Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

An Improved Geometric Approach for Palette-based Image Decomposition and Recoloring

Yili Wang[1], Yifan Liu[1,2] and Kun Xu[1,2]*

[1] Tsinghua University, Beijing [2] Beijing National Research Center for Information Science and Technology
* Corresponding author, xukun@tsinghua.edu.cn

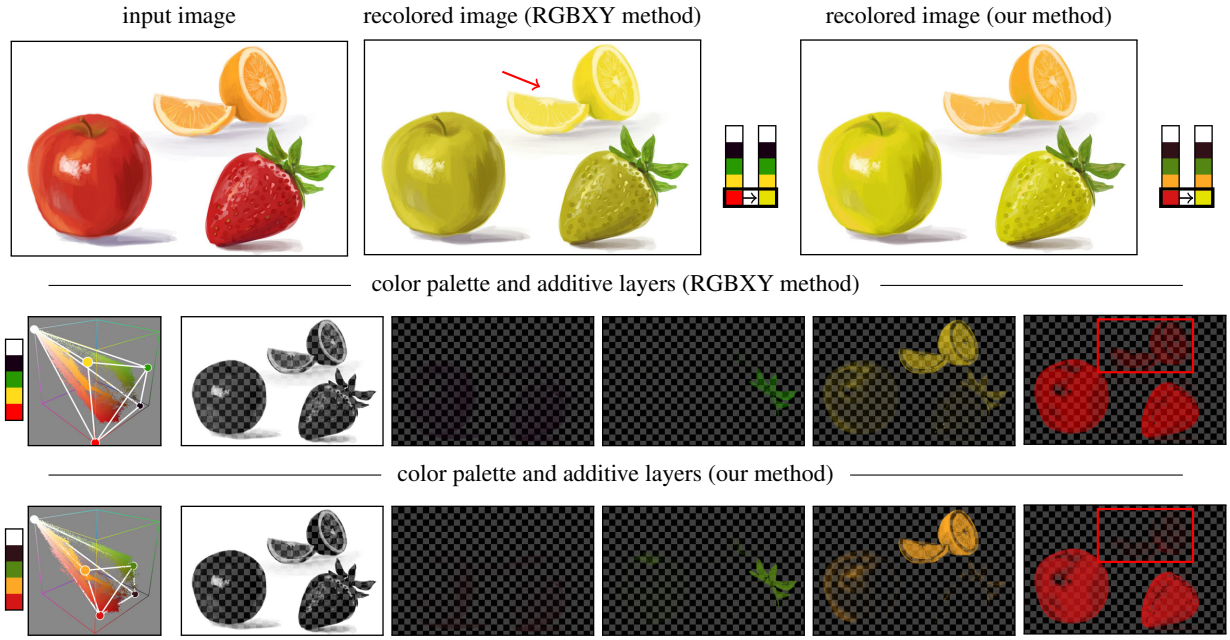


Figure 1: *Comparison of our method with the convex hull based RGBXY method [TEG18] for palette based image decomposition and recoloring. Our palette colors are more representative and our layer opacities are more sparse. For recoloring, to change the red apple and strawberry in the input image to 'light yellow', our method is able to complete the task by intuitively adjusting one palette color (i.e., 'red') due to the above advantages, however, the RGBXY method will inevitably introduce strong color bleeding artifacts to the orange.*

Abstract

Palette-based image decomposition has attracted increasing attention in recent years. A specific class of approaches have been proposed basing on the RGB-space geometry, which manage to construct convex hulls whose vertices act as palette colors. However, such palettes do not guarantee to have the representative colors which actually appear in the image, thus making it less intuitive and less predictable when editing palette colors to perform recoloring. Hence, we proposed an improved geometric approach to address this issue. We use a polyhedron, but not necessarily a convex hull, in the RGB space to represent the color palette. We then formulate the task of palette extraction as an optimization problem which could be solved in a few seconds. Our palette has a higher degree of representativeness and maintains a relatively similar level of accuracy compared with previous methods. For layer decomposition, we compute layer opacities via simple mean value coordinates, which could achieve instant feedbacks without precomputations. We have demonstrated our method for image recoloring on a variety of examples. In comparison with state-of-the-art works, our approach is generally more intuitive and efficient with fewer artifacts.

CCS Concepts

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

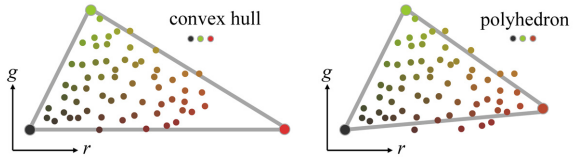
1. Introduction

Palette-based image decomposition [TLG17, AASP17, TEG18] has becoming more and more popular in recent years. They have demonstrated impressive results for image editing and provide an intuitive interface which is easy to learn and use for novice users. In these works, a *color palette*, which is a small set of colors, is first extracted from the image to represent its color distribution. The image is then decomposed into a set of layers, where each layer is a spatially varying opacity map modulated by a corresponding palette color, such that the image could be reproduced by blending all the layers. After layer decomposition, users could easily edit the appearance of the image by modifying palette colors, or manipulating the layers.

Among existing works on palette-based image decomposition, the convex hull based approaches [TLG17, TEG18] are state-of-the-art ones. They observe that image colors imply a geometric structure in the RGB-space and could be enclosed by an RGB-space convex hull, whose vertices correspond to a color palette. Based on this observation, they extract a small size color palette through iterative simplification of the convex hull. To decompose the image into layers, in a preprocessing stage, they build a 5D convex hull in the RGBXY space and tessellate the 5D convex hull into non-overlapping simplices, which typically takes a few minutes. At the run-time stage, layer opacity values could be efficiently computed through barycentric coordinates with respect to the tessellated 5D simplices. They also provide a tool to visualize the geometric structures of the color palette (i.e., the convex hull) and image colors in the RGB-space, helping users understand their geometric relationships. Compared to other works, they are able to produce the most satisfactory results considering both result quality and computation efficiency.

Despite the benefits, however, its extracted palette colors may have a low degree of representativeness. On the one hand, since the palette colors are vertices of RGB-space convex hull, they do not necessarily exist in the image. On the other hand, dominant image colors, which are in the middle of RGB space, could not be selected as palette colors due to geometric properties of convex hulls. Such an example is shown in Figure 1. The input image includes three objects: a red apple, a red strawberry and an orange. However, the convex hull based method does not select a dominant color 'orange' as a palette color, but instead select 'yellow', which does not exist in the image. This makes further recoloring less intuitive and less predictable, i.e., adjusting palette color 'red' will largely influence the appearance of the orange.

To address this problem, we propose an improved geometric approach for palette-based image decomposition. We follow Tan et al. [TLG17, TEG18] to use a polyhedron to represent the color palette. The motivation of our method is to increase the degree of representativeness for the palette, and still maintains a high level of reconstruction accuracy. As illustrated in Figure 2 (b), our method will move the red vertex much closer to existing pixel colors, at the cost of slightly increasing reconstruction error (i.e., a few colors now become outside). We formulate the palette extraction process as an optimization problem considering both reconstruction accuracy and degree of representativeness as constraints. First, we enforce the polyhedron to enclose as many pixels as possible, i.e., maintaining reconstruction accuracy as high as possible. Secondly, we would also like the vertices of the polyhedron to be representative with respect to the image, i.e., constraining each individual vertex close to existing colors in the image as much as possible. The optimization could be efficiently solved in a few seconds through an iterative solver. As shown in Figure 1, our method successfully selected 'orange' as a palette color, and could change the appearance of the red apple and strawberry in the input image without noticeable artifacts by intuitively adjusting one palette color.



(a) The convex hull based method (b) Our method

Figure 2: *Illustration of our palette extraction method (b), compared with the RGB-space convex hull based method [TLG17] (a). While the convex hull based method produces palette colors far away from existing colors (i.e., a red vertex), our generated palette colors are closed to existing colors and hence have a higher degree of representativeness.*

For layer decomposition, we use mean value coordinates (MVCs) [JSW05, FKR05] to compute layer opacities, based on the fact that any colors inside the polyhedron could be expressed as a linear interpolation of polyhedron vertices with associated MVCs. MVCs have analytic formulas and could be computed efficiently. The mathematical properties of MVCs also ensure that they are continuous anywhere and are smooth inside the polyhedron. MVC based decomposition has the advantages of high efficiency, ease of implementation, small memory consumption, and nice parallelizability, and it is demonstrated to produce smooth layers without any precompuatation.

We have shown the effectiveness of our method for image recoloring on a variety of examples. It provides instant feedbacks, and no precomputations are needed. In comparison with existing palette-based image editing works, we make the following contributions:

- We introduced an improved geometric approach for color palette extraction, considering both reconstruction accuracy and degree of representativeness. An iterative optimization scheme is also proposed to solve for the palette within seconds.
- We also provided a simple MVC based layer decomposition scheme, which is rather efficient and is able to produce spatially smoother layers.
- Based on the above technical components, we give a palette-based recoloring system which is generally more intuitive and efficient and produces less artifacts, compared to state-of-the-art approaches.

2. Related Works

Color Palette Extraction. A color palette (or color theme), is a small set of colors to depict the color distributions of an image. Lin

et al. [LH13] trained a regression model to extract color palettes from images. Some other works [CFL*15, NPCB17, ZXST17] use k-means clustering of pixel colors to select a palette of representative colors, which is able to capture dominant colors in an image. Tan et al. [TLG17] proposed a geometric approach for palette extraction. They first compute an RGB-space convex hull which encloses all pixel colors, then iteratively simplify the convex hull to have a small number of vertices. The vertices of the simplified convex hull correspond to the generated palette colors. Since the hull vertices must reside inside the unit RGB cube, some pixel colors will probably lie outside the simplified convex hull which introduces reconstruction errors. Tan et al. [TEG18] later proposed a scheme to automatically determined a palette size, given a predefined reconstruction error threshold. In all the above approaches, the palette colors are constant colors. Differently, Aksoy et al. [AASP17] proposed to extract a palette of non-constant color distributions represented by Gaussians. Non-constant color palettes could better reconstruct color distributions in images, but they are more complex to represent and manipulate. We use constant color palettes due to its simplicity and ease of use.

Extracted color palettes could be manipulated to recolor images [WYW*10, CFL*15], patterns [LRFH13], or image groups [NPCB17]. Researchers have also proposed techniques to aesthetically rate color palettes [OAH11], and to order color palettes for meaningful interpolation between them [PFC17].

Palette-based Decomposition. After a color palette is extracted, the next step is to decompose the image into multiple layers where each layer is typically a spatially varying opacity map modulated by a corresponding palette color, such that the input image could be reproduced by blending all the layers. One category of methods [TLG17, ZXST17, LFDH17, AASP17] formulates layer decomposition as optimization problems, which enforce the composition constraint (i.e., blending all layers results in the image). To make layer opacities smooth and sparse, additional constraints concerning spatially smoothness and sparsity may also be included in the optimization. Optimization-based methods could produce visually pleasing results, however, the time cost is usually high, taking minutes to hours to process a single image.

Another category of methods [TLG17, TEG18] computes layer opacities through interpolation instead of solving a costly linear or non-linear system. In the work of [TLG17], besides an optimization based approach to compute layer opacities, they also proposed an interpolation based one. Specifically, after tessellating the RGB-space convex hull into non-overlapping tetrahedrons, any pixel color could then be represented as a linear combination of the 4 vertices of its enclosing tetrahedron using barycentric coordinates. This is named as an *as-sparse-as-possible* (ASAP) approach, where each pixel has at most 4 non-zero layer opacities. The ASAP approach is rather fast, however, since the generated layer opacities have only C^0 continuity, it suffers from speckling artifacts. The ASAP approach is later extended to 5D RGBXY space [TEG18], i.e. 3D RGB colors + 2D XY positions. They first build a 5D RGBXY-space convex hull and further compute layer opacities using barycentric coordinates with respect to tesselated simplices in the 5D space. Compared to ASAP, the produced layer opacities are much spatially smoother, and computing of the opacities is still very fast. However, building the 5D convex hull is slow, which usually takes minutes for a single image. Different from the above approaches, our method utilizes mean-value coordinates (MVCs) [FKR05, FKR05] to compute layer opacities, due to its simplicity, efficiency, and smoothness nature.

The problem of layer decomposition has also been investigated from several other aspects. Besides linear blending modes (i.e., alpha blending and additive mixing), some works have been proposed to support non-linear blending between layers [KG18, SKSF18]. Besides dealing with natural and bitmap images, some works are designed to take physical paintings as input [TDLG17, AMSL17, TDSG15], or decompose images into vectorized layers [RLMB*14, FLB17].

Matting and Intrinsic Decomposition. It is worth to note that other image applications also utilize layer decomposition, such as image matting [LLW07, LRAL08, SJTS04, CLT13] and intrinsic image decomposition [GJAF09, STL08, BPD09, CRA11]. However, their use of layers are different from palette-based image decomposition. The purpose of image matting is to separate foreground objects from backgrounds. Strokes or trimaps are given as input to solve for alpha mattes and layer colors for foreground objects. The values of alpha mattes tend to be exactly zero or one, except in boundary areas. Each layer tends to capture an entire foreground object (or the background) and hence allows to have spatially different colors. Differently, in palette-based editing, each layer has a constant color and only allows spatial changes in opacity. Intrinsic image decomposition aims at decomposing an image into a shading layer (caused by illumination) and a reflectance layer, where the image could be reproduced by multiplication of the two layers. In contrast, palette based editing uses linear blending modes in general. Besides, as another major difference, both image matting and intrinsic image decomposition do not involve the computation of a color palette.

Image Recoloring. Besides palette-based decomposition, color transfer and edit propagation are two other major techniques to perform image recoloring. Earlier works of color transfer [RAGS01, WAM02] automatically recolors a given image through global color mapping by matching color statistics with a reference image. More sophisticated methods are further proposed to recolor an image by transferring from Internet images [CZG*11] or image databases [HZMH14]. Color transfer is fully automatic, however, it provides little controls to reflect user preferences. In edit propagation [LLW04, LFUS06, AP08, PL07, XLJ*09, LJH10], users are required to put sparse strokes indicating different color edit operators (i.e., hue change) on an image, and the algorithms will propagate the sparse edits to the whole image by enforcing similarity constraints. Edit propagation provides detailed user controls, however, the need to draw strokes and specify edit operators for strokes is somehow difficult for novice users to learn and use. In contrast, palette-based editing only requires users to change palette colors, providing a good balance between the range of control and ease of use.

3. Background and Overview

Given an image **I**, the goal of palette based decomposition is to extract a palette **V** of constant colors and to decompose the image

into a set of additive layers of spatially varying opacities (i.e., each layer corresponds to a particular palette color), which satisfies that each pixel color is equal to the sum of palette colors weighted by the corresponding layer opacities. Mathematically, for each pixel, we have:

$$\sum_{\mathbf{v}\in\mathbf{V}} w_i^{\mathbf{v}}\mathbf{v} = \mathbf{i}, \quad \forall \mathbf{i} \in \mathbf{I}, \tag{1}$$

where $\mathbf{i}$ denotes the color of the pixel in the image, $\mathbf{v} \in \mathbf{V}$ denotes a palette color, $w_i^{\mathbf{v}}$ denotes the opacity of the layer corresponding to palette color $\mathbf{v}$. Besides, the opacities over all layers are constraint to sum to one: $\sum_{\mathbf{v}} w_i^{\mathbf{v}} = 1$. The above problem (Equation 1) is severe under-constrained. Generally, there exist infinite number of solutions satisfying Equation 1 if the size of palette $|\mathbf{V}| \geq 4$.

Following previous works, we also employ a two-step scheme. In the first step, we extract a color palette from the image (i.e., determine $\mathbf{v}$ in Equation 1). In the second step, we decomposite the image into layers using the extracted palette (i.e., determine $w_i^{\mathbf{v}}$ in Equation 1).

Our palette extraction approach is built on top of the existing convex hull based approaches [TLG17, TEG18]. They compute a simplified convex hull of all pixel colors in the RGB space, and then use the vertices of the convex hull as palette colors. The pixel colors inside the convex hull could be accurately reconstructed from the vertices while outside pixel colors may introduce reconstruction errors. In most cases, since only a few pixel colors are outside, a high reconstruction accuracy could still be maintained. However, it is likely to generate "hidden" palette colors, which do not exist and are distant from existing image colors in the RGB-space, making it less intuitive and less predictable when editing palette colors to perform recoloring. We follow the method proposed by Tan et al. [TLG17], using a geometric structure, i.e., a polyhedron, to represent a color palette in the 3D RGB color space. We formulate the palette extraction process as an optimization problem. In our optimization, we not only enforce a high reconstruction accuracy as was done in Tan et al. [TLG17], but also constrain the vertices of the polyhedron, which are palette colors, to be as *representative* as possible, i.e. enabling them to have a tendency moving towards the existing pixel colors. We use an iterative approach to solve this optimization problem. The polyhedron is initialized using the convex hull produced by Tan et al. [TLG17]. In each iteration, we alternatively select a vertex to optimize for its best position. The optimization converges quickly after a few iterations. Our extract palette has a higher degree of representativeness compared with [TLG17] and still maintains a high reconstruction accuracy. Details are given in Section 4.

For layer composition, we find that setting layer opacities as mean value coordinates (MVCs) [FKR05, FKR05] with respect to the polyhedron vertices provides a simple, efficient, and effective solution. MVCs have many advantages: they have closed-form formulas and low computation cost, and they are very smooth by definition. Compared with other alternatives, MVC based decomposition is more efficient without precomputations and produces spatially smoother layers. Details are given in Section 5.

Furthermore, we show how they could be used in image recoloring. We demonstrated the superiority of our approach over state-of-the-art methods on a variety of examples. Details are given in Section 6.

4. Palette Extraction

In this section, we explain how to extract a color palette from an image. Following [TLG17], we also employ a geometric approach in the RGB space. Tan et al. [TLG17] uses a convex hull of all pixel colors, whose vertices act as the palette. It ensures a low reconstruction error since only a few pixel colors lie outside the convex hull. However, Tan's method probably results in vertices which have low degree of representativeness. As illustrated in Figure 2 (a), the red vertex on the right side is far away from existing colors. In contrast, as illustrated in Figure 2 (b), our method will move the red vertex much closer to existing pixel colors, at the cost of slight increasing reconstruction error (i.e., a few colors now become outside). A higher degree of representativeness makes palette editing more intuitive and predictable in recoloring applications.

4.1. Formulation

We formulate our palette extraction process as an optimization problem. Given an image $\mathbf{I}$, we project all pixels to the 3D RGB color space, where each pixel (color) corresponds to a 3D point in the RGB space. Besides, We use a polyhedron in the RGB space to represent a color palette. The polyhedron is denoted as $\mathbf{P} = (\mathbf{V}, \mathbf{F})$, where $\mathbf{V}$ is the set of vertices and $\mathbf{F}$ is the set of faces. In the following, since some terms are actually the same thing, i.e., a *point* and a *pixel color*, a *polyhedron* and a *color palette*, a *vertex* and a *palette color*, we use them interchangeably for simplicity of description.

Our goal is to generate a polyhedron $\mathbf{P}$ that minimizes the following energy function $F(\mathbf{P})$:

$$\arg\min_{\mathbf{P}} F(\mathbf{P}), \quad F(\mathbf{P}) = \lambda \cdot R(\mathbf{P}, \mathbf{I}) + S(\mathbf{V}, \mathbf{I}). \tag{2}$$

Equation 2 includes a reconstruction loss $R(\cdot)$ and a representative loss $S(\cdot)$. λ is a parameter to control the relative contributions between the two losses.

The reconstruction loss $R(\cdot)$ accounts for the accuracy of reconstruction, and it is defined as:

$$R(\mathbf{P}, \mathbf{I}) = \frac{1}{|\mathbf{I}|} \sum_{\mathbf{i}\in\mathbf{I}} \|\mathbf{i} - \mathbf{i}_P\|, \tag{3}$$

where $\mathbf{i}$ denotes a point (pixel color), $\mathbf{i}_P$ denotes the projected point from $\mathbf{i}$ to the polyhedron $\mathbf{P}$, which is defined as the closest point on (or inside) the polyhedron from point $\mathbf{i}$. Note that the projected point $\mathbf{i}_P = \mathbf{i}$ when $\mathbf{i}$ is inside the polyhedron. Equation 3 sums over the distances from all points in $\mathbf{I}$ to their projected points. The reconstruction error only comes from those points lying outside the polyhedron, which could not be accurately reconstructed. The inside points do not introduce any errors.

The representative loss $S(\cdot)$ accounts for the degree of representativeness of vertices. It is defined as:

$$S(\mathbf{V}, \mathbf{I}) = \frac{1}{|\mathbf{V}|} \sum_{\mathbf{v}\in\mathbf{V}} \|\mathbf{v} - \mathbf{v}_c\|, \tag{4}$$

where $\mathbf{v}$ denotes a vertex of the polyhedron, $\mathbf{v}_c$ denotes the center of the neighboring points of $\mathbf{v}$, which is computed by:

$$\mathbf{v}_c = \frac{1}{M} \sum_{\mathbf{i} \in N(\mathbf{v})} \mathbf{i}, \tag{5}$$

where $N(\mathbf{v})$ is the set of M *nearest neighbor points* from $\mathbf{v}$, and $M = |N(\mathbf{v})|$ is a parameter to control the size of neighborhood set. This term enforces each vertex close to a reasonable number of existing points, making it have a higher degree of representativeness.

Note that when computing the neighborhood set of each vertex, we restrict that a point belongs to the neighborhood set of at most one vertex. When a point is found to be close to multiple vertices, we assign it to only one neighborhood set of the vertex which is closest to the point. This constraint is added to avoid two vertices move towards too close.

4.2. Iterative Optimization

It is non-trivial to solve the minimization problem defined in Equation 2. First, both the reconstruction loss and the representative loss are highly non-linear. Secondly, since Equation 2 requires enumerating over all points in the image, it is costly to evaluate.

Since the colors in natural images are usually smooth and coherent, to reduce the cost of evaluating the minimization target, instead of enumerating all points to compute an accurate energy function, we could enumerate over a small set of randomly sampled points for an approximated one. In this way, the energy function $F(\mathbf{P})$ in Equation 2 could be approximated by:

$$F(\mathbf{P}) \approx \lambda \cdot R(\mathbf{P}, \mathbf{I}_{sample}) + S(\mathbf{V}, \mathbf{I}_{sample}), \tag{6}$$

where $\mathbf{I}_{sample}$ denotes the set of randomly sampled points. We will evaluate the choice of sampling numbers $K = |\mathbf{I}_{sample}|$ in Section 6.1. In our implementation, we typically use $K = 40k$ samples.

We then employ an iterative approach to efficiently solve for Equation 6, as below:

1. **Initialization.** We use the convex hull generated by [TLG17] as the initialization of our polyhedron $\mathbf{P}$. We automatically determine the number of vertices (i.e., size of color palette) using the scheme given by [TEG18]. Nevertheless, the users can also manually specify the number of vertices, e.g., increase it for better reconstruction accuracy. Note that during the optimization process, the topology (i.e., the connectivity relation between vertices) and the number of vertices are not changed. We only optimize for vertex positions.
2. **Refinement of a vertex.** We select a single vertex $\mathbf{v}$ for refinement, and keep all other vertices fixed. Denote its initial position before refinement as $\mathbf{v}_0$.
   a. Compute the neighbor center point of $\mathbf{v}_0$ according to Equation 5, denoted as $\mathbf{v}_{c,0}$;
   b. To reduce search space, we restrict the vertex $\mathbf{v}$ to be on the line where $\mathbf{v}_0$ and $\mathbf{v}_{c,0}$ are located:
   $$\mathbf{v} = (1-k)\mathbf{v}_0 + k\mathbf{v}_{c,0}, \tag{7}$$
   where k is a scalar which we restricted in $[-0.5, 1]$.
   c. To avoid falling into a local minimum, we first sample a certain number of k with equal intervals in the range $[-0.5, 1]$ (in our implementation, we sample 10 values of k), and only keep the k with the minimal energy function (Equation 6). After that, we employ a conventional local optimization algorithm [JDP98] to obtain the optimal k through local refinement of the energy function.
3. **Cycle through all vertices.** We select another vertex that cycles through all vertices, and then repeat Step 2 to refine the position of the selected vertex. Typically, only 2 or 3 cycles are enough to converge.

Our optimization process could be viewed as trading the reconstruction loss (Equation 3) with the representative loss (Equation 4). Initially, when the polyhedron is the convex hull, the reconstruction loss is small but the representative loss is probably large. After optimization, our generated polyhedron maintains a low representative loss (i.e., a high degree of representativeness) as well as a relative low reconstruction loss. Note that we do not guarantee the generated polyhedron to be a convex shape. We believe that a convex shape is not necessary, as long as the majority of the points are tightly enclosed in polyhedron.

5. MVC based Decomposition

After obtaining the color palette (i.e., the polyhedron), the next step is to decompose the image into additive layers. Let's briefly review state-of-the-art approaches to perform layer decomposition, which could be classified into optimization-based ones and interpolation-based ones. Optimization-based approaches [TLG17, AASP17] could produce visually pleasing and smooth layers, however, they are very slow, usually take minutes to even hours for a single image. Interpolation based approaches include the ASAP (as-sparse-as-possible) method proposed in [TLG17] and the RGBXY method [TEG18]. The ASAP method is fast and produces sparse layer opacities (i.e., each pixel has at most 4 non-zero layer opacities.), however, it has only C^0 continuity and easily produces speckling artifacts during recoloring. The RGBXY method [TEG18] extends the idea of ASAP interpolation to the 5D RGBXY space. Its interpolation is also fast, and its results are much more smooth, however, it requires a preprocessing step to construct a 5D convex hull for each image, which take minutes and gigabytes memories to process a single image with megapixels. Overall, we find that existing approaches are either not efficient enough, require additional preprocessing, or have severe problem of spatial smoothness. The limitations of existing approaches motivated us to find a new approach for layer decomposition.

Mean value coordinates (MVCs) [JSW05, FKR05] are widely used for transfinite interpolation problems. They are designed to construct harmonic-like interpolant. They have several nice mathematical and geometric properties. First, they have analytic formulas and computing them are rather simple and efficient. Secondly, they are continuous everywhere and are smooth inside the polyhedron [JSW05].

Hence, we simply set the layer opacities of each inside pixel as its MVCs with respect to the polyhedron vertices. For pixels outside the polygon, we project the pixel color to its closest point on

the polyhedron and use the MVCs of the projected point as its layer opacities instead. We utilize the pseudo-code provided by [JSW05] to implement MVCs. The use of MVCs provides advantages in aspects of high efficiency, ease of implementation, small memory consumption, and nice parallelizability. The computational time and memory cost are both negligible. Thanks to its smoothness nature, we also find MVC based decomposition produces fewer artifacts in recoloring applications, compared to state-of-the-art works.

As pointed out by [TLG17], MVCs do not guarantee the sparsity of layer opacities. However, it is not a big issue since our palette extraction method largely improves the sparsity. This is because the action of enforcing a palette color to be close to existing pixel colors is equivalent to making those pixel colors have a higher opacity value close to 1 with respect to the palette color. Overall, our generated layer opacities are more sparse compared with those obtained by the RGBXY method [TEG18].

6. Evaluations and Results

We tested our method on a PC with a 3.6 GHz Intel Core i9-9900K CPU and an RTX 2080 graphics card. Our method is implemented in C++ and CUDA. We use the Python code provided by [TEG18] to obtain the convex hull as the initial value of our polyhedron. In the following, we first evaluate the parameters used in our method (Section 6.1), then present results of our method and comparisons to state-of-the-art works (Section 6.2).

6.1. Parameter Evaluation

There are four parameters in our palette extraction algorithm (Section 4), including: parameter λ (in Equation 2), the number of randomly sampled points K (in Equation 6), the size of the neighborhood M (in Equation 5), and the number of iterations in optimization.

Parameter λ controls the relative contributions between the reconstruction loss and the representative loss. Intuitively, smaller λ will produce tighter polyhedrons with higher degrees of representativeness, but introduces more reconstruction errors. A trade-off needs to be made. Figure 3 shows the reconstructed images and the generated polyhedrons using different λ for two examples. The results are consistent with our intuition. The convex hulls or polyhedrons generated with a large value of λ (i.e., $\lambda = 200$) enclose a majority of pixel colors and ensure very small reconstruction errors, but the vertices may still be far from existing colors. A small value of λ (i.e., $\lambda = 1$) provides nice representativeness, however, it leads to higher reconstruction errors (i.e., 0.82% and 0.39%, respectively) since more pixel colors will be outside the polyhedron. Overall, we find that setting $\lambda = 20$ provides a good trade-off.

Sample point number K**.** In Table 1, we evaluate different choices of the number of randomly sampled points K. Recall that we randomly select K points, instead of using all points, to approximately evaluate the energy function for optimization. Intuitively, larger K provides better accuracy but requires a higher computational cost. For each input image, we generate a ground truth polyhedron by optimizing the accurate energy function considering all points (Equation 2). For each setting of K, we generate a polyhedron using our approximated energy function (Equation 6) by randomly sampling K points. When K is larger than the resolution of the image, we use all points instead. We record the RMSE difference of the generated polyhedron with the ground truth polyhedron (i.e., by computing the difference of their vertices) and the time for optimization (i.e., the time for generating the initial convex hull is not included). We find that setting $K = 40k$ provides a good trade-off, maintaining a low difference to ground truth ($\leq$ 1.0%) and high computational efficiency (timing $\leq$ 3s).

image	res.	vt.	$K = 10k$	$K = 40k$	$K = 160k$	$K = 490k$
shop	1.6M	5	1.5%/0.4s	0.79%/1.3s	0.62%/5.1s	0.49%/14.7s
roadsign	4.6M	5	0.96%/0.5s	0.72%/1.6s	0.42%/5.5s	0.22%/16.6s
fruit	0.27M	5	1.0%/0.3s	0.70%/1.2s	0.23%/5.0s	0.0%/8.3s
building	0.24M	7	0.46%/0.7s	0.18%/2.8s	0.13%/10.5s	0.0%/16.1s
rowboat	0.23M	6	1.2%/0.45s	0.46%/1.8s	0.27%/6.4s	0.0%/9.6s

Table 1: *Evaluation of different choices of the number of randomly sampled points K. From left to right, for each example, we provide image name, image resolution, the vertex number of the generated polyhedron, and the statistics for a different setting of K, respectively. For each setting of K, we provide the RMSE difference of the generated polyhedron with the ground truth polyhedron and the optimization time. Setting $K = 40k$ provides a good trade-off.*

Neighborhood size M **and number of iterations.** First, we find that M could not be too small, i.e., ten or fewer. A too small number of neighbors may result in unstable results which can be easily affected by outliers or noises. Second, we have tested different values of M on a variety of images, from 0.05% to 2% of the number of sample points (K). In most cases, we do not notice big changes in terms of reconstruction and representative errors. Hence, we empirically set it equal to 0.1% of the number of sampled points, i.e., when we sample $K = 40k$ points, we set the size of neighborhood $M = 40$. As for the number of iterations, we find that our iterative approach converges very quickly, setting the number of iterations to 2 is enough to reach convergence in tested examples.

In the following, all results are generated using the suggested parameter values.

6.2. Comparisons and Results

Extracted Palette and Layers. In Figure 4, we provide the extracted color palettes and decomposed layers generated by our method for 9 input images. Those generated by the RGBXY method are also provided for comparison. For the majority of tested images, our generated layers are more sparse than those generated by the RGBXY method. The reason is that our extracted palette colors are more representative and are closer to existing pixel colors hence the pixels tend to have opacity values closer to one for the nearby palette color and closer to zero for other distant palette colors. Full resolution layer images could be found in the supplemental materials. In the following, we further show a variety of recoloring results using our method, and assess the quality of our color palettes and layers by comparing with state-of-the-art methods through recoloring applications, to see whether it is intuitive to edit and whether artifacts are produced.

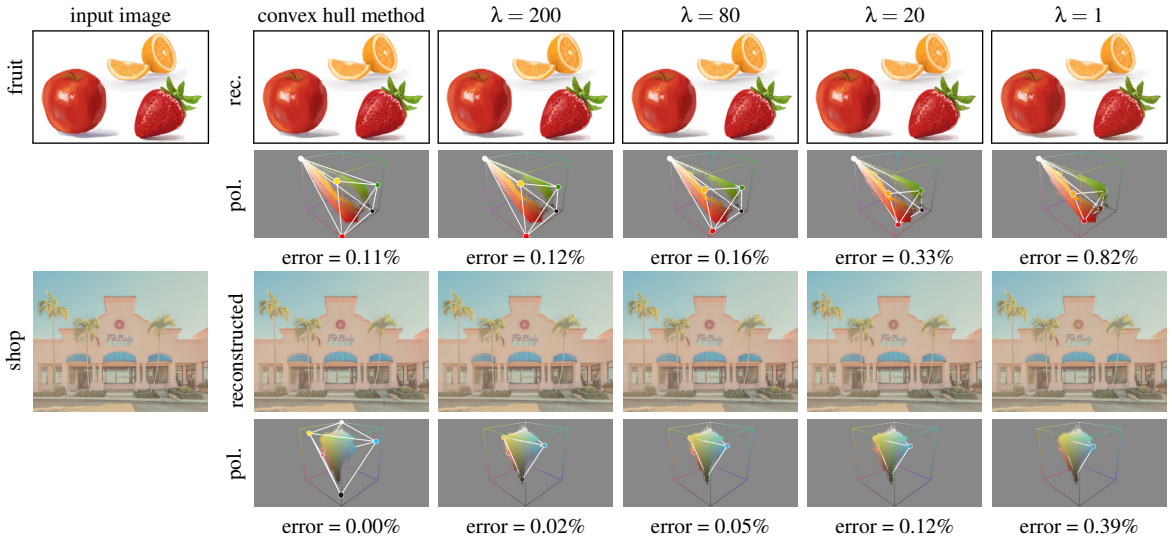


Figure 3: *Evaluation of parameter λ. From left to right, for each example, we provide the input image, the reconstructed images (top) and the polyhedrons (bottom) generated by the convex hull method [TLG17, TEG18] and by our method using* $\lambda = 200, 80, 20, 1$*, respectively. For each setting, The L1 reconstruction errors are also given. Overall, setting* $\lambda = 20$ *provides a good trade-off between the accuracy of reconstruction and the degree of representativeness.*

Recoloring. We have used our method to recolor more than 40 images. Due to limited space, we only show some of them in Figure 5. All the recolored images are given in the supplemental materials. Our method is able to produce vivid recoloring results on a variety of scenes.

Note that we have made improvements on two components. One is an improved geometric palette extraction approach (Section 4), the other is MVC-based layer decomposition (Section 5). We further separately assess these two components and then compare our overall method with state-of-the-art approaches.

In Figure 6, we compare our improved palette extraction method (Section 4) and the original convex hull-based palette extraction method [TLG17] for image recoloring. For a fair comparison, both methods use the same layer decomposition scheme, i.e., MVC-based decomposition. For each example, we use each method to generate a palette and then use this palette to recolor the input image by changing only one color of the palette. From the results, we can see that the convex hull-based method may easily lead to 'color bleeding' artifacts since its palette colors are usually far from existing colors and are less representative. In the 'postcard' image, changing the color of a yellow envelope leads to an apparent change of another red envelope (as highlighted in a circle). In the 'surf' image, changing the color of the surfboard leads to a large scale of background change. In contrast, our palette extraction method produces desirable recoloring results for all examples through intuitive interactions.

In Figure 7, we compare our MVC-based decomposition method (Section 5) with other layer decomposition methods in image recoloring, including the ASAP approach [TLG17], the RGB optimization approach [TLG17], and the RGBXY approach [TEG18]. For a fair comparison, all methods use the same palette extraction method, i.e., the convex hull-based method [TLG17]. For each example, for each method, we perform the same palette edits to recolor the image. From the results, we could find that all other approaches are likely to produce artifacts in areas where colors are slowly changing, such as sky, fogs, and soft shadow areas. The ASAP and RGB optimization approaches sometimes produce undesirable sharp boundaries, as shown in the 'bridge' image and the 'lake' images. The RGBXY approach sometimes produces artifacts like color bleeding (i.e., in the 'roadsign' image, the color of the road sign is changed when editing the color of sky), noises in areas with slow color changing (i.e., in the top right area of the 'lake' image), and soft boundaries (soft shadows in the 'toy' image). In contrast, our MVC based decomposition consistently produces smooth and visually pleasant results.

In Figure 8, we compare our overall approach with existing approaches in image recoloring, including the unmixing based approach (Aksoy et al. [AASP17]), Chang et al. [CFL*15], and the RGBXY approach [TEG18]. Each approach uses its own palette extraction and layer decomposition methods. From the results, we could see that Chang et al. [CFL*15] and the RGBXY approach are easy to produce non-smooth results or color bleeding artifacts. The results of the unmixing based approach are nice, however, it is very slow (i.e., their paper reported that it takes 4 hours and 25G memory to process a 100M image) and its user interaction is complicated and user-unfriendly (i.e., needs to export layers to Photoshop for further adjustment due to the use of non-constant palette colors). In comparison, our method generates comparable image recoloring results in a more efficient way as well as provides a more convenient interface for novice users to use.

Performance. The detailed timings are given in Table 2. For all examples, our palette optimization algorithm takes only a few

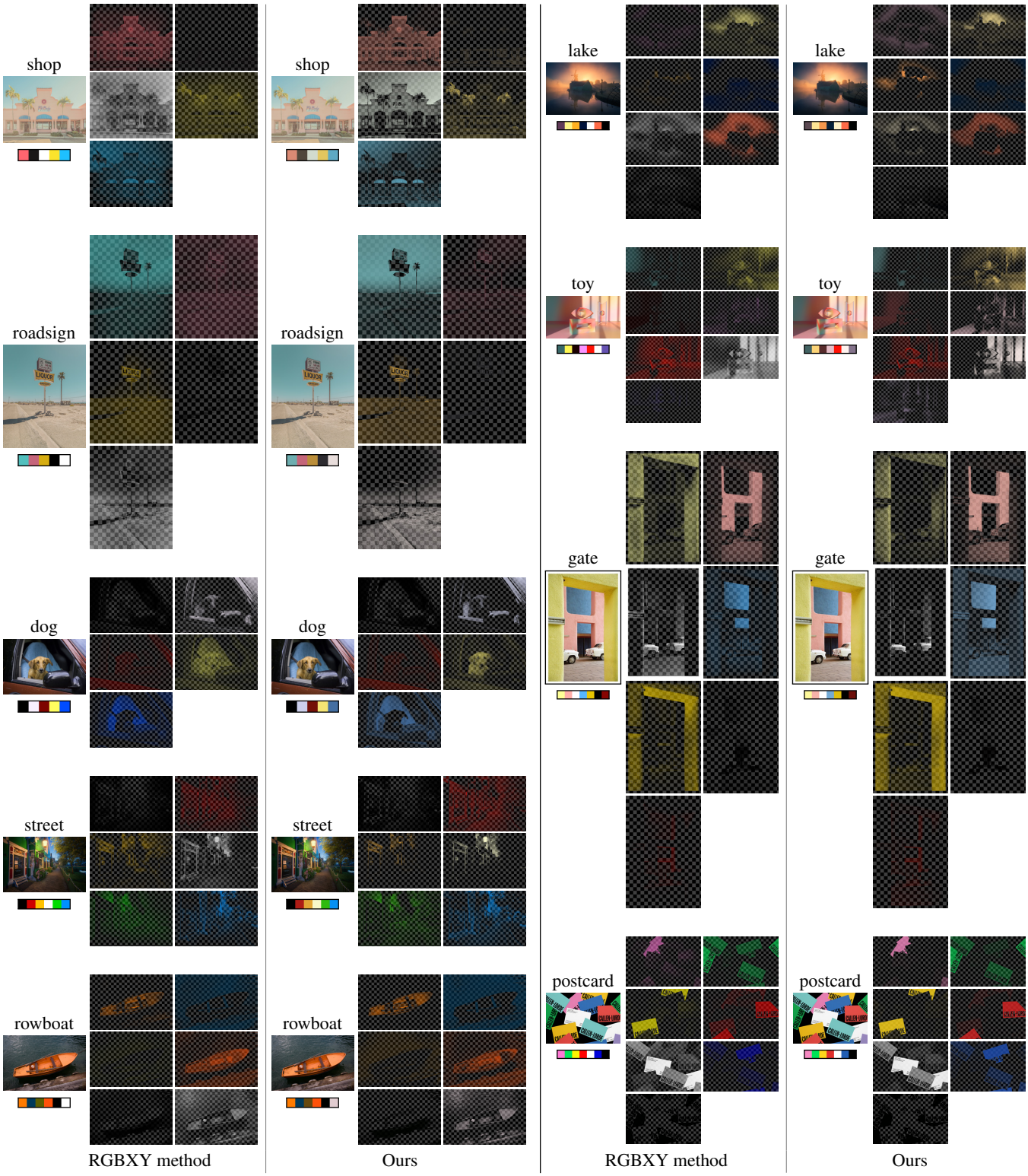


Figure 4: *Comparison of our generated color palettes and layers with those generated by the convex hull-based RGBXY method [TEG18]. On a variety of examples, our generated layers are more sparse than those generated by the RGBXY method, since our palette extraction method is able to select more representative colors to be the palette colors. Full resolution layer images could be found in the supplemental materials.*

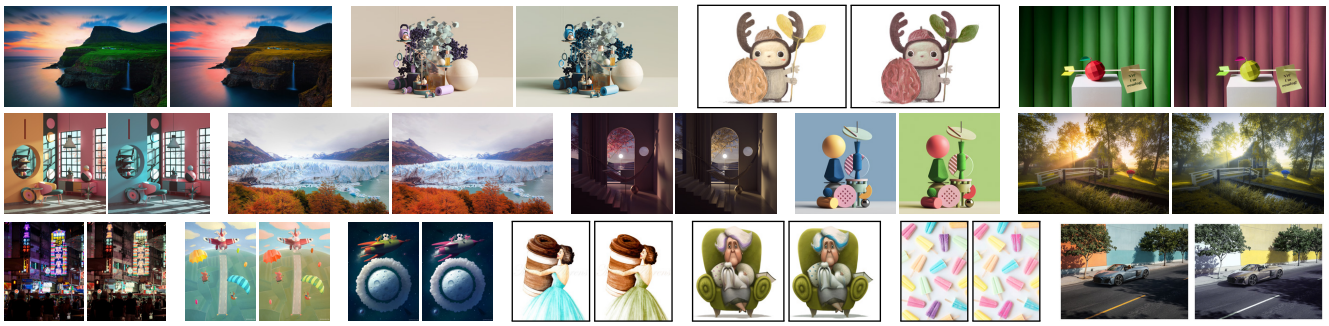


Figure 5: *Image recoloring results of our approach. For each example, we show the input image on the left and the recolored image on the right. Our method is able to produce vivid results on a variety of scenes.*

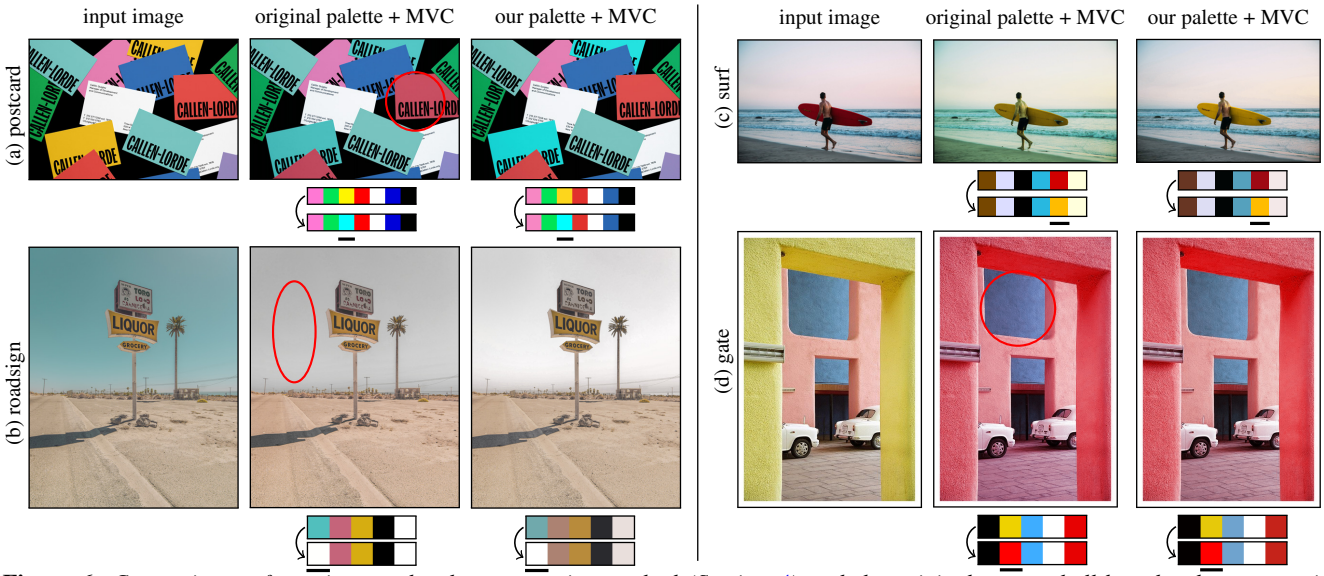


Figure 6: *Comparisons of our improved palette extraction method (Section 4) and the original convex hull-based palette extraction method [TLG17] in image recoloring. From left to right, we provide the input image, and the two recolored images (together with the unmodified and modified palettes) using the two methods, respectively. The edited colors are marked using underlines. Some artifacts are marked using red circles. From the figure shown above, we could easily find that our palette extraction method produces fewer artifacts.*

seconds, while generating the initial palette using the convex hull based algorithm takes more time. Due to the high efficiency and nice parallelizability, the time for MVC-based layer decomposition and image recoloring is almost negligible.

Failure case. If an image has distracting colors that encompass much of the RGB space, our method would be less successful. In such cases, our method may produce a palette with little difference to the original palette generated by the convex hull based approaches. Figure 9 shows such a failure example. However, such cases do not occur very often. Our method is robust for most cases including natural images with complex textures, which usually still have simple geometric structures in the RGB space.

7. Conclusion and Future works

This paper presents an improved geometric approach for palette-based image decomposition and recoloring. We use a polyhedron in the RGB space to represent a color palette. We then formulate the palette extraction process as an optimization problem considering both degree of representativeness and reconstruction accuracy. The optimization could be efficiently solved in a few seconds. We further propose to use mean value coordinates to compute layer opacities, which is efficient, easy to implement and parallelize. Overall, in comparison with state-of-the-art works, our approach is generally more intuitive and efficient with fewer artifacts for image decomposition and recoloring.

Our method could be further improved and extended in several ways. First, since one design goal of our palette extraction algorithm is high efficiency, we only optimize for vertex positions while do not change the topology. If more time budget (e.g., hours) is allowed, it is worthwhile to investigate more sophisticated algorithms such as genetic programming that simultaneously optimize for vertex positions and polyhedron topology to find a global minimum. Secondly, we currently follow the convex hull-based approaches which operate in the RGB space, however, it is also valu-

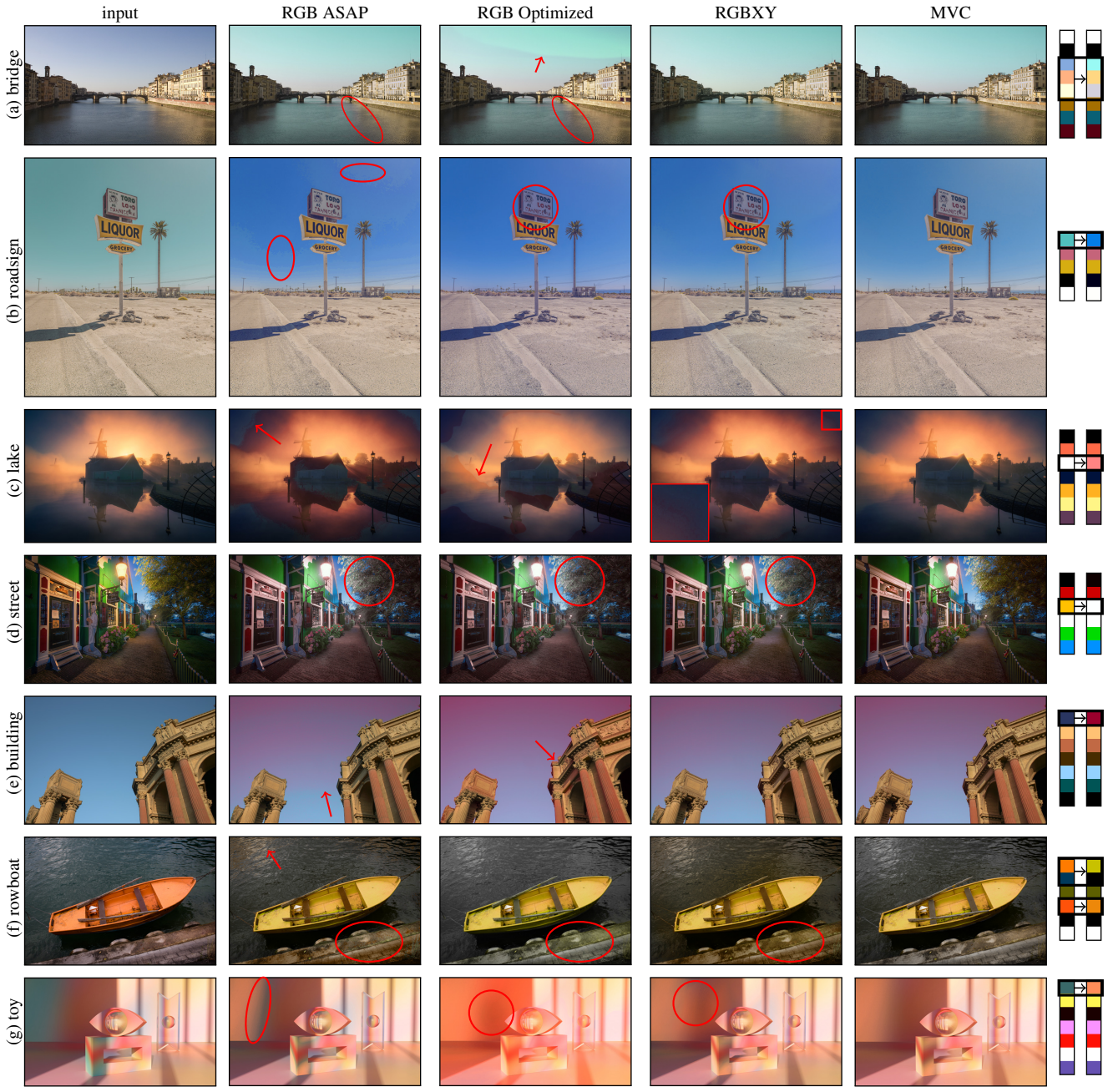


Figure 7: *Comparisons of our MVC based decomposition method (Section 5) with other layer decomposition methods for image recoloring, including the RGB ASAP approach [TLG17], the RGB optimization approach [TLG17], and the RGBXY approach [TEG18]. For fair comparison, all methods use the convex hull-based method for palette extraction. For each example, for each method, we make the same palette edits to the example and save the recolored image. From left to right, we provide the input images and the recolored images of all methods. The edited colors are marked using black rectangles, and artifacts are marked using red arrows, circles, and rectangles. From the figure shown above, we could find that our MVC-based decomposition method produces smoother results with fewer artifacts.*

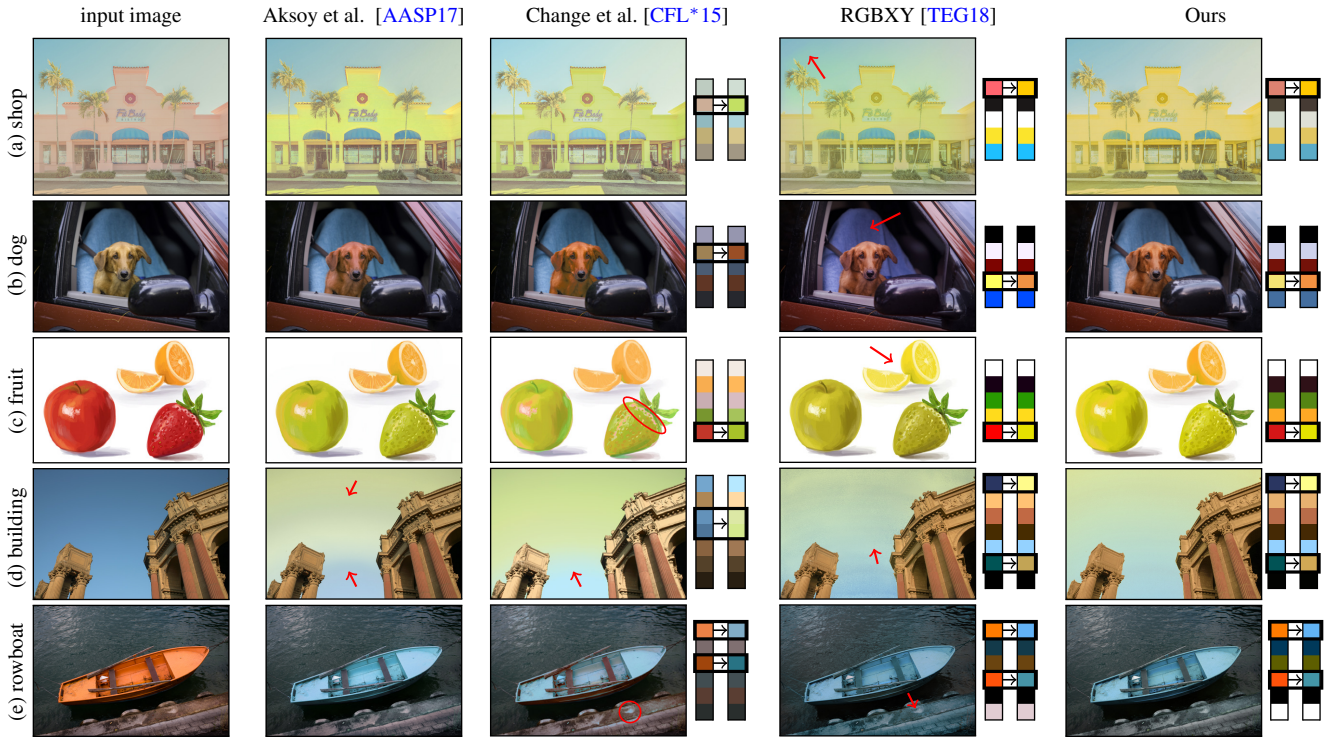


Figure 8: *Comparison of our approach and other image recoloring approaches, including Aksoy et al. [AASP17], Chang et al. [CFL*15], and the RGBXY approach [TEG18]. Artifacts are highlighted using red circles and arrows. Each approach uses its own palette extraction and layer decomposition methods.*

image	res.	vt. num	init. tm.	opt. tm.
fruit (Fig. 1)	0.27M	5	5.4s	1.2s
postcard (Fig. 6 (a))	0.96M	7	7.5s	3.0s
roadsign (Fig. 6 (b))	4.6M	5	11.6s	1.6s
surf (Fig. 6 (c))	0.70M	6	8.0s	1.8s
gate (Fig. 6 (d))	0.42M	7	10.4s	2.9s
bridge (Fig. 7 (a))	0.22M	8	8.1s	3.9s
lake (Fig. 7 (c))	5.4M	7	11.9s	3.0s
street (Fig. 7 (d))	0.27M	6	6.0s	1.5s
building (Fig. 7 (e))	0.24M	7	11.1s	2.8s
rowboat (Fig. 7 (f))	0.23M	6	9.3s	1.8s
toy (Fig. 7 (g))	0.92M	5	5.7s	1.3s
shop (Fig. 8 (a))	1.6M	5	6.5s	1.3s
dog (Fig. 8 (b))	1.0M	5	35.9s	1.3s

Table 2: *Performance table for palette extraction. For each image, from left to right, we provide the image name, image resolution, palette size, the time to obtain the initial convex hull and the time for palette optimization.*

able to discover and analyze geometric structures in different color spaces such as LAB. Besides, we are also interested in extending our method to deal with videos. In video color editing applications, our MVC-based decomposition method could support fast previewing of a single frame.

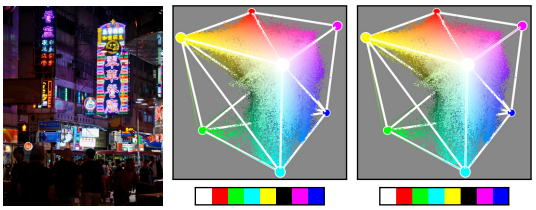


Figure 9: *A failure case. The input image has distracting colors that encompass much of the RGB space. The palette generated by our method (right) is almost the same as that generated by the convex hull based method [TLG17, TEG18] (middle).*

Acknowledgement. We thank the anonymous reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China (Project No.: 61822204, 61521002).

References

[AASP17] AKSOY Y., AYDIN T. O., SMOLIĆ A., POLLEFEYS M.: Unmixing-based soft color segmentation for image manipulation. *ACM Transactions on Graphics (TOG) 36*, 2 (2017), 19. 2, 3, 5, 7, 11

[AMSL17] AHARONI-MACK E., SHAMBIK Y., LISCHINSKI D.: Pigment-based recoloring of watercolor paintings. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (2017), ACM, p. 1. 3

[AP08] AN X., PELLACINI F.: Appprop: All-pairs appearance-space edit propagation. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 40:1–40:9. 3

[BPD09] BOUSSEAU A., PARIS S., DURAND F.: User-assisted intrinsic images. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 130. 3

[CFL*15] CHANG H., FRIED O., LIU Y., DIVERDI S., FINKELSTEIN A.: Palette-based photo recoloring. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 139. 3, 7, 11

[CLT13] CHEN Q., LI D., TANG C.-K.: Knn matting. *IEEE transactions on pattern analysis and machine intelligence 35*, 9 (2013), 2175–2188. 3

[CRA11] CARROLL R., RAMAMOORTHI R., AGRAWALA M.: Illumination decomposition for material recoloring with consistent interreflections. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 43. 3

[CZG*11] CHIA A. Y.-S., ZHUO S., GUPTA R. K., TAI Y.-W., CHO S.-Y., TAN P., LIN S.: Semantic colorization with internet images. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 156:1–156:8. 3

[FKR05] FLOATER M. S., KÓS G., REIMERS M.: Mean value coordinates in 3d. *Computer Aided Geometric Design 22*, 7 (2005), 623–631. 2, 3, 4, 5

[FLB17] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 180:1–180:11. 3

[GJAF09] GROSSE R., JOHNSON M. K., ADELSON E. H., FREEMAN W. T.: Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 2335–2342. 3

[HZMH14] HUANG H.-Z., ZHANG S.-H., MARTIN R. R., HU S.-M.: Learning natural colors for image recoloring. *Computer Graphics Forum 33*, 7 (2014), 299–308. 3

[JDP98] J. D. POWELL M.: Direct search algorithms for optimization calculations. *Acta Numerica 7* (04 1998), 287–336. doi:10.1017/S0962492900002841. 5

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics (TOG) 24*, 3 (2005), 561–566. 2, 5, 6

[KG18] KOYAMA Y., GOTO M.: Decomposing images into layers with advanced color blending. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 397–407. 3

[LFDH17] LIN S., FISHER M., DAI A., HANRAHAN P.: Layerbuilder: Layer decomposition for interactive image and video color editing, 2017. arXiv:1701.03754. 3

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R.: Interactive local adjustment of tonal values. *ACM Trans. Graph. 25*, 3 (July 2006), 646–653. 3

[LH13] LIN S., HANRAHAN P.: Modeling how people extract color themes from images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), ACM, pp. 3101–3110. 3

[LJH10] LI Y., JU T., HU S.-M.: Instant propagation of sparse edits on images and videos. *Computer Graphics Forum 29*, 7 (2010), 2049–2054. 3

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 689–694. 3

[LLW07] LEVIN A., LISCHINSKI D., WEISS Y.: A closed-form solution to natural image matting. *IEEE transactions on pattern analysis and machine intelligence 30*, 2 (2007), 228–242. 3

[LRAL08] LEVIN A., RAV-ACHA A., LISCHINSKI D.: Spectral matting. *IEEE transactions on pattern analysis and machine intelligence 30*, 10 (2008), 1699–1712. 3

[LRFH13] LIN S., RITCHIE D., FISHER M., HANRAHAN P.: Probabilistic color-by-numbers: Suggesting pattern colorizations using factor graphs. *ACM Trans. Graph. 32*, 4 (July 2013), 37:1–37:12. 3

[NPCB17] NGUYEN R., PRICE B., COHEN S., BROWN M. S.: Group-theme recoloring for multi-image color consistency. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 83–92. 3

[OAH11] O'DONOVAN P., AGARWALA A., HERTZMANN A.: Color compatibility from large datasets. *ACM Transactions on Graphics (TOG) 30*, 4 (2011), 63. 3

[PFC17] PHAN H. Q., FU H., CHAN A. B.: Color orchestra: Ordering color palettes for interpolation and prediction. *IEEE transactions on visualization and computer graphics 24*, 6 (2017), 1942–1955. 3

[PL07] PELLACINI F., LAWRENCE J.: Appwand: Editing measured materials using appearance-driven optimization. *ACM Trans. Graph. 26*, 3 (July 2007). 3

[RAGS01] REINHARD E., ADHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Computer Graphics and Applications 21*, 5 (July 2001), 34–41. 3

[RLMB*14] RICHARDT C., LOPEZ-MORENO J., BOUSSEAU A., AGRAWALA M., DRETTAKIS G.: Vectorising bitmaps into semi-transparent gradient layers. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 11–19. 3

[SJTS04] SUN J., JIA J., TANG C.-K., SHUM H.-Y.: Poisson matting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 315–321. 3

[SKSF18] SHUGRINA M., KAR A., SINGH K., FIDLER S.: Color sails: discrete-continuous palettes for deep color exploration. *arXiv preprint arXiv:1806.02918* (2018). 3

[STL08] SHEN L., TAN P., LIN S.: Intrinsic image decomposition with non-local texture cues. In *2008 IEEE Conference on Computer Vision and Pattern Recognition* (2008), IEEE, pp. 1–7. 3

[TDLG17] TAN J., DIVERDI S., LU J., GINGOLD Y.: Pigmento: Pigment-based image analysis and editing. *arXiv preprint arXiv:1707.08323* (2017). 3

[TDSG15] TAN J., DVOROŽŇÁK M., SỲKORA D., GINGOLD Y.: Decomposing time-lapse paintings into layers. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 61. 3

[TEG18] TAN J., ECHEVARRIA J., GINGOLD Y.: Efficient palette-based decomposition and recoloring of images via rgbxy-space geometry. In *SIGGRAPH Asia 2018 Technical Papers* (2018), ACM, p. 262. 1, 2, 3, 4, 5, 6, 7, 8, 10, 11

[TLG17] TAN J., LIEN J.-M., GINGOLD Y.: Decomposing images into layers via rgb-space geometry. *ACM Transactions on Graphics (TOG) 36*, 1 (2017), 7. 2, 3, 4, 5, 6, 7, 9, 10, 11

[WAM02] WELSH T., ASHIKHMIN M., MUELLER K.: Transferring color to greyscale images. *ACM Trans. Graph. 21*, 3 (July 2002), 277–280. 3

[WYW*10] WANG B., YU Y., WONG T.-T., CHEN C., XU Y.-Q.: Data-driven image color theme enhancement. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 146:1–146:10. 3

[XLJ*09] XU K., LI Y., JU T., HU S.-M., LIU T.-Q.: Efficient affinity-based edit propagation using k-d tree. *ACM Transactions on Graphics 28*, 5 (2009), 118:1–118:6. 3

[ZXST17] ZHANG Q., XIAO C., SUN H., TANG F.: Palette-based image recoloring using color decomposition optimization. *IEEE Transactions on Image Processing 26*, 4 (2017), 1952–1964. 3