels with relatively denser stroke history tend to gather more newly painted lighting or shadow strokes, whereas pixels with relatively sparse stroke history are often shaded with less newly painted effect strokes. Based on this principle, our algorithm first estimates the density of stroke history and then generates the lighting effects.

3.1 Estimating Stroke Density

First of all, we consider the simple problem of blending two strokes:

$$\boldsymbol{c} = \alpha_1 \boldsymbol{c}_1 + \alpha_2 \boldsymbol{c}_2 \quad \text{s.t.} \quad \alpha_1, \alpha_2 > 0, \quad \alpha_1 + \alpha_2 = 1 \tag{1}$$

where $\{\boldsymbol{c}_1, \boldsymbol{c}_2\} \in \mathbb{R}^3$ are the stroke colors, $\{\alpha_1, \alpha_2\} \subset \mathbb{R}$ are the blending weights, and $\boldsymbol{c} \in \mathbb{R}^3$ is the generated color. In this case, we define the stroke density k as:

$$k = \frac{1 - \sqrt{\alpha_1^2 + \alpha_2^2}}{1 - \frac{1}{\sqrt{2}}} \tag{2}$$

where we can see that if the two strokes are equally blended ($\alpha_1 = \alpha_2 = 0.5$), then k becomes 1. On the contrary, if only one stroke color is visible (*e.g.*, $\alpha_1 = 0, \alpha_2 = 1$), then k becomes 0. In this way, the stroke density measures to what extent the stroke colors are visible. The stroke density increases when multiple stroke colors are visible, and it decreases to zero when only one stroke color can be observed.

Similarly, we consider a set C including n stroke colors and a set $\mathcal{A}$ including n blending weights:

$$C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n\} \quad \text{and} \quad \mathcal{A} = \{\alpha_1, \ldots, \alpha_n\} \tag{3}$$

where each $\boldsymbol{c}_i \in \mathbb{R}^3$ is a stroke color and each $\alpha_i \in \mathbb{R}$ is a blending weight. The blending of all these stroke colors can be formulated as:

$$\boldsymbol{c} = \sum_{i=1}^{n} \alpha_i \boldsymbol{c}_i, \quad \text{s.t.} \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i > 0, \ i = 1, \ldots, n \tag{4}$$

where $\boldsymbol{c}$ is the generated color. In this case, the stroke density k can be written with Eq. (2) as:

$$k = \frac{1 - \sqrt{\sum_{i=1}^{n} \alpha_i^2}}{1 - \frac{1}{\sqrt{n}}} \tag{5}$$

When all strokes are equally blended ($\alpha_i = \frac{1}{n}$), then k becomes 1. On the contrary, if no strokes are blended and only one color is visible (*e.g.*, $\alpha_1 = 1$), then k becomes 0. The stroke density increases

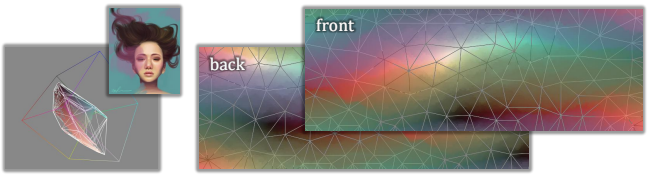


Fig. 3. **Palette extraction.** The RGB-convex hull of the input image is regarded as the palette. We visualize all colors in the front side (looking from the white position) and the back side (looking from the black position) of the mesh using RizomUV unfolding tool.

when more stroke colors can be observed, and decreases when less stroke colors are visible.

We then solve the problem of stroke density map estimation given a palette. In this paper, a palette refers to a specific set of colors used to paint a target image. In other words, all pixel colors of a given image should be reproducible by mixing or sampling the colors from its palette. Here we use a convex-hull-based palette because its dense and continuous mesh geometry can significantly meliorate and simplify our stroke density computation, which will be discussed later.

To identify such mesh-like palette, we compute the RGB-space convex hull of all observed pixel colors using the quick hull algorithm [Bradford et al. 1996; Tan et al. 2016] as

$$\mathcal{M} = \mathrm{QH}_3\left(\{\, \boldsymbol{c}_i \mid i = 1, \ldots, W \times H \,\}\right) \tag{6}$$

where $\boldsymbol{c}_i \in \mathbb{R}^3$ is an observed pixel color, $\mathrm{QH}_3(\cdot)$ is the function to compute the 3D convex hull, and $\mathcal{M}$ is the mesh of the convex hull, including all faces and all vertices. The convex hull is a tight wrapping of the colors, and, as shown in Fig. 3, the colors on the surfaces of the convex hull are regarded as the palette. The palette is a mesh in RGB space, and the colors on it can be visualized by flattening the mesh into a 2D surface using UV unfolding.

In order to estimate the stroke density, our main idea is to discover possible color combinations in the palette that could reproduce each pixel color, and then compute the stroke density using the blending weights of the discovered combinations. we first randomly sample n different points $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n\}$ in the palette $\mathcal{M}$. The more points are sampled, the more accurate the estimation will be. We denote the quantity of these points as n. We will later approximate $n \to +\infty$ for optimal accuracy. First of all, the barycentre $\boldsymbol{g}$ of these randomly sampled points can be formulated as:

$$\boldsymbol{g} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{c}_i \tag{7}$$

For each pixel position p, the input of our estimation algorithm is a pixel color $\boldsymbol{c}_p \in \mathbb{R}^3$, and the output is the corresponding stroke density $k_p \in \mathbb{R}$. For each color $\boldsymbol{c}_p$, we cast a ray from $\boldsymbol{g} \in \mathbb{R}^3$ to $\boldsymbol{c}_p \in \mathbb{R}^3$ in the RGB-space. Such a ray will intersect with the surface of $\mathcal{M}$ at one point, namely $\boldsymbol{h}_p \in \mathbb{R}^3$. Given $\{\boldsymbol{g}, \boldsymbol{c}_p, \boldsymbol{h}_p\}$ are collinear,

Algorithm 1 Stroke density map computation.

Input: An image consist of RGB vectors $\{\boldsymbol{c}_{1,\ldots,H\times W}\}$, where all pixel RGB color $\boldsymbol{c}_x = (r_x, g_x, b_x)$;
Output: A map K of the pixel-wise stroke density $\{k_{1,\ldots,H\times W}\}$;

1: $\{\mathcal{V}, \mathcal{F}\} \leftarrow \mathit{ConvexHull}(\{\boldsymbol{c}_{1,\ldots,H\times W}\})$; // Calculate the RGB convex hull, where $\mathcal{V}$ is a set of the vertices on the hull and $\mathcal{F}$ is a set of the faces on the hull.
2: $\boldsymbol{g} \leftarrow \frac{1}{s}\sum_{i=1}^{n}\frac{s_i}{3}\sum_{t=1}^{3}\mathcal{F}_{i,t}$; // Calculate the barycentre $\boldsymbol{g}$, where $\mathcal{F}_{i,t}$ means the t-th vertex in the i-th triangle in $\mathcal{F}$, s means the total area of the convex hull, and s_i means the area of the i-th triangle.
3: $\mathcal{M} \leftarrow \mathit{Mesh}(\mathit{vertices} = \mathcal{V}, \mathit{faces} = \mathcal{F})$; // Build up the mesh $\mathcal{M}$ in RGB space.
4: $\mathcal{R} \leftarrow \mathit{Ray}(\mathit{position} = \boldsymbol{g}, \mathit{directions} = \{\frac{\boldsymbol{c}_1-\boldsymbol{g}}{|\boldsymbol{c}_1-\boldsymbol{g}|}, \ldots, \frac{\boldsymbol{c}_{H\times W}-\boldsymbol{g}}{|\boldsymbol{c}_{H\times W}-\boldsymbol{g}|}\})$; // Cast rays in RGB space.
5: $\mathcal{H} \leftarrow \mathit{Intersection}(\mathcal{R}, \mathcal{M})$; // Calculate the hit points between the ray set $\mathcal{R}$ and the mesh $\mathcal{M}$, where $\mathcal{H} = \{\boldsymbol{h}_{1,\ldots,H\times W}\}$.
6: $K \leftarrow \{\frac{|\boldsymbol{c}-\boldsymbol{h}_1|}{|\boldsymbol{g}-\boldsymbol{h}_1|}, \ldots, \frac{|\boldsymbol{c}-\boldsymbol{h}_{H\times W}|}{|\boldsymbol{g}-\boldsymbol{h}_{H\times W}|}\}$;
7: **return** $K = \{k_{1,\ldots,H\times W}\}$;

the color of $\boldsymbol{c}_p$ can be formulated as:

$$\begin{aligned}\boldsymbol{c}_p &= \left(1 - \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\right)\boldsymbol{h}_p + \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\boldsymbol{g} \\ &= \left(1 - \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\right)\boldsymbol{h}_p + \frac{1}{n}\sum_{i=1}^{n}\frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\boldsymbol{c}_i\end{aligned} \tag{8}$$

where $|\cdot|$ is the Euclidean distance. Then, the given color $\boldsymbol{c}_p$ can then be written as color blending with Eq. (3)

$$C = \{\boldsymbol{c}_0 = \boldsymbol{h}_p\} \cup \{\boldsymbol{c}_i \mid i = 1 \ldots n\} \tag{9}$$

$$\mathcal{A} = \{\alpha_0 = \left(1 - \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\right)\} \cup \{\alpha_i = \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{n|\boldsymbol{g} - \boldsymbol{h}_p|} | i = 1 \ldots n\} \tag{10}$$

The stroke density k_p can then be computed from the blending weights $\mathcal{A}$ at the limit where $n \to +\infty$ as $\boldsymbol{g}$ approaches the barycentre of $\mathcal{M}$ at:

$$\begin{aligned}k_p &= \lim_{n\to+\infty}\frac{1 - \sqrt{\left(1 - \frac{|\boldsymbol{c}_p-\boldsymbol{h}_p|}{|\boldsymbol{g}-\boldsymbol{h}_p|}\right)^2 + \sum_{i=1}^{n}\left(\frac{|\boldsymbol{c}_p-\boldsymbol{h}_p|}{n|\boldsymbol{g}-\boldsymbol{h}_p|}\right)^2}}{1 - \frac{1}{\sqrt{n+1}}} \\ &= 1 - \sqrt{\left(1 - \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\right)^2 + \lim_{n\to+\infty}\frac{1}{n^2}\sum_{i=1}^{n}\left(\frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\right)^2} \\ &= 1 - \left(1 - \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\right) = \frac{|\boldsymbol{c}_p - \boldsymbol{h}_p|}{|\boldsymbol{g} - \boldsymbol{h}_p|}\end{aligned} \tag{11}$$

We can see this allows the stroke density to be estimated easily given the pixel color $\boldsymbol{c}_p$ at position p, barycentre of the convex hull $\boldsymbol{g}$, and the intersection with the ray cast from $\boldsymbol{g}$ to $\boldsymbol{c}_p$ denoted as $\boldsymbol{h}_p$.

As shown in Fig. 4, this theoretical result is consistent to our previous claims: when all the colors in the palette are equally mixed, the k becomes 1 because the resulting color $\boldsymbol{c}_p$ is exactly the barycentre of the palette mesh $\mathcal{M}$ and $|\boldsymbol{c}_p - \boldsymbol{h}_p| = |\boldsymbol{g} - \boldsymbol{h}_p|$. On the contrary, when the resulting color is exactly on $\mathcal{M}$, the k decreases to zero

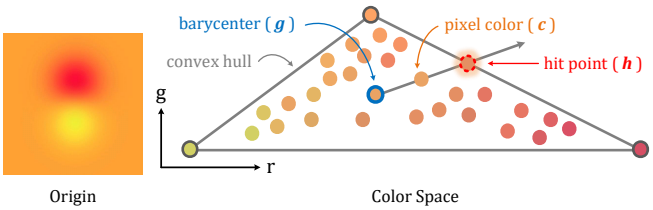


Fig. 4. **Visualization of the RGB-space geometry.** A color point c can be described in function of the convex hull barycentre g and the intersection of a ray cast from g to c and the convex hull surface which we denote as p.

because $|\boldsymbol{c}_p - \boldsymbol{h}_p| = 0$. Finally, we provide a summarized version of the stroke density estimation algorithm in Alg. 1.

3.2 Generating Lighting Effect

Our lighting effect generating algorithm is designed to mimic the artists' lighting effect workflow. As shown in Fig. 5, our proposed algorithm shares many similarities with the standard artist workflow. Screenshots of the artists' original illustration stroke history are shown in Fig. 5-(a) and the measured stroke density is presented in Fig. 5-(b). To obtain the effects, artists first paint the coarse effect layer and then retouch it into a refined effect layer. Then, the effect is applied to the original illustration to achieve Fig. 5-(f).

In the workflow from Fig. 5-(a) to Fig. 5-(f), the artist is conditioned to only paint routine soft lighting effects without any other enhancements such as tone adjustment or adding additional details. Our algorithm is designed to simulate such effect. We would like to also point out that if the artists are asked to try best to finish all possible shading, it may take them hours to achieve the effects with their personal styles as shown in Fig. 5-(g).

Before we describe the detailed algorithm, we present several observations based on the artist's workflow. Firstly, we randomly sample patches in the painted effect in Fig. 5-(e). We divide these patches into two groups: low stroke density patches and high stroke density patches. We observe that areas with relative low stroke density generally correspond to ambient and smooth colors, whereas areas with relatively high stroke density are likely to be painted with highlights or shadows. We will present this experiment in detail later in §4.2.

Based on this observation, we generate a lighting effect map by mimicking artists' coarse-to-fine workflow. Given a single image in Fig. 5-(h) without available stroke history, we directly estimate the stroke density in Fig. 5-(j). Afterwards, using the normalized channel intensity map, we generate a coarse effect map, which can then be refined to achieve the final lighting effect. Finally, the lighting effects are applied to the illustration as shown in Fig. 5-(n).

In particular, we here discuss the case of a point light source. Our objective is to compute a lighting effect map $\boldsymbol{S} \in \mathbb{R}^{W\times H\times 3}$ from the input image $\boldsymbol{R} \in \mathbb{R}^{W\times H\times 3}$, stroke density map $\boldsymbol{K} \in \mathbb{R}^{W\times H}$, light source $\boldsymbol{l} \in \mathbb{R}^3$, light intensity $\gamma \in \mathbb{R}$, and ambient intensity $O \in \mathbb{R}$. We compute the lighting effects for each channel of the image separately, and denote a channel c as of the original image and lighting effect map as $\boldsymbol{R}_c \in \mathbb{R}^{W\times H}$ and $\boldsymbol{S}_c \in \mathbb{R}^{W\times H}$, respectively.

3.2.1 Normalization. To avoid the high-frequency noise, we calculate the low-frequency normalized channel intensity $\boldsymbol{N}_c$ of the original image channel $\boldsymbol{R}_c$ with

$$\boldsymbol{N}_c = \frac{\rho(\boldsymbol{R}_c) - \min(\rho(\boldsymbol{R}_c))}{\max(\rho(\boldsymbol{R}_c)) - \min(\rho(\boldsymbol{R}_c))} \tag{12}$$

where $\rho(\cdot)$ is a Gaussian filter. For the Gaussian filter we use a 64×64 pixel window and $\sigma = 16$ for 512px images.

3.2.2 Coarse Effect Generation. We then generate the coarse lighting effect $\boldsymbol{E}$ shown in Fig. 5-(l). The motivation of this step is to create an initial draft of the lighting effects, which is similar to artists' coarse effect composition in Fig. 5-(c). Because our coarse effect is only used to provide low-frequency features for further refinements, it is not necessary to create effects that are consistent with artists' coarse effect. This step also produces the color influence of nearby objects, *e.g.*, the red hair in Fig. 5-(d) can influence the face color with red or blue color offsets on the face boundary.

We compute the channels $\boldsymbol{E}_i$ in the coarse lighting effect map $\boldsymbol{E}$ in a pixel-wise fashion. We build up a xyz coordinate system using the x-axis and y-axis of original image and a new z-axis perpendicular to the image panel. We denote the position of the light source as $\boldsymbol{l} = [l_x, l_y, l_z]^\intercal$. For any pixel location $\boldsymbol{p} = [p_x, p_y]^\intercal$, we define a distance p_d and angle θ as

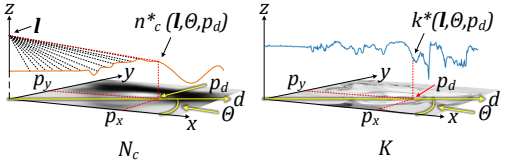
$$p_d = \sqrt{(p_y - l_y)^2 + (p_x - l_x)^2} \tag{13}$$

and

$$\sin\theta = \frac{p_y - l_y}{p_d}, \quad \cos\theta = \frac{p_x - l_x}{p_d} \tag{14}$$

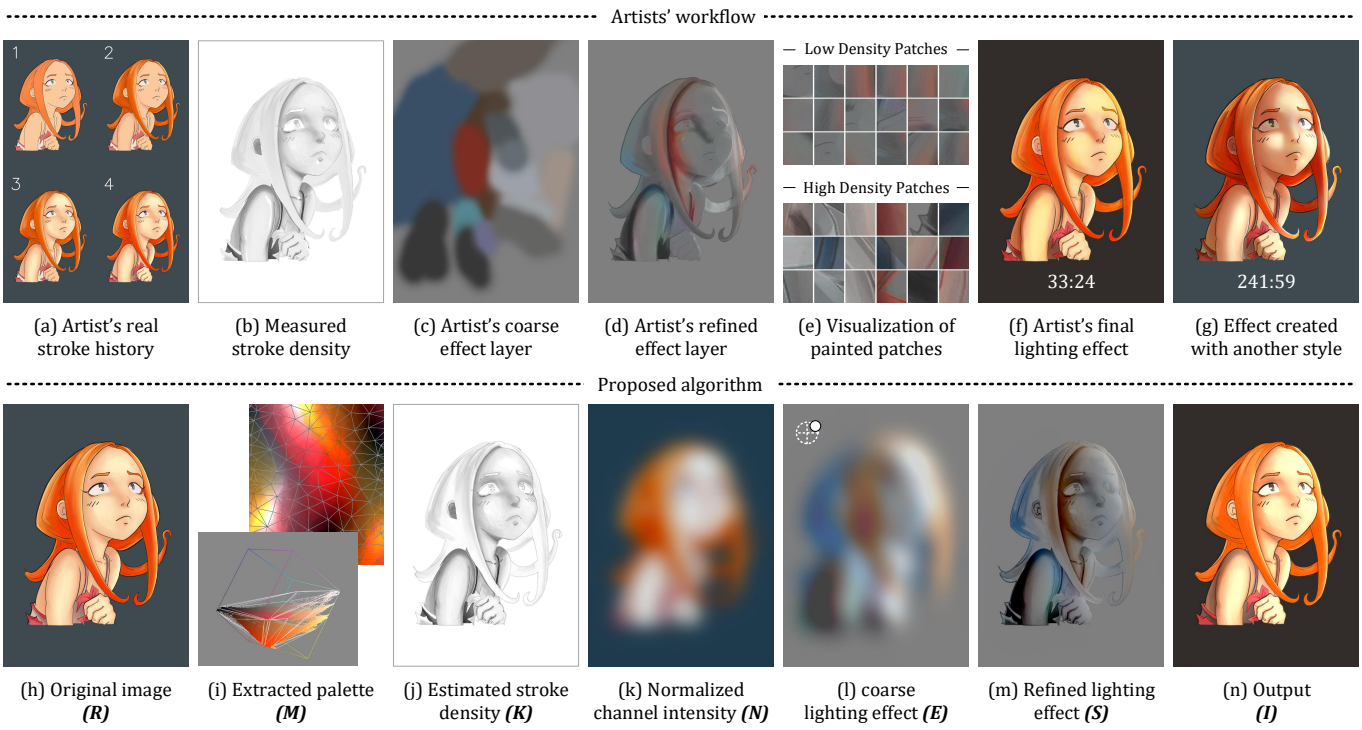
We then define the following wave functions

$$e_c^*(\boldsymbol{l}, \theta, p_d) = e_c(l_x + p_d\cos\theta,\ l_y + p_d\sin\theta) \tag{15}$$
$$n_c^*(\boldsymbol{l}, \theta, p_d) = n_c(l_x + p_d\cos\theta,\ l_y + p_d\sin\theta) \tag{16}$$
$$k^*(\boldsymbol{l}, \theta, p_d) = k(l_x + p_d\cos\theta,\ l_y + p_d\sin\theta) \tag{17}$$



where $e_c(x, y)$, $n_c(x, y)$, and $k(x, y)$ are the pixel values at the location $[x, y]^\intercal$ for the coarse lighting effect map $\boldsymbol{E}_c$, normalized input image channel $\boldsymbol{N}_c$, and stroke density map $\boldsymbol{K}$, respectively. We observe that if we fix $\boldsymbol{l}$ and θ, these functions can be viewed as waves that only depend on p_d. We use bilinear interpolation when the coordinates x and y are not integer values.

Each peak of the wave function $n_i^*(\cdot)$ has a side facing the light source and a side facing away from the light source. We can increase the intensity of the front side and reduce the intensity of the back side to simulate a light source. Similar to the *shape-from-shadow* algorithm [Kender and Smith 1984], all values in $\boldsymbol{E}_c$ can be written



(a) Artist's real stroke history (b) Measured stroke density (c) Artist's coarse effect layer (d) Artist's refined effect layer (e) Visualization of painted patches (f) Artist's final lighting effect (g) Effect created with another style

(h) Original image ***(R)*** (i) Extracted palette ***(M)*** (j) Estimated stroke density ***(K)*** (k) Normalized channel intensity ***(N)*** (l) coarse lighting effect ***(E)*** (m) Refined lighting effect ***(S)*** (n) Output ***(I)***

Fig. 5. **Comparison of the artist workflow with the proposed approach.** In the first row is components in the artist' manual workflow, and in the second row is components in the pipeline approach of our algorithm.

in pixel wise as:

$$e_i(p_x, p_y) = e_i(l_x + p_d \cos\theta,\ l_y + p_d \sin\theta) = e_i^*(\boldsymbol{l}, \theta, p_d) = \underbrace{\frac{[n_i^*(\boldsymbol{l}, \theta, \Delta p_d + p_d) - n_i^*(\boldsymbol{l}, \theta, p_d),\ \Delta p_d]}{|[n_i^*(\boldsymbol{l}, \theta, \Delta p_d + p_d) - n_i^*(\boldsymbol{l}, \theta, p_d),\ \Delta p_d]|}}_{\text{wave}} \cdot \underbrace{\left(\frac{[l_z,\ p_d]^\intercal}{|[l_z,\ p_d]|}\right)}_{\text{light}} \tag{18}$$

where Δp_d is a scaling scalar, which we set to $\Delta p_d = 1$ in all our experiments. Every unknown pixel value in the coarse lighting effect map $\boldsymbol{E}_c$ is estimated by calculating the dot production of the wave direction unit vector and the light direction unit vector.

3.2.3 Refinement. When artists are painting lighting effects, pixels with relatively low stroke density tend to be shaded with ambient gray, whereas pixels with relatively high stroke density are likely to gather diversified highlight or shadow colors. Based on this principle, we refine the coarse lighting effect map using the stroke density map with

$$\boldsymbol{S}_i = \gamma\, \boldsymbol{E}_i \odot \boldsymbol{K} + O \tag{19}$$

where $\odot$ is the Hadamard product, and we recommend to set default ambient intensity to $O = 0.55$. This refinement step can also be calculated pixel-wise as

$$\begin{aligned} s_i(p_x, p_y) &= s_i(l_x + p_d \cos\theta,\ l_y + p_d \sin\theta) \\ &= \gamma\, e_i^*(\boldsymbol{l}, \theta, p_d)\ k^*(\boldsymbol{l}, \theta, p_d) + \boldsymbol{O} \end{aligned} \tag{20}$$

where $s_c(p_x, p_y)$ is the value in $\boldsymbol{S}_i$ at $\boldsymbol{p} = [p_x, p_y]^\intercal$. This refinement encourages the shading values to stay away from the ambient intensity when the pixel stroke density is high. On the contrary, if some pixels are of low stroke density, this refinement discourages the shading, and these pixels will be shaded with the ambient intensity. Finally, the lighting effect can be multiplied to the original image with

$$\boldsymbol{I} = \boldsymbol{R} \odot \boldsymbol{S} \tag{21}$$

where $\boldsymbol{I} \in \mathbb{R}^{W \times H \times 3}$ is the rendered result. In cases where the light source has colors, the channels in $\boldsymbol{S}$ can be multiplied with the light color. Additionally, this algorithm also supports other types of light sources, *e.g.*, lights at infinity or spot lights. Please refer to our supplementary material for other types of the light sources.

4 EXPERIMENTS

In this section, we validate the assumptions about the colors, strokes, and painted lighting effects. Additionally, we provide quantitative and qualitative analysis which includes perceptual user studies.

4.1 Qualitative Results

We compare our stroke density maps estimated on single images to the real stroke density maps measured with artists' stroke history, and, simultaneously, we compare our generated lighting effects to those created manually by artists, as shown in Fig. 6. In particular, we conduct two user studies by dividing the participated artists into two groups: unconditioned group and conditioned group.

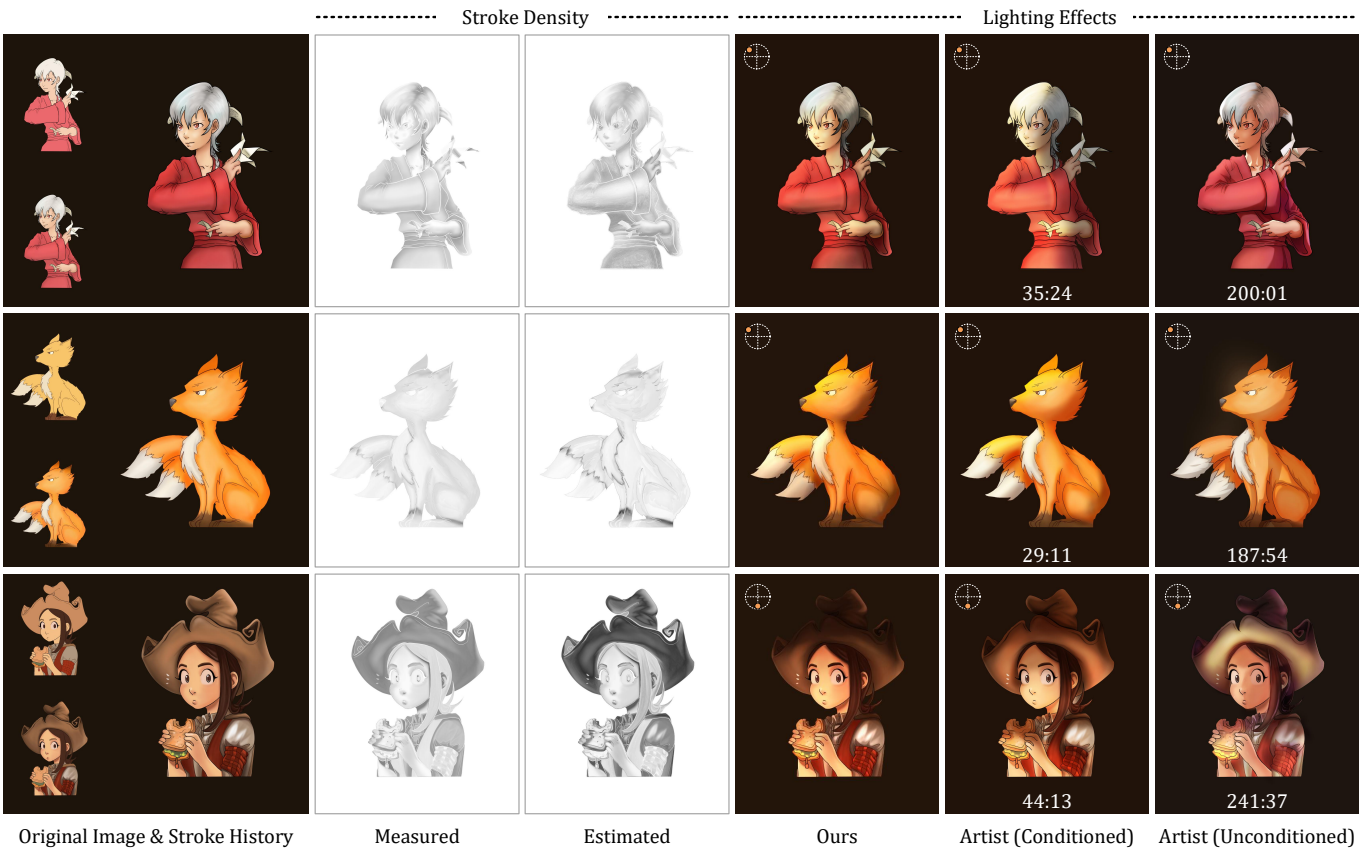


Fig. 6. **Comparison on stroke density and lighting effects.** We compare our estimated stroke density to the real stroke density measured on the stroke history, and, at the same time, compare our generated lighting effects to artists' manually created ones.

In the unconditioned group, we do not show our tool to the artists and directly ask them to try best to relight illustrations according to some simple lighting directions. We use results from this group to analyze to what extent our synthesized effects are similar to those created manually by artists.

In the conditioned group, we allow artists to test our tool with arbitrary images to familiarize themselves with our effect style, and we then ask them to paint effects with similar styles. We use results in this group to exploit how long time does it take for the artists to create effects that have similar style to our generated ones.

It is notable that the artists, especially those in the conditioned group, are prohibited from using our tools when painting, in order to ensure that they are drawing with their own artistic visions instead of simply reproducing our generated results. Furthermore, we would like to point out that the lighting effect creation is a labor-intensive task, and we report the recorded painting time for each manually created effect in corresponding figures. We also include comparison with multiple lighting direction in Fig. 7. Although the current algorithm cannot replace the manually created effects which takes artists hours to create, *i.e.*, the artist results in the unconditioned group are generally better than the algorithm results, the majority of our generated effects are of satisfactory visual quality and are acknowledged by those professional artists as usable. Finally, results in the conditioned group show that it take artists about 30 minutes to create effects with the style similar to ours.

4.2 Connection between Colors, Strokes, and Manually Painted Artistic Lighting Effects

As shown in Fig. 8, artists are invited to paint the example illustrations, and we record their strokes to compute the ground truth stroke density, and, simultaneously, estimate the stroke density using the single canvas images as each stroke is drawn. Finally, we visualize the error between the ground truth and the estimation, and report the Peak Signal to Noise Ratio (PSNR). The raw data of this experiment is provided in the supplementary material.

During the digital painting process, our color-geometry-based stroke density estimation algorithm becomes increasingly accurate because more and more colors in the canvas are evenly distributed in the RGB color space. Although this estimation is not a perfect measurement, it yields visually similar stoke density maps and can achieve PSNR scores at roughly 26, which demonstrates that this algorithm is a reliable source of stoke density estimation.

Moreover, as shown in Fig. 9, we study the connection between artists' stroke history and their painting effects. We invite artists to

Fig. 7. **Comparison on lighting effects with multiple directions.** We compare our generated effects against the manually created lighting effects painted by the artists. Besides, we present artists' recorded painting time in each manually created examples.

manually paint artistic lighting effects, and then randomly sample patches over their created effects in two groups: low stroke density group and high stroke density group. These patches are sampled by simply using a binary threshold to divide the stroke density map into low density part and high density part. Random pixel positions are then sampled and used as the central position of each patch. In order to avoid the bias caused by the background and the stroke density offsets between large objects, we ask artists to manually create background masks to exclude background patches, and then use adaptive threshold[1] when we binarize the stroke density maps.

In the visualization, we can see that patches with low stroke density are likely to be painted with relatively ambient, smooth, and uniform effect color, whereas pixels with high stroke density tends to gather sharp and intensive effect details and diversified lighting and shadow. In order to highlight this observation, we also calculate the mean standard deviation of these two patch groups, and we find that areas with high stroke density have significantly larger standard deviation than that of low stroke density. This evidence

[1] *THRESH_GAUSSIAN_C* in OpenCV.

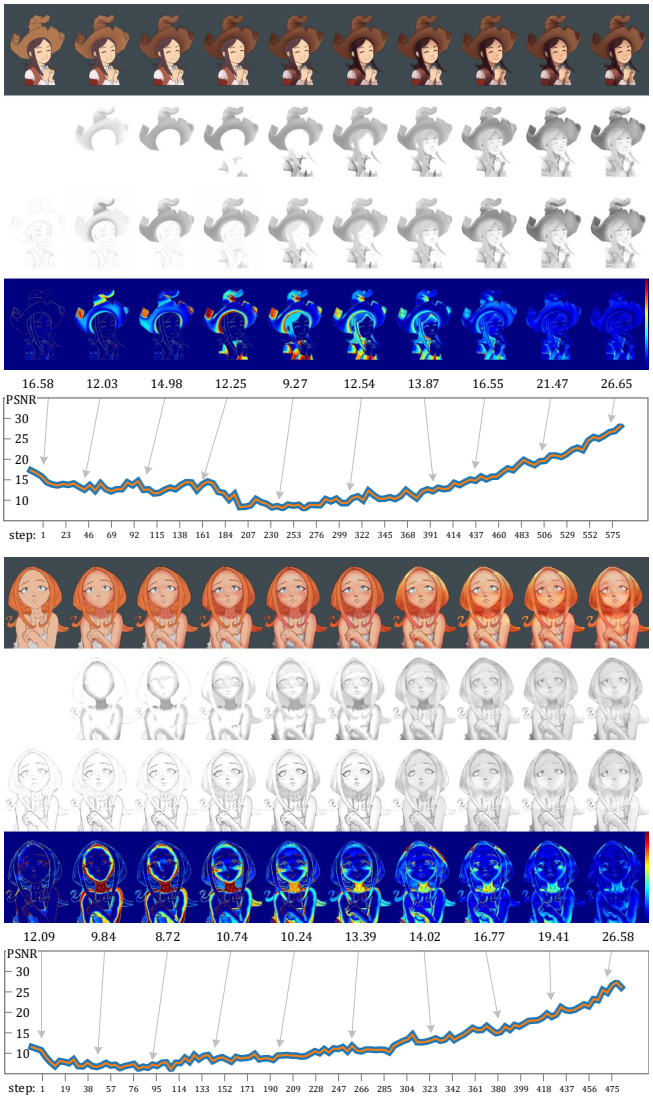


Fig. 8. **Stroke density estimation accuracy.** We validate our stroke density estimation algorithm by comparing the estimated stroke density to the real stroke density measured from recorded stroked history. For each example, in the first row we show the artists' canvases, in the second row we show the measured stroke density computed on the stroke history, in the third row we show the estimated stroke density calculated on single canvas images, and in the fourth row we show the error map between the ground truth and the estimation. We also illustrate the diagrams of this error in the last row of each example, where the x-axis is the painting stroke steps, and the y-axis is the PSNR accuracy.

further demonstrates that areas with relatively high stroke density tends to gather diversified lighting and shadow effects.

4.3 Significance of Using Stroke Density

We conduct experiments to study possible alternatives to the stroke density, as shown in Fig. 10. Firstly, we present results from a line

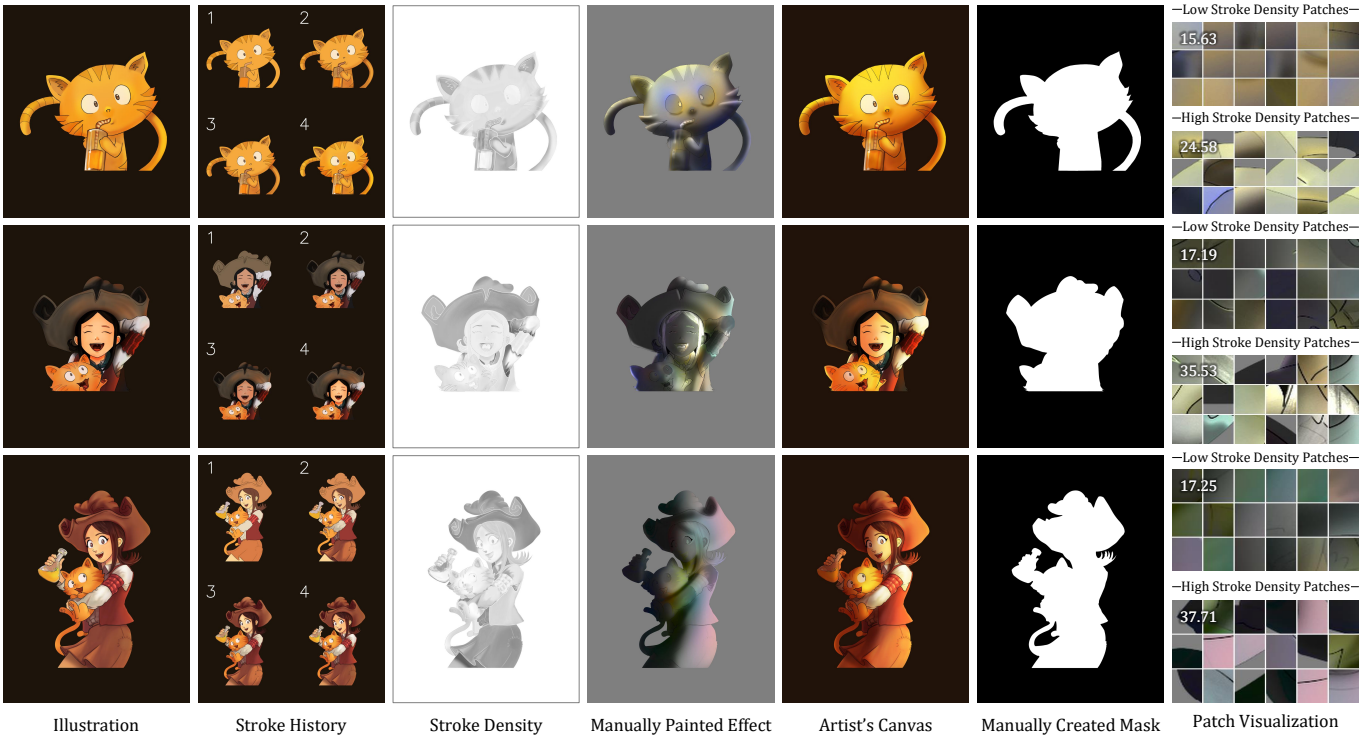


Fig. 9. **Relationship between stroke density and painted lighting effects.** We study the relationship between the density of stroke history and the manually painted lighting effects. From left to right is the original image, screenshots of the stroke history, stroke density measured on the real stroke history, lighting effects created by artists, artists' final canvases, background masks manually created by artists, and the patches sampled from the painted effect. We also present the mean standard deviation of all pixel colors in each patch group at the left-top corner of each group in the last col.

drawing normal map estimation method *DeepNormal* [Hudon et al. 2018], and then use a sprite rendering tool *Unity SpriteLamp* to produce the baseline results. This method requires a line drawing and a mask for each illustration, and we invite artists to manually create the inking lines and the masks. Although this method can perform relighting, it is not well-suited for digital painting because many distortions of the original illustration details can be observed.

We also replace the stroke density map $\boldsymbol{K}$ in our algorithm with the alternative maps, *e.g.*, the luminance map and the background map as shown in Fig. 10, where the lighting effects becomes unusable. Finally, we invite artists to manually create some simple gradient maps to replace the lighting effect $\boldsymbol{S}$ in our algorithm. We observe that these simple gradients are only able to generate very simple and non-natural looking illumination. Compared to these alternatives, our algorithm is the best choice to achieve effects that are visually similar to the artists' manually painted ones. We also include more alternative map formulations as well as animated ablative study in the supplementary material.

4.4 Comparison with Non-digital-painting Relighting

As our approach does not rely on the real digital painting stroke history, it is applicable to images even not created using strokes. We show several results comparing with the surface optimization approach of [Wu et al. 2008] and the CNN-based approach of [Yu and Smith 2019] in Fig. 11, where we can see how the optimization-based approach suffers from overly simple model fitting, while the CNN-based approach has trouble handling diversified image contents. We also show a visual comparison against the classic *height-to-normal* approach of [Kender and Smith 1984] using a Sobel filter, as shown in Fig. 12, where the height-based relighting results are not well-suited for lighting effects. Although our generated lighting effects may not strictly match the accurate physical structure of the 3D objects, these results are usable from human perception, and can help artists in the matte painting or image light editing workflow. Please also refer to the supplementary material for more animated results and interactive results.

In our generated effects, we can see that the low frequency domain plays the dominant role in directionality. We would like to point out that the high frequency domain is also important to make the generated effects naturally adjust to the image structure. In Fig. 13, we present results only using the low frequency coarse effect $\boldsymbol{E}$ without refinement and then compare it to the effect $\boldsymbol{S}$ with high frequency domain refined using our algorithm, where structural inconsistency artifacts can be found if the high frequency patterns are not retouched using our proposed method.

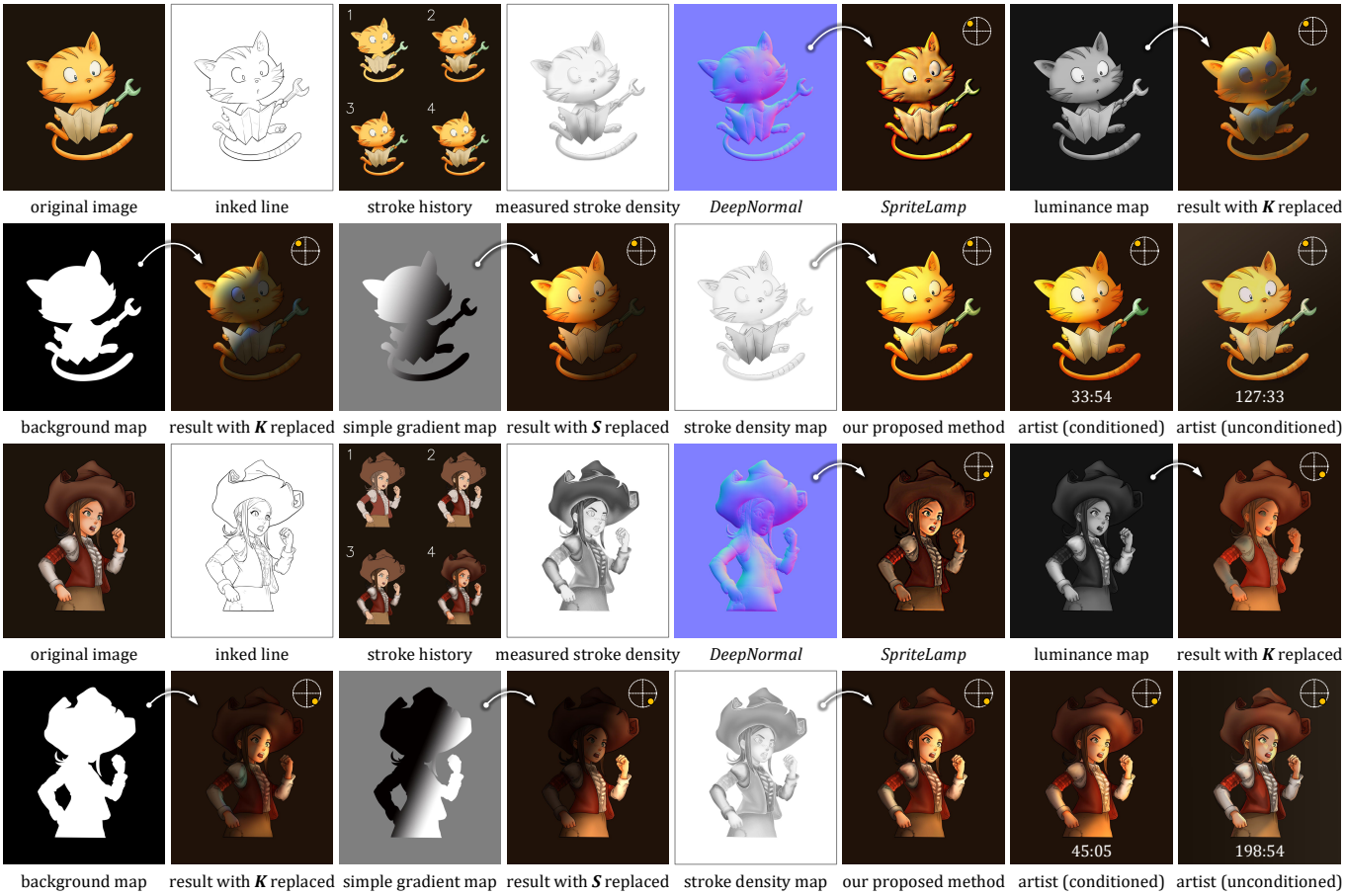


Fig. 10. **Ablation study.** We present ablation results from using other alternative algorithms without using stroke density maps, as well as results obtained from our algorithm by replacing stroke density maps with other alternative maps.

4.5 Perceptual User Study

In order to perceptually evaluate and compare our approach with existing methods, we perform a perceptual user study focusing on usability aspects of different relighting algorithms. In particular, 6 users participated and were asked to relight 10 images with various approaches. For each image, we randomly select one user to give new lighting condition instructions to the other users, and have each of the remaining five users relight the image with the new conditions, each using a different approach. Finally, the user that gave the new lighting condition ranks the results. Besides providing the ranking results, we measure the time it takes to process an image with each approach on average. Each approach is used in two stages which are evaluated separately. The first stage consists of using the approach as is, while the second stage consists of using professional photo editing software as post-processing. The two stages are ranked separately. Full details are provided in the supplemental material.

We compare our approach with a *shape-from-shadow*-based algorithm [Kender and Smith 1984], a surface optimization algorithm [Wu et al. 2008], and a recent deep learning-based approach [Yu and Smith 2019]. As professional photo editing software, we use Adobe PhotoShop Lighting Effects. Results are shown in Table 1. We find that users prefer our approach over all other approaches in both the first stage, using a single tool, and the second stage, post-processing with professional tools. Of the other approaches we find users prefer the professional tools, followed [Wu et al. 2008]. The low performance of [Yu and Smith 2019] is likely explained by the low generalization to general images caused by the learning procedure. We additionally find that our approach also provides a significant decrease in the interaction time. When including post-processing, our approach takes only roughly 50% of the time of using the professional photo editing software. We note that the other approaches take longer and obtain worse results than using only the professional photo editing software. This highlights the strengths of our approach, which is able to quickly and realistically relight a diversity of images.

4.6 Color Geometry Model Variations

We present experiments with different color geometry models and configurations. In particular, we replace our convex hull palette with the simplified hull of [Tan et al. 2018], and replace the estimated

Fig. 11. **Comparison with non-digital-painting relighting approaches.** For each input image we provide two results with different lighting directions, which is illustrated in each example.

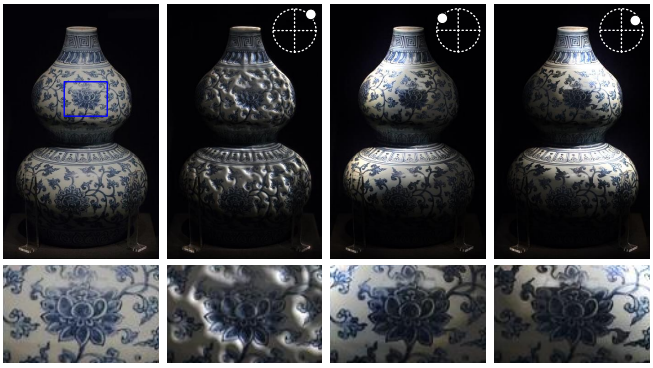


Fig. 12. **Comparison with height-to-normal-based relighting.** We compare our algorithm to the shape-from-shadow transform, where the per-pixel intensity is used as the height map.

Fig. 13. **Comparison to results without high frequency refinement.** We compare our results with those without high frequency refinement using stroke density maps. We directly use the coarse lighting effect $\boldsymbol{E}$ to shade the original image for the results in the second column, whereas the results in the third column is obtained using our proposed approach.

stroke density map with the calculated stroke density based on the actual layer decomposition from [Tan et al. 2016] in Fig. 14, where the results may suffer from *foggy* artifacts. We also replace the stroke density map with the image luminance map as shown in Fig. 15. Finally, we can pre-process noisy input images, *e.g.*, images with JPEG artifacts, with a CNN model, *e.g.*, SRCNN [Dong et al. 2015] to improve the quality of the estimated stroke density map, as shown in Fig. 16. Please refer to the supplemental material for additional ablative results.

4.7 Masked Lighting Effect Generation

In specific use cases where the input image contains complicated background structure, or when separated lighting effects for the foreground objects are required, we can use third-party image segmentation methods, *e.g.*, Mask-RCNN [He et al. 2018], to mat independent object components, and then generate lighting effects for the separated segments. In particular, we can directly multiply masks to the coarse effect map $\boldsymbol{E}$. Because the stroke density maps show strong capability to refine coarse lighting effects, even simple and aliased masks predicted by Mask-RCNN may yield satisfactory results, as shown in Fig. 18.

Table 1. **Average Interaction Time and User Ranking.** We report the recorded interaction time and user ranking (1 to 5 indicates worst to best) of different algorithm and tool combinations. For the interaction time using the combination of professional tools and a single approach, we show the time corresponding to the professional tools and single approach separately. Best results are shown in bold.

Single Tool / Combinations	Time (s)	Preference Rank
[Kender and Smith 1984]	14.4	1.8 ± 0.4
[Wu et al. 2008]	18.1	3.0 ± 0.0
[Yu and Smith 2019]	24.6	1.2 ± 0.4
Ours	**7.2**	**4.0 ± 0.0**
Professional Tool (PT) only	46.6	4.1 ± 0.3
PT + [Kender and Smith 1984]	66.8 + 14.4	1.8 ± 0.4
PT + [Wu et al. 2008]	58.5 + 18.1	3.0 ± 0.0
PT + [Yu and Smith 2019]	73.7 + 24.6	1.2 ± 0.4
PT + Ours	**17.9 + 7.2**	**4.9 ± 0.3**

4.8 Additional Types of Light Sources

Our algorithm supports multiple types of light sources, which can be implemented as a plug-in for professional photo editing software, *e.g.*, Adobe Photoshop Lighting Effect Filter, allowing users to relight the image in a content-aware manner. Please refer to the supplementary for detailed formulations.

4.9 Limitations

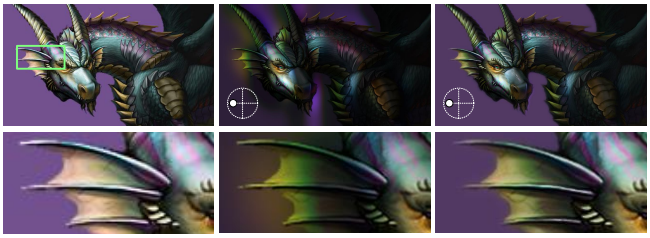
As shown in Fig. 17, our generated lighting effects may contain halo-like artifacts, but these artifacts are not visible in the final result. Besides, as shown in Fig. 19, our approach is unable to modify hard shadows. Complicated background object structures or backgrounds that are significantly brighter than the foreground objects may also interfered the wave transform. Nevertheless, in many cases, this limitation can be addressed by combining third-party image segmentation approaches as we mentioned before. Moreover, the directionality of this algorithm is limited to the front side of the image canvas, and it is unable to put a light source behind the image objects as this would require 3D scene understanding.

5 CONCLUSION

We have presented an algorithm to generate digital painting lighting effects to help simplify the current labor-intensive effect composition workflow in illustrations or matte paintings. We have studied the connection between the colors, strokes, and painted lighting effects, and developed an efficient coarse-to-fine approach based on estimating the stroke density of the illustration. Qualitative results,

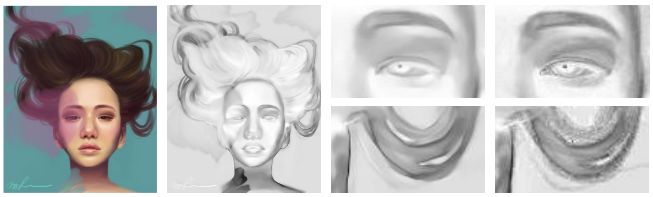
Original Image | Simplified Convex Hull | Decomposition-based Stroke Density | Proposed Approach

Fig. 14. **Comparison to results with other color geometry algorithm configurations.** We analyze the effect of replacing our convex-hull-based palette with the simplified hull proposed in [Tan et al. 2016], and also replacing our proposed stroke density estimation algorithm with the actual stroke decomposition and density calculation. In each case we visualize the lighting effect map and the final rendered result. The light position used is indicated in the original figure. Plane © Anonymous.



Original Image | Guided by Luminance | Guided by Stroke Density

Fig. 15. **Replacing stroke density maps with naive luminance maps.** We compare an image processed using our approach guided by luminance maps and stroke density maps.



(a) (b) (c) (d)

Fig. 16. **Influence of the pre-processing.** (a) The original image compressed with 0.85 JPEG quality. (b,c) The estimated stroke density map with pre-processing. (d) The result without pre-processing.

including a perceptual user study, corroborate the capability of our algorithm to help artists in their daily lighting effect workflows. We hope that our research will stimulate more researches related to artistic lighting effect creation.

ACKNOWLEDGMENTS

We would like to thank Chengze Li for the helpful discussions. We would like to thank all our reviewers for their insightful, constructive, and high-quality comments. This work is supported by National Natural Science Foundation of China (61972059, 61773272), The Natural Science Foundation of the Jiangsu Higher Education Institutions of China (19KJA230001), Suzhou Technology Development Plan (Key Industry Technology Innovation-Prospective Application Research Project SYG201807), Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (93K172016K08), the Priority Academic Program Development of Jiangsu Higher Education Institutions. This work is also supported by JST PRESTO (Simo-Serra, Grant Number: JPMJPR1756).

Original Image | Lighting Effect | Output

Fig. 17. **Invisible halo-like artifacts.** Some halo-like Artifacts, as marked with the red arrow, can be found surrounding the leaves in the lighting effects in the second column. However, in the final result of the third column, these artifacts become invisible.

REFERENCES

Elad Aharoni-Mack, Yakov Shambik, and Dani Lischinski. 2017. Pigment-Based Recoloring of Watercolor Paintings. *NPAR* (2017).

Yagiz Aksoy, Tung Ozan Aydin, Aljosa Smolic, and Marc Pollefeys. 2017. Unmixing-based soft color segmentation for image manipulation. *ACM Transactions on Graphics* (2017).

Yagiz Aksoy, Changil Kim, Petr Kellnhofer, Sylvain Paris, Mohamed Elgharib, Marc Pollefeys, and Wojciech Matusik. 2018. A Dataset of Flash and Ambient Illumination Pairs from the Crowd. *ECCV* (2018).

J.T. Barron and J. Malik. 2012. Color constancy and intrinsic images and shape estimation. *ECCV* (2012).

H. G. Barrow and J. M. Tenenbaum. 1978. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, A. Hanson and E. Riseman (Eds.). Academic Press, 3–26.

Sean Bell, Kavita Bala, and Noah Snavely. 2014. Intrinsic Images in the Wild. *ACM Transactions on Graphics* 33, 4 (2014).

Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. 2013. OpenSurfaces: A Richly Annotated Catalog of Surface Appearance. *ACM Transactions on Graphics* 32, 4 (2013).

Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. 2015. Material Recognition in the Wild with the Materials in Context Database. *CVPR* (2015).

Barber C. Bradford, Dobkin David P., and Huhdanpaa Hannu. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Software* (1996).

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

Fig. 18. **Additional application: masked lighting effect generation.** Our algorithm can be combined with simple foreground segmentation methods like Mask-RCNN to generate object-level lighting effects.

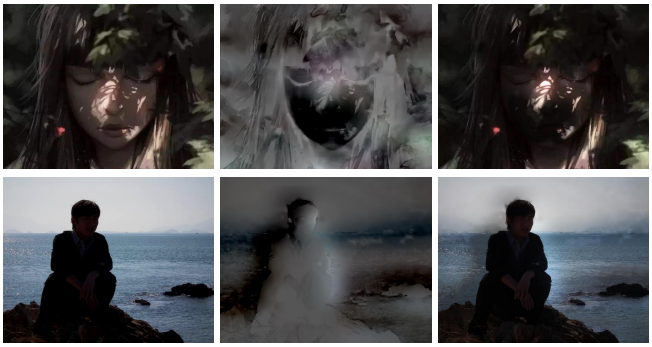


Fig. 19. **Limitations.** Our algorithm is easily interfered by hard shadows and is unable to modify them. Additionally, our algorithm may fail when the background is significantly brighter than the foreground. Nevertheless, this shortcoming can be meliorated when masks are applied to the proposed algorithm.

Jiansheng Chen, Guangda Su, Jinping He, and Shenglan Ben. 2010. Face Image Relighting using Locally Constrained Global Optimization. *ECCV* (2010).

Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. 2000. Acquiring the reflectance field of a human face. *the 27th annual conference on Computer graphics and interactive techniques* (2000).

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2015. Image Super-Resolution Using Deep Convolutional Networks. *TPAMI* (2015).

Graham Finlayson and Han Gong amd Robert B. Fisher. 2017. Color Homography: Theory and Applications. *TPAMI* (2017).

P.V. Gehler, C. Rother, M. Kiefel, L. Zhang, and B. Scholkopf. 2011. Recovering intrinsic images with a global sparsity prior on reflectance. *NIPS* (2011).

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. 2018. Mask R-CNN. *CVPR* (2018).

Matis Hudon, Rafael Pages, Mairead Grogan, and Aljosa Smolic. 2018. Deep Normal Estimation for Automatic Shading of Hand-Drawn Characters. *ECCV* (2018).

Yoshihiro Kanamori and Yuki Endo. 2018. Relighting humans: occlusion-aware inverse rendering for full-body human images. In *ACM Transactions on Graphics*.

John R. Kender and Earl Smith. 1984. Shape from Darkness: Deriving Surface Information from Dynamic Shadows. *National Conference on Artificial Intelligence* (1984).

Yuki Koyama and Masataka Goto. 2018. Decomposing Images into Layers with Advanced Color Blending. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 397–407.

Sharon Lin, Matthew Fisher, Angela Dai, and Pat Hanrahan. 2017. LayerBuilder: Layer Decomposition for Interactive Image and Video Color Editing. *Arxiv* (2017).

Wojciech Matusik, Matthew Loper, and Hanspeter Pfister. 2004. Progressively-Refined Reflectance Functions from Natural Illumination. *Eurographics Workshop on Rendering* (2004).

Pieter Peers and Philip Dutre. 2005. Inferring reflectance functions from wavelet noise. *the Sixteenth Eurographics conference on Rendering Techniques* (2005).

Pieter Peers, Dhruv K Mahajan, Bruce Lamond, Abhijeet Ghosh, Wojciech Matusik, Ravi Ramamoorthi, and Paul Debevec. 2009. Compressive light transport sensing. *ACM Transactions on Graphics* (2009).

Pieter Peers, Naoki Tamura, Wojciech Matusik, and Paul Debevec. 2007. Post-production Facial Performance Relighting using Reflectance Transfer. *ACM Transactions on Graphics* (2007).

Julien Philip, Michael Gharbi, Tinghui Zhou, Alexei, Efros, and George Drettakis. 2019. Multi-view Relighting using a Geometry-Aware Network. *SIGGRAPH* (2019).

Soumyadip Sengupta, Angjoo Kanazawa, Carlos D. Castillo, and David W. Jacobs. 2018. SfSNet: Learning Shape, Refectance and Illuminance of Faces in the Wild. In *CVPR*.

M. Serra, O. Penacchio, R. Benavente, and M. Vanrell. 2012. Names and shades of color for intrinsic image estimation. *CVPR* (2012).

Jianbing Shen, Xiaoshan Yang, Yunde Jia, and Xuelong Li. 2011. Intrinsic Images Using Optimization. *CVPR* (2011).

YiChang Shih, Sylvain Paris, Fredo Durand, and William T. Freeman. 2013. Data-driven Hallucination for Different Times of Day from a Single Outdoor Photo. *ACM Transactions on Graphics* (2013).

Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. 2018. Interactive Sketch-Based Normal Map Generation with Deep Neural Networks. *ACM on Computer Graphics and Interactive Techniques* 1, 1 (jul 2018), 1–17.

Tiancheng Sun, Jonathan T. Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. 2019. Single Image Portrait Relighting. *SIGGRAPH* (2019).

Daniel Sykora, Ladislav Kavan, Martin Cadik, Ondrej Jamriska, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. 2014. Ink-and-Ray: Bas-Relief Meshes for Adding Global Illumination Effects to Hand-Drawn Characters. *ACM Transactions on Graphics* 33, 2 (apr 2014), 1–15.

Jianchao Tan, Jose Echevarria, and Yotam Gingold. 2018. Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. *ACM Transactions on Graphics* (2018).

Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. 2016. Decomposing Images into Layers via RGB-space Geometry. *ACM Transactions on Graphics* (2016).

Tai-Pang Wu, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. 2008. Interactive normal reconstruction from a single image. *ACM Transactions on Graphics* 27, 5 (dec 2008), 1.

Tai-Pang Wu, Chi-Keng Tang, Michael S. Brown, and Heung-Yeung Shum. 2007. ShapePalettes: Interactive Normal Transfer via Sketching. *ACM Transactions on Graphics* (2007).

Qiuying Xu, Yotam Gingold, and Karan Singh. 2015. Inverse Toon Shading: Interactive Normal Field Modeling with Isophotes. *Sketch-Based Interfaces and Modeling* (2015).

Ye Yu and William A. P. Smith. 2019. InverseRenderNet: Learning single image inverse rendering. *CVPR* (2019).

Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. 2017. Palette-Based Image Recoloring Using Color Decomposition Optimization. *IEEE Transactions on Image Processing* (2017).