Computational Visual Media
https://doi.org/10.1007/s41095-022-0314-4 Vol. 10, No. 1, February 2024, 97–118

Research Article

Hierarchical vectorization for facial images

Qian Fu[1,2,*], **Linlin Liu**[3,*], **Fei Hou**[4,5], **and Ying He**[1] (✉)

Abstract The explosive growth of social media means portrait editing and retouching are in high demand. While portraits are commonly captured and stored as raster images, editing raster images is non-trivial and requires the user to be highly skilled. Aiming at developing intuitive and easy-to-use portrait editing tools, we propose a novel vectorization method that can automatically convert raster images into a 3-tier hierarchical representation. The base layer consists of a set of sparse diffusion curves (DCs) which characterize salient geometric features and low-frequency colors, providing a means for semantic color transfer and facial expression editing. The middle level encodes specular highlights and shadows as large, editable Poisson regions (PRs) and allows the user to directly adjust illumination by tuning the strength and changing the shapes of PRs. The top level contains two types of pixel-sized PRs for high-frequency residuals and fine details such as pimples and pigmentation. We train a deep generative model that can produce high-frequency residuals automatically. Thanks to the inherent meaning in vector primitives, editing portraits becomes easy and intuitive. In particular, our method supports color transfer, facial expression editing, highlight and shadow editing, and automatic retouching. To quantitatively evaluate the results, we extend the commonly used FLIP metric (which measures color and feature differences between two images) to consider illumination. The new metric, illumination-sensitive FLIP, can effectively capture salient changes in color transfer results, and is more consistent with human perception than FLIP and other quality measures for portrait images. We evaluate our method on the FFHQR dataset and show it to be effective for common portrait editing tasks, such as retouching, light editing, color transfer, and expression editing.

Keywords face editing; vectorization; Poisson editing; color transfer; illumination editing; expression editing

1 Introduction

Vector images are ubiquitous in the digital era and essential for image production in digital media. Their key advantages include compact representation, resolution independence, being easily editable and capable of representing image content in a simple and visually pleasing manner. Although vector graphics has a long history, early methods could only represent simple colors (such as piecewise constant or linear variation). Due to this limited representation power, they were confined to a few applications, such as fonts and clip art. Recent diffusion-based vector primitives, such as diffusion curves [1] and their bi-Laplacian variant [2], significantly improve the expressiveness of vector graphics. The user sketches sparse curves and sets color constraints, and a solver diffuses colors to create an image.

Diffusion curves (DCs) can produce smooth, non-linear color functions that mimic photo-realistic images. At present, few algorithms exist for converting natural photographs into diffusion curves [1, 3, 4]. These algorithms are fully automatic and efficient.

* Qian Fu and Linlin Liu contributed equally to this work.
1 School of Computer Science and Engineering, Nanyang Technological University, 639798, Singapore. E-mail: Q. Fu, qian.fu@data61.csiro.au; Y. He, yhe@ntu.edu.sg (✉).
2 Data61, Commonwealth Scientific and Industrial Research Organisation, Sydney 2015, Australia.
3 Interdisciplinary Graduate School, Nanyang Technological University and Alibaba Group, 639798, Singapore. E-mail: linlin001@ntu.edu.sg.
4 State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China. E-mail: houfei@ios.ac.cn.
5 University of Chinese Academy of Sciences, Beijing 100049, China.
Manuscript received: 2022-05-27; accepted: 2022-09-21

However, they often produce a large number of diffusion curves, making post-editing tedious and time-consuming. Moreover, all DC vectorization methods face a common challenge: diffusion curves should not intersect, since intersecting curves often lead to unpleasant artifacts in color diffusion due to incompatible or contradictory boundary colors at intersections. As a result, it is difficult to produce hierarchical diffusion curve images.

This paper provides a method for converting portrait photos into hierarchical vector graphics. To overcome the aforementioned limitations of DCs, we adopt Poisson vector graphics (PVG) [5], which is a mixed representation of diffusion curves and a new type of vector primitive, *Poisson regions* (PRs). Unlike diffusion curve images that define colors by diffusing user-specified boundary colors from sparse curves to the entire image domain, Poisson regions, as Laplacians of colors, are relative values. Using PRs, PVG can explicitly separate colors and tones, which facilities color and tone editing. Furthermore, PVG can tolerate intersection between diffusion curves and Poisson regions [5]. The built-in advantages of PVG are of benefit to artists, who can quickly sketch vector primitives and easily organize them into various hierarchical levels without worrying about their intersection or self-intersection. Despite the many desirable features for PVG *authoring*, PVG *vectorization* is an ill-posed problem, since a pixel of the input image may be assigned to several hierarchical levels of different types.

To tackle this challenge, we develop a hybrid method that combines both deep neural networks, which have proven highly successful in 2D image processing, and traditional optimization techniques. Technically, we first use off-the-shelf tools including face parsing, face retouching, edge detection, and highlight/shadow extraction to automatically convert portrait photos into hierarchical vector primitives. Then we solve linear least squares to compute the boundary colors of diffusion curves and the Laplacians of Poisson regions. Our vectorization method converts a portrait photo into a 3-level hierarchical PVG representation. The base level is a set of sparse diffusion curves, which characterize the salient geometric features and encode the low-frequency colors of skin and hair. The middle level consists of highlight and shadow Poisson regions, both of which are large and fully editable. Finally, the top level contains 2 types of pixel-sized Poisson regions for small facial details (pimples, freckles, etc.) and high-frequency residuals. The base level DCs are always used, whereas the PRs in the upper levels are flexible and can be used arbitrarily, yielding various interesting visual effects and making photo retouching easy and intuitive. Specifically, we show that the user can edit color and illumination by modifying the boundary colors of the diffusion curves and the Laplacians of Poisson regions in the base and middle levels. To support geometric editing, we train a deep generative model that can produce high-frequency residuals automatically when the geometries of diffusion curves are changed.

Thanks to the linearity of the Laplace operator, we can reuse blending functions from raster images to vector images. In particular, we define alpha blending, linear dodge, and linear burn for the hierarchically organized PRs and show that they can be effectively used to edit highlights and shadows. We evaluate our method on the CelebAMask-HQ [6] and the Flickr-Faces-HQ-Retouched (FFHQR) [7] datasets, and demonstrate its effectiveness quantitatively and visually on various portrait editing tasks, such as color transfer, illumination, and expression editing. Figure 1 shows an example of the proposed vectorization method.

Our method takes advantage of both vectorization and deep learning in that we make use of mature data-driven techniques in portrait preprocessing and geometric editing, while applying vector primitives in coarse-to-fine attribute editing. The resulting proposed method is a unified framework for portrait editing and retouching.

In summary, we make the following contributions in the paper:

- a new method for converting portrait images into hierarchical vector representation, which supports color transfer, illumination, facial features and expression editing;
- three easy-to-use blending functions: alpha blending, linear dodge, and linear burn, for the hierarchically organized Poisson regions; and
- a novel quantitative evaluation measure for portraits, which can detect differences in highlights, shadows, color, and features between two images.

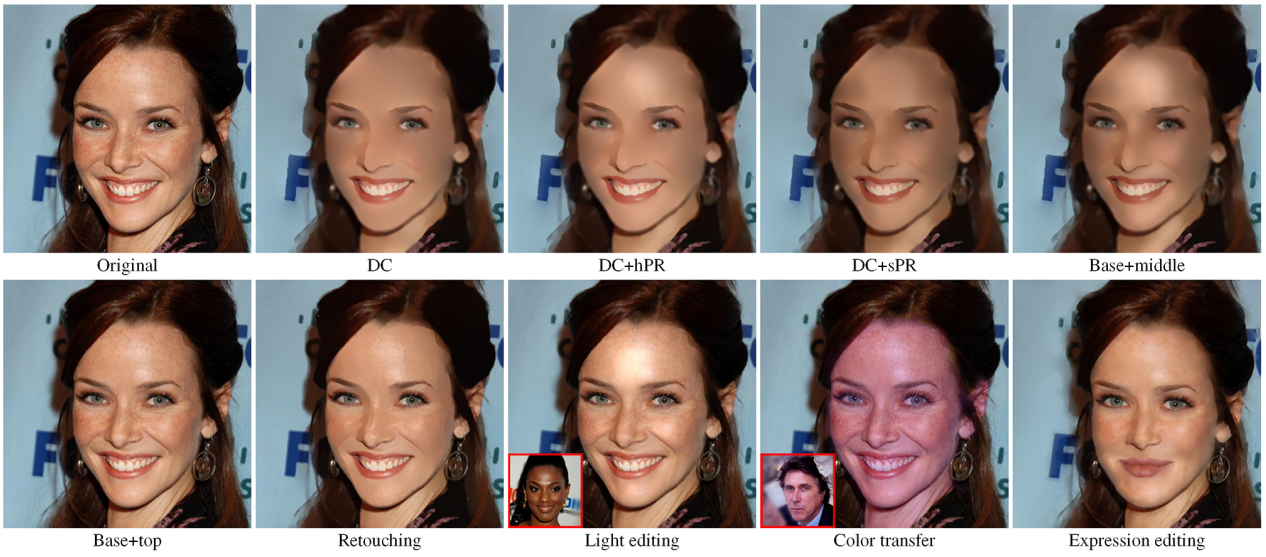


Fig. 1 Our method converts a portrait photo into a 3-level vector image, consisting of diffusion curves (DCs) in the base level, 2 types of large Poisson regions (hPR and sPR) in the middle level, and 2 types of pixel-sized Poisson regions (fPR and rPR) in the top level. Since our method explicitly separates colors from illumination and facial details in different levels, it is easy to use, and enables coarse-to-fine editing in an intuitive manner. For example, it suffices to edit colors and expressions by modifying the boundary colors and geometries of DCs in the base level, to edit illumination by modifying the PRs in the middle level, and to retouch the image by removing fPRs in the top level.

2 Related work

2.1 Intrinsic images

The intrinsic image decomposition retrieves intrinsic properties of an image and separates it into illumination, reflectance, and specular components. Some representative recent works are based on dense conditional random fields [8], near-infrared image aided energy minimization [9], and a global–local spherical harmonics lighting model [10]. There are also approaches tailored for portrait images. For example, recent deep learning approaches [11, 12] are able to decompose a face image into shape, reflectance, and illuminance, facilitating semantic facial editing.

2.2 Vector graphics

Various vector image formulations can be applied to portraits. Gradient meshes [13], which are quadrilateral meshes, represent a raster image by Ferguson patches in the planar domain and allow color manipulation in a direct fashion. However, topological constraints prevent this method from dealing with regions with holes. Lai et al. [14] proposed a fully automatic method for generating gradient meshes from multiply connected domains. Chen et al. [15] proposed a method for encoding editable geometries and fine image details in a hybrid way. Their method is efficient and supports interactive color editing, material replacement, and image magnification. Besides spline functions, subdivision surfaces [16, 17] are also a popular scheme for image vectorization. These methods represent images by triangle tessellations, which are easier to construct than gradient meshes. Furthermore, sparse curve based representations are more flexible than mesh-based representations. For example, Zhang et al. [18] vectorized 2D animation videos by identifying decorative lines separately from colored regions to output visually flicker-free, easy-to-edit animations. The more recent diffusion curves [1, 2, 19] and Poisson vector graphics [5] allow a user to specify constraints only on a set of sparse curves or regions, and then compute the color function by solving partial differential equations, e.g., Laplace's and Poisson's equations.

2.3 DC vectorization

Orzan et al. [1] adopted Canny edges [20] as diffusion curves and solved a linear least squares problem to compute their boundary colors. Xie et al. [3] pointed out that, in natural images, Canny edges usually do not overlap the Laplacian maxima, so fitting diffusion curves in the gradient domain is not accurate. To solve the problem, they proposed extracting and fitting diffusion curves in the Laplacian and bi-Laplacian domains. Their method is fully automatic, efficient, and accurate. However, for natural images with rich colors and features, it often

produces thousands of diffusion curves, including many short ones. So many primitives make post-editing of both geometry and color difficult. Zhao et al. [4] proposed a shape optimization method for extracting smooth DCs from natural images. But it also results in too many primitives, making the vectorized images unsuitable for editing. Lu et al. [21] proposed a vectorization method for RGB-D images, which extracts DCs from both color space and depth. Their method works only for indoor images, due to the strict working conditions of current RGB-D cameras.

2.4 Portrait images & video editing

Portrait images and video editing have been studied extensively, resulting in a large body of literature. The works that are most relevant to ours are lighting transfer using optimal mass transport [22] and deep neural networks [23, 24], vectorization guided color transfer [25], style transfer through deep image analogies [26], deep learning based color transfer [27], geometry editing [28], expression synthesis [29], contrast transfer [30], style transfer [31], and sketch-based face editing [32–34]. Neural model based methods have delivered promising results for many tasks. However, training deep neural networks, especially generative adversarial networks (GAN), is prone to many problems [35, 36]. A large amount of training data is often needed to achieve good results. Moreover, each neural model is usually designed to handle a single task, thereby requiring users to maintain multiple models to support a wide range of tasks. Our method, in contrast, is more flexible in that it allows the users to choose different combinations of the decomposed levels to suit their applications.

3 Hierarchical Poisson vector graphics

3.1 Overview

Poisson vector graphics [5] extends the popular diffusion curves [1]. A PVG representation consists of at least one diffusion curve and an arbitrary number of Poisson regions. Our goal is to convert a portrait image into hierarchy of Poisson vector graphics. To make the resulting PVG editable, we require a small number of vector primitives. Each diffusion curve γ is a 2-sided curve with a boundary color function g defined on both sides. The boundary colors are diffused to produce smooth colors between curves. Therefore, diffusion curves can represent low-frequency colors, for use at the base level.

A Poisson region Ω is defined by a closed curve and a non-zero Laplacian constraint f associated with the points inside the region. Since Poisson regions can intersect each other as well as diffusion curves, they are ideal for defining additional levels of detail on top of diffusion curves. Specifically, we define 4 types of PRs at the middle and top levels to encode highlights, shadows, fine details, and residuals, which are distinguished by prefixes h, s, f, and r, respectively.

To render PVG images, we solve Poisson's equation:

$$\Delta u(\boldsymbol{x}) = \begin{cases} 0, & \forall \boldsymbol{x} \in D \setminus \{\gamma \bigcup \Omega\} \\ f(\boldsymbol{x}), & \forall \boldsymbol{x} \in \Omega \end{cases} \tag{1}$$

with the Dirichlet boundary condition:

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \gamma \tag{2}$$

where D is the image domain and $u \in [0, 1]$ is the color function.

In hierarchical PVG, the user can control the geometries and boundary colors g of diffusion curves in the base level, and the geometries and Laplacian constraints of Poisson regions in the middle level. As Poisson regions in the top level are small and many may be present, directly editing them is infeasible. Instead, the user can turn each type of PR on or off, and multiply them by a global coefficient.

In raster image editing, alpha blending combines a foreground $\boldsymbol{x}_1$ with a background $\boldsymbol{x}_2$ to create the appearance of partial or full transparency. The blended image is $(1-\alpha)\boldsymbol{x}_1 + \alpha\boldsymbol{x}_2$, where $\alpha \in [0, 1]$ is the opacity. Thanks to the linearity of the Laplace operator, we can naturally define alpha blending in PVG. Given two Poisson regions with Laplace constraints f_1 and f_2 respectively, we define the blended PVG as

$$\Delta u(\boldsymbol{x}) = \begin{cases} 0, & \forall \boldsymbol{x} \in D \setminus \{\gamma \bigcup \Omega\} \\ (1-\alpha) f_1(\boldsymbol{x}) + \alpha f_2(\boldsymbol{x}), & \forall \boldsymbol{x} \in \Omega \end{cases} \tag{3}$$

Linear dodge brightens the base color to reflect the blend color by increasing the brightness in each color channel. For raster images, it can be written as $\boldsymbol{x}_1 + \boldsymbol{x}_2$ for two raster layers $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. In PVG, we define linear dodge as

$$\Delta u(\boldsymbol{x}) = \begin{cases} 0, & \forall \boldsymbol{x} \in D \setminus \{\gamma \bigcup \Omega\} \\ f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}), & \forall \boldsymbol{x} \in \Omega \end{cases} \tag{4}$$

Linear burn, which is the opposite to linear dodge, darkens the base color to reflect the blend color by decreasing the brightness in each channel. In raster image blending, it can be written as $\boldsymbol{x}_1 + \boldsymbol{x}_2 - 1$. We define linear burn in PVG as

$$\begin{cases} \Delta u(\boldsymbol{x}) = 0, & \forall \boldsymbol{x} \in D \setminus \{\gamma \bigcup \Omega\} \\ \Delta\left(u(\boldsymbol{x}) + 1\right) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}), & \forall \boldsymbol{x} \in \Omega \end{cases} \tag{5}$$

As Figs. 5 and 11 later show, applying linear dodge and linear burn to middle-level PRs allows the user to edit highlights and shadows.

3.2 Algorithmic pipeline

Figure 2 shows the algorithmic pipeline of our hierarchical vectorization method. We give overview the components here, and provide technical details in Sections 4 and 5. Table 1 summarizes typical uses cases for portraits and corresponding operations on the hierarchical PVG. Figure 3 shows an example of our hierarchical PVG.

3.2.1 Preprocessing

Let I be the input portrait image. Let I_{r} be the retouched image of I. Each image in the FFHQR dataset [7] is already associated with a retouched image created by a professional artist. For general portrait images, one can apply the auto face retouching algorithm [7] to even out the skin and remove imperfections and oily glare. We also compute a highlight image I_{h} by applying the highlight removal algorithm [37]. We then compute a highlight-compensated retouched image I_{rh} by adding highlights to I_{r}. See Fig. 4.

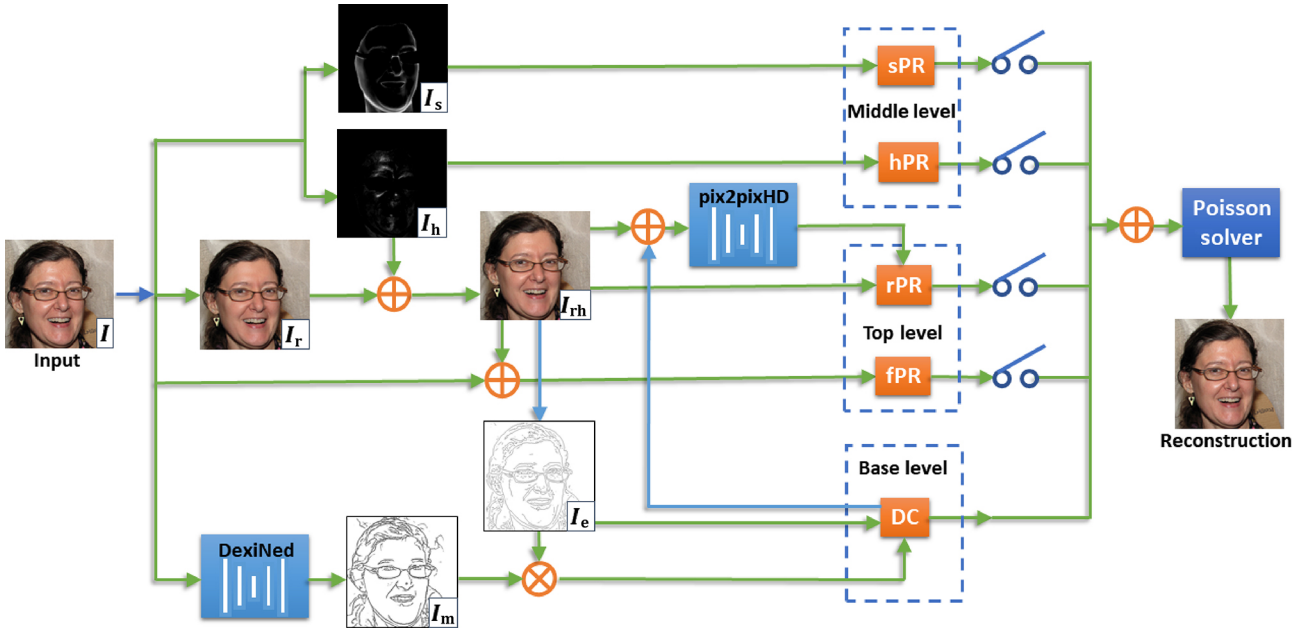


Fig. 2 Algorithmic pipeline for hierarchical vectorization.

Table 1 Hierarchical decomposition of portrait images

Level	Information	Vector primitives	Usage	Operation
Base	Salient geometric features & low-frequency colors	DCs	Color transfer & editing	Changing boundary colors
			Geometry editing	Adding/deleting curves & changing curves' geometry
Middle	Illumination	hPRs	Editing highlights	Adding/editing PRs with positive Laplacians
		sPRs	Editing shadows	Adding/editing PRs with negative Laplacians
Top	Fine details & residuals	fPRs	Restoring fine details	Multiplying the Laplacians of all fPRs by a global coefficient
		rPRs	Face retouching	Setting fPR=0 and rendering only rPRs and DCs
			Geometry editing	Generating rPRs using the deep generative model with the modified DCs as input

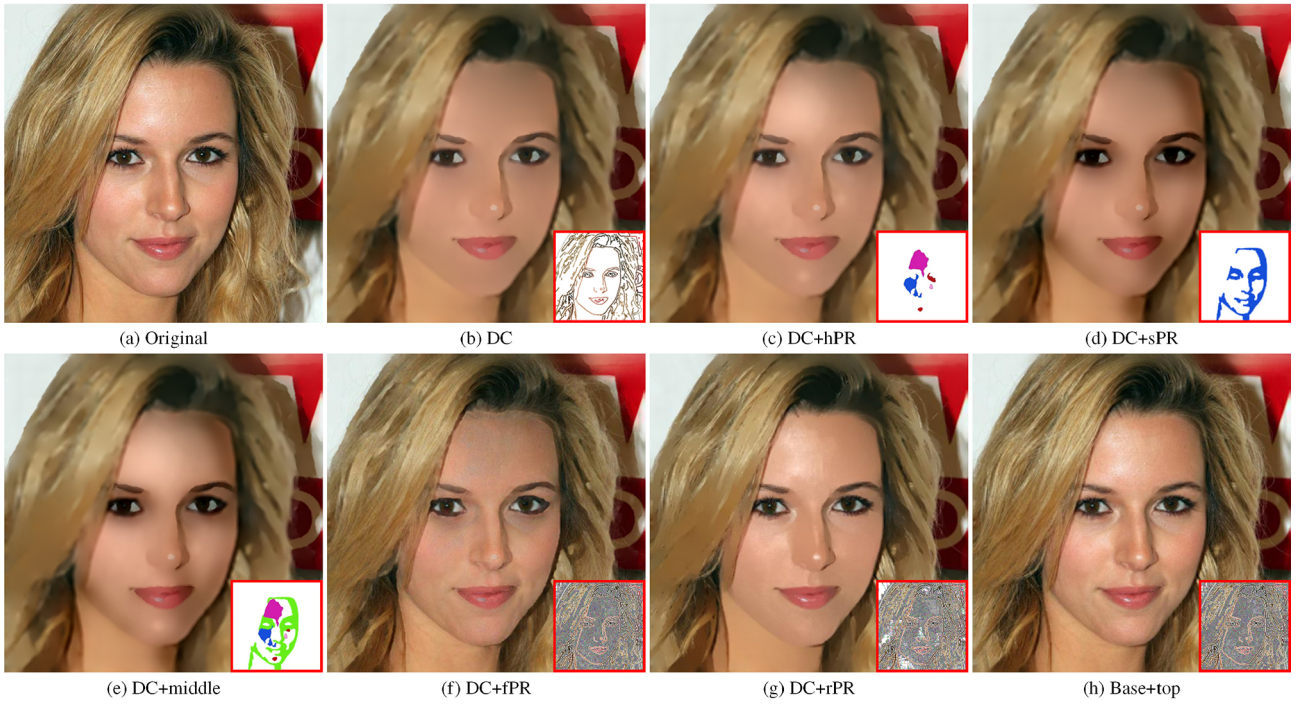


Fig. 3 An example of hierarchical PVG. The vector primitives are shown in the small insets. There are 358 DCs (5.2% px) and 7 PRs in this example. Less than 30 of the diffusion curves represent the hair and face contours as well as salient facial features, such as mouth, nose, and eyes, which are used in geometry editing. Our method can produce (e) a photo-realistic clipart effect with the base and middle levels, and (g) an automatic retouching effect with the base level and rPR.

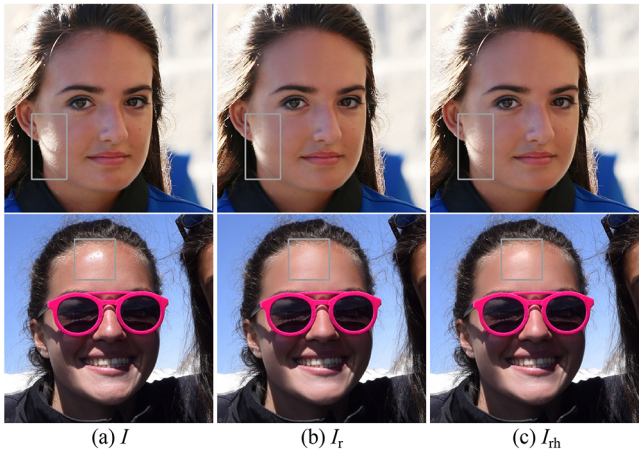


Fig. 4 Preprocessing. Given an input image I, we compute a retouched image I_{r} and a highlight-compensated retouched image I_{rh}. Differences are highlighted in greyed boxes.

3.2.2 Extracting diffusion curves (DCs)

We adopt a two-step method for computing DCs. First, we apply the probability edge algorithm [38] to extract strong edges I_{e} in the retouched image I_{rh} and use the colors on the edges to define the boundary colors g of DCs. These DCs, which encode both skin colors and illumination, may be used in geometry editing (e.g., changing expressions). In color transfer and light editing, we require skin colors to be separated from illumination. So we modify the DexiNed neural network [39] to compute salient facial feature curves as an edge mask I_{m}. Applying I_{m} to the probability edge induced DCs can effectively reduce interference on skin color by light.

3.2.3 Extracting highlight & shadow Poisson regions (hPRs & sPRs)

We apply a highlight removal algorithm [37] to extract specular highlights I_{h} from the original image I. Then we soften the boundaries of the highlights using a median filter. Applying the method of Ref. [37] to the inverse image $(1-I)$ yields shadows I_{s}. We assign each hPR a positive constant Laplacian, and each sPR a negative constant Laplacian.

3.2.4 Extracting fine detail Poisson regions (fPRs)

The fine details are small-scale features, such as pimples and pigmentation. We define fPRs as the Laplacian of the difference between the input image I and the highlight-compensated retouched image I_{hr}: $\triangle(I - I_{\mathrm{hr}})$.

3.2.5 Extracting residual Poisson regions (rPRs)

Residual PRs are pixel-sized PRs defined as the Laplacian of the highlight compensated retouched image I_{hr}. It is worth noting that rPRs are not meant for direct editing. Instead, they are provided to preserve photorealism of the results. For applications that do not change the geometry of the DCs (e.g., color transfer, light editing), we simply keep the extracted rPRs unchanged. For other applications which change DC geometries (e.g., expression editing), we use a deep generative model, which takes the modified DCs as input, to generate new rPRs. For this reason, we call them "residual" Poisson regions.

3.3 Properties

Our 3-level, hierarchical PVG representation has several properties beneficial for face image retouching and editing.

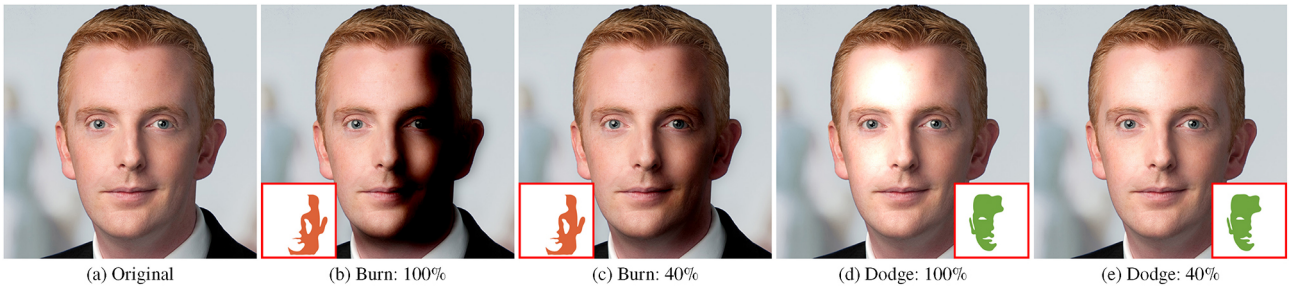
First, the vector primitives are organized in a hierarchical structure: contours and salient facial features are in the base level, and fine details (such as pimples and uneven skin color) and high-frequency residuals are in the top level. The middle level models highlights and shadows, and does not contain geometric information. The user can manipulate the base level primitives (sparse diffusion curves) to modify the face geometry. Since there are only a few PRs in the middle level, the user can also directly modify their geometries and Laplacians to edit illumination. For the PRs in the top level, the user can apply filters and/or use brushes to modify them.

Second, our portrait PVG explicitly separates colors and tones, which follows the basic drawing principle adopted by artists. There are only a small number of DCs in the base level, representing hair and face contours and salient facial features, such as eyes, month, and nose. Each DC is explicitly assigned boundary colors on both sides. To change face and hair color, it suffices for the user to modify the boundary color of a few relevant DCs. If the input image is vectorized using DCs only [1, 3, 4], editing colors is tedious and time-consuming, since the user has to work on a large number of DCs. Our method further supports automatic color transfer from a reference image to the input image, where the two images may have very different facial geometries, colors, and head poses. The 2 types of PRs in the middle level enable the user to directly edit illumination. Each PR is assigned a constant integral Laplacian in Eq. (7). The user can modify both the geometries and the integral Laplacians of PRs to edit illumination.

Third, unlike diffusion curves that are strictly intersection-free, Poisson regions are flexible in that they allow intersection and self-intersections, and can be freely placed and blended. This feature is particularly desirable in illumination editing that involves highlight and shadow PRs in the middle level (see Fig. 11 later). Also, combining the middle-level PRs and the base-level DCs yields a visually pleasing clipart effect (see Fig. 3(e)).

Fourth, the linearity of the Laplace operator allows us to introduce three linear blending modes: alpha blending, linear dodge, and linear burn, which allow the user to edit the highlight and shadow PRs in the middle level in an easy and intuitive manner. See Fig. 5 for an example.

Fifth, our method takes advantage of the editability of vector primitives. All vector primitives in the base and middle levels can be freely edited by the user. However, the high-frequency signals in the top level



(a) Original (b) Burn: 100% (c) Burn: 40% (d) Dodge: 100% (e) Dodge: 40%

Fig. 5 Applying linear blending functions to edit highlights and shadows. (b, c) We add an sPR with integral Laplacian $(-20, -20, -20)$ to create shadows on the right cheek. (d, e) We add an hPR with integral Laplacian $(20, 20, 20)$ to create highlights. By tuning the opacity, we obtain visually pleasing results in (c) and (e).

are not meant for direct editing. They are used to preserve photorealism of the results. In applications that do not involve changing the geometry of diffusion curves (e.g., color transfer and illumination editing), we simply keep the top-level PR unchanged. For other applications (such as expression editing) in which the user changes the geometries of diffusion curves, we adopt a deep generative model to automatically yield the primitives of residual PRs. As a result, the user can freely modify the base level primitives without worrying about the primitives of upper levels. See Fig. 8 later for an illustration.

4 DC extraction

4.1 Approach

Diffusion curves are a sparse set of curves with boundary colors assigned on both sides. Diffusion curve images are rendered by diffusing the boundary colors from the curves to the entire image domain. Existing DC extraction methods rely on edge and feature detectors. Traditional edge detector methods, such as Refs. [20, 38], consider only local color changes and often yield short and incomplete curves. It is known that diffusion curve images are highly sensitive to the positions of curves. For example, if a region's boundary, which is supposed to be a closed curve, turns out to be incomplete, colors inside the region will diffuse out of it, causing color leaking artifacts.

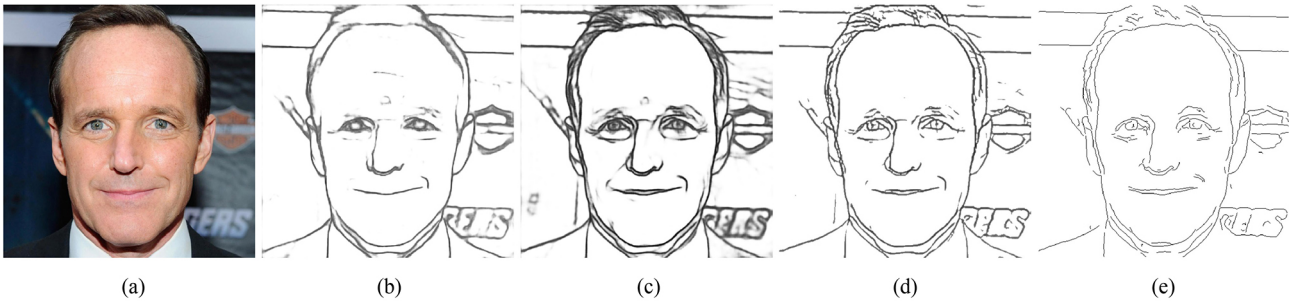
To address such issues, we adopt the following strategy. First, we apply a classic edge detection algorithm [38] to obtain strong feature lines, which are located at points with high color gradients. These lines are accurate in terms of position, but they often mix skin colors and illumination, which are supposed to be separate. Second, we compute a highlight-compensated retouched image I_{rh} by minimizing the objective function in Eq. (6):

$$I_{\mathrm{rh}} = \arg\min_{\varepsilon}(I - I_{\mathrm{r}} - \varepsilon I_{\mathrm{h}})I_{\mathrm{h}} + I_{\mathrm{r}} \tag{6}$$

where I is the input image, I_{h} the highlight image, and I_{r} the retouched image. Third, we adopt a modified DexiNed network (explained below) to extract edges in I_{rh} and smooth them to produce an edge mask I_{m}. Fourth, applying the mask to the probability edges, we obtain diffusion curves which characterize the main geometric features and are less sensitive to illumination. Finally, we assign each pixel on either side of the extracted diffusion curve a color which corresponds to a pixel color of the largest gradient difference. To edit the color of a specific facial component, we use the off-the-shelf face parsing model by Lee et al. [6] to segment facial images, and then apply the corresponding masks of the specified facial components to edit diffusion curve color. See Fig. 6 for an example edge mask and Fig. 7 for extracted DCs with and without the edge mask. In contrast with previous methods [20, 38], which rely only on local color changes, our method considers both colors and facial features since the DC mask extraction model is trained on annotated portrait images.

4.2 Portrait-tailored DexiNed

DexiNed [39] is a deep neural network based edge detector, originally trained on the BIPED dataset with 250 annotated outdoor images. We manually annotate 27 portrait images and combine them with the BIPED dataset, and re-train DexiNed. We also adopt the data augmentation script provided by BIPED at `https://github.com/xavysp/MBIPED.git` for data augmentation. In addition, we apply self-training [40, 41], where we use the model trained on



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

Fig. 6 Example DC mask: (a) input image I, (b) output of the original DexiNed model, (c) output of our retrained model, (d) DC mask I_{m} by smoothing (c), and (e) extracted features $I_{\mathrm{e}} \cap I_{\mathrm{m}}$ after applying the mask I_{m} to the probability edges.

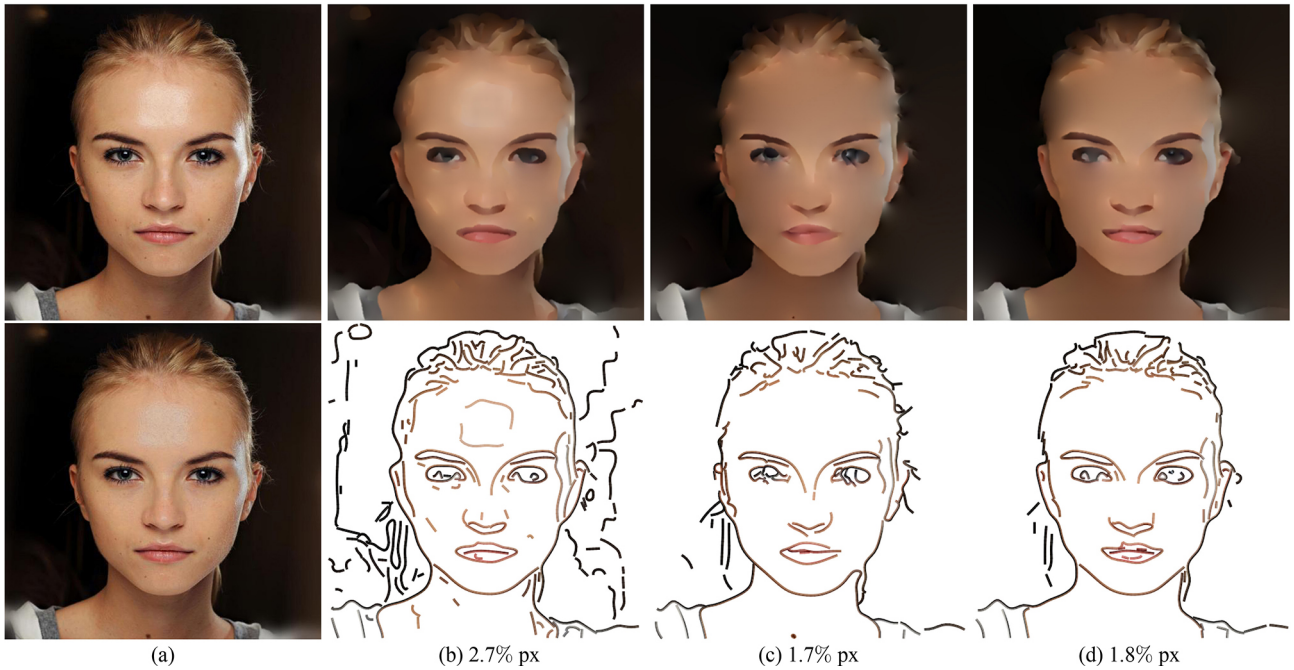


Fig. 7 DC extraction. The percentages are the ratio of the number of pixels in DCs to the number of pixels in the image. (a, above) Input image I, (a, below) highlight-free image $I - I_{\mathrm{h}}$ (bottom). (b) The DCs I_{e} generated by the probability edge method [38] often mixes skin colors and illumination. (c) The DCs extracted by the DexiNed model can reduce the mixing of skin color and illumination. However, locations are not as accurate as those of probability edges. (d) We compute an edge mask I_{m} using DexiNed and then apply it to the probability edges I_{e}. $I_{\mathrm{e}} \cap I_{\mathrm{m}}$ provides high-quality DCs, reducing interference of facial lighting, and can be used for facial color transfer and light editing.

manually annotated data to annotate 500 randomly sampled portrait images, and then combine them with the hand-annotated data to train the final model. Given any portrait image, our re-trained model is able to generate the DC mask by assigning a probability to each pixel. We use a smoothing algorithm to further process the DC mask: (i) apply a Gaussian filter to the DC mask and then set the pixels below a given threshold (0.35 in our implementation) to 0, (ii) move a 5×5 sliding window across the DC mask with stride 1, and filter out the bottom 20% values in each window, and (iii) repeat step (i) to generate the final DC mask.

5 Hierarchical PR extraction

5.1 Approach

In the hierarchical PVG representation, Poisson regions are in the middle and top levels. The middle level characterizes highlights and shadows, and the top level encodes residuals and fine details. There are two types of middle-level PRs, hPRs and sPRs, which are large and editable regions. A typical portrait PVG has only a small number of hPRs and sPRs (no more than ten), so the user can edit them directly. The top-level fPRs and rPRs are small, pixel-sized regions, which encode high-frequency signals such as pimples, pigmentation, and residuals. Note that the top-level PRs are not meant for direct editing due to their large number and small size.

5.2 Highlights & shadows

To facilitate highlight and shadow editing, we define 2 types of editable PRs in the middle level. With the highlight removal algorithm [37], we can extract highlights I_{h} and shadows I_{s} from the original image I and its inverse image $(1 - I)$. After that, we apply a median filter to the highlight image I_{h} and then extract boundaries to determine non-trivial hPRs with relatively smooth boundaries; sPRs are filtered in the same way.

Unlike DCs, PRs are 2-dimensional objects. Let Ω_i be a Poisson region. Following Ref. [5], we require that the area integral of the Laplacian $\int_{\Omega_i} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 0$ vanishes. This zero-sum requirement is a necessary condition for local shading control. Denote by $\Omega_i' \subset \Omega_i$ the region such that the boundary $\partial\Omega_i$ is shrunk a distance δ inwards along the boundary normals. In our implementation, we set $\delta = 5\%$ of the diagonal of Ω_i. We control Ω_i' by specifying the integral Laplacian

P_i for Ω_i' as

$$P_i = \int_{\Omega_i'} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{7}$$

Then we assign the Laplacian for each point of $\Omega_i \setminus \Omega_i'$ as

$$\frac{-P_i}{\int_{\Omega_i \setminus \Omega_i'} \mathrm{d}\boldsymbol{x}}$$

to satisfy the zero-sum condition, where P_i is given by the mean Laplacian value over Ω_i'. In addition, the user can edit highlights and shadows by specifying different integral Laplacians P_i for Ω_i' or directly modifying its geometry to produce various illumination effects.

5.3 Fine details & residuals

Fine details and residuals are the two types of PRs in the top level. Both are pixel-sized and usually significantly outnumber the primitives in the base and middle levels. Hence, these PRs are not meant for direct editing; they also differ in terms of usage and operations.

We define residual Poisson regions as the Laplacian of the highlight compensated retouched image I_{rh}, $f_{\mathrm{r}} = \Delta I_{\mathrm{rh}}$. Then we define fine details Poisson regions as $\Delta I - f_{\mathrm{r}}$, which contains facial details such as pimples and pigmentation.

Since residual PRs are coupled with the *geometries* of diffusion curves, rPRs need to be updated whenever the user changes the shapes of DCs. Note that rPRs are pixel-sized Poisson regions that are not meant for direct editing. Instead, we train a deep generative model to encode high-frequency residuals. Specifically, we modify the image-to-image translation model, pix2pixHD [42], which takes diffusion curve images (DCI) as input and outputs residuals. We train the model to reconstruct the corresponding rPRs from the given DCIs. As Fig. 8 shows, the user only needs to modify sparse diffuse curves to edit expression without worrying about the high-frequency residuals. Throughout the paper, we use rPR′ to denote the residual Poisson regions that are generated

Fig. 8 Generating residual PRs using deep learning. Rightmost column: clockwise from top left: original DCs, filtered DCs by applying DC masks, DC masks, and extracted edges from portrait-tailored DexiNed.

by the deep generative model, in contrast to rPR, which is computed from the highlight compensated retouched image I_{rh}. We also use rPR$''$ to denote the residual Poisson regions generated by the deep generative model that takes the original DCs *with DC masks* as input. In practice, rPR$'$ is often used in geometry editing when the original light is desired, whereas rPR$''$ is used in situations where light can be removed (Fig. 8(d) vs. Fig. 8(e)). Using fine detail Poisson regions, the reconstructed vector images are photo-realistic (Fig. 8(c) vs. Fig. 8(d)).

6 Illumination-sensitive image quality metrics

To quantitatively evaluate the results of color and highlight transfer, we propose a new metric, called illumination-sensitive IS-FLIP or IS-FLIP-ct, that measures the difference for the reconstruction images and the color-transferred images. IS-FLIP extends recent FLIP metric [43], which is a difference evaluator with a particular focus on differences between rendered images and corresponding ground truths. Given two images I and J, FLIP computes the per-pixel color and feature differences δE_{c} and $\delta E_{\text{f}} \in [0,1]$, respectively. The FLIP difference value, δE_{F}, per pixel is then given by

$$\delta E_{\text{F}} = \delta E_{\text{c}}^{(1-\delta E_{\text{f}})}$$

FLIP produces a map that approximates the difference perceived by humans when alternating between I and J [43]. Though FLIP is a powerful difference evaluator, it does not consider illumination differences. Since human vision is more sensitive to large illumination differences such as highlights and shadows rather than small details in portrait images, FLIP is not an effective measure for portrait images.

IS-FLIP improves FLIP by adding a term for illumination difference δE_{i}:

$$\delta E_{\text{i}}(I,J) = \left(|I_{\text{h}} - J_{\text{h}}| + |I_{\text{s}} - J_{\text{s}}|\right)/2$$

where I_{h} and J_{h} are extracted highlights obtained by the highlight removal algorithm [37], and I_{s} and J_{s} are extracted shadows obtained by applying the algorithm from Ref. [37] on the inverted images. We then define IS-FLIP δE_{I} as

$$\delta E_{\text{I}} = \delta E_{\text{i}}^{(1-\delta E_{\text{F}})} \tag{8}$$

We adopt δE_{I} to evaluate the accuracy of vectorization, since it can capture the differences between an image and an alternative image based on human perception.

Note that IS-FLIP involves only two images I and J. To suit IS-FLIP for color transfer, we define a variant, called IS-FLIP-ct, which considers the input image I, the reference image R, and the transferred image J. Let $M(I,R)$ be the transferred result by applying a baseline histogram matching method M (e.g., MATLAB's `imhistmatch` function) to I and R. Specifically, we can rewrite δE_{F} by measuring the feature differences between J and $M(I,R)$:

$$\delta E_{\text{F}}^{\text{ct}}(I,J,R) \triangleq \delta E_{\text{F}}(M(I,R),J)$$

Then we define IS-FLIP-ct for color transfer as

$$\delta E_{\text{I}}^{\text{ct}} = \delta E_{\text{i}}^{\left(1-\delta E_{\text{F}}^{\text{ct}}\right)} \tag{9}$$

where the superscript denotes the variable for color transfer. As Fig. 9 shows, IS-FLIP-ct is more effective than FLIP and IS-FLIP for evaluating the quality of color transfer results.

There are also other metrics to evaluate the quality of color transfer, such as structural similarity index measure (SSIM) [44], the KL divergence, and Hellinger distance [27]. Later, in Fig. 13, we show FLIP, SSIM, and the proposed IS-FLIP measures. HistoGAN [27] adopts the KL divergence and Hellinger distance to measure the similarity between the target histogram and the histogram of the color-changed images. Notice that KL divergence and Hellinger distance focus on only the difference of two color distributions, and do not consider illumination changes. Figure 15 later shows color transfer results of different methods. Judging quality by KL divergence and Hellinger distance which were used in Ref. [27], the order from high to low quality is (g), (f), (d), (b), (c), and (h). However, if measured by our proposed IS-FLIP, the order is (b), (c), (f), (g), (h), and (d). From Figures 15 and 13, in comparison with FLIP and SSIM, we observe that IS-FLIP and IS-FLIP-ct are more consistent with human perception, since they consider color as well as illumination and meanwhile avoid identifying trivial details which are similar to the human eye.

7 Experimental results

7.1 Evaluation

We evaluated our method on the CelebAMask-HQ dataset [6], which consists of 30,000 high-resolution

Fig. 9 The IS-FLIP-ct metric δE_I^{ct} is more effective than FLIP δE_F and IS-FLIP δE_I for evaluating color transfer results. The input image I and the color transferred result J have the same facial features but different colors. However, due to significant color change between J and I, both FLIP and IS-FLIP, which take color changes as part of the difference metric, yield large error values. IS-FLIP-ct, in contrast, can reduce such effects with a reference color transfer result $M(I,R)$, so is more effective than FLIP and IS-FLIP for evaluating color transfer results. We visualize the error metrics to compare their effectiveness: the less obvious the features in the error maps, the more effective the error metric is for measuring color transfer quality. Human perception also confirms the efficacy of IS-FLIP-ct.

photos of celebrities. Each image has a segmentation mask for facial attributes, such as hair, face, brows, eyes, nose, lips, ear, neck, and clothing. To demonstrate the advantages of our hierarchical PVG, we applied it to color transfer, which can be mathematically modeled with optimal mass transport (OMT). We compared our method with 3 raster-image based OMT methods and 2 vectorization based OMT methods. The raster-image based methods apply OMT to the original images, the intrinsic images, and the L_1 smoothed images. The vectorization based methods are 2-level PVG [25], and intrinsic image-based PVG which vectorizes each decomposed layer using Ref. [25]. Table 2 reports the mean and variance of the IS-FLIP-ct and FLIP metrics on 500 representative images (Nos. 1500–1999) of the CelebAMask-HQ dataset. To make a fair comparison with intrinsic image based methods, we

Table 2 Evaluating vectorization accuracy and color transfer quality using FLIP, IS-FLIP, and IS-FLIP-ct measures. DL-PVG means DCs combined with a deep generative model for geometry editing

	Vectorization				Color transfer			
	Mean (%)		Variance		Mean (%)		Variance	
	IS-FLIP	FLIP	IS-FLIP	FLIP	IS-FLIP-ct	FLIP	IS-FLIP-ct	FLIP
Raster	—	—	—	—	6.02	13.00	1.57E-3	3.68E-3
L_1 smoothing	—	—	—	—	6.09	13.41	1.54E-3	3.56E-3
Intrinsic raster	2.16	6.82	1.54E-4	3.04E-4	9.69	19.42	2.41E-3	2.76E-3
Intrinsic PVG	5.72	15.96	6.56E-4	9.18E-4	9.92	19.48	2.45E-3	2.76E-3
Conventional PVG	0.16	0.81	3.59E-7	4.68E-6	6.17	14.46	1.62E-3	3.32E-3
Hierarchical PVG	**0.20**	**1.02**	**4.53E-7**	**5.68E-6**	**5.25**	**12.99**	**1.18E-3**	**2.52E-3**
DL-PVG	**2.40**	**7.55**	**1.03E-4**	**3.37E-4**	—	—	—	—

Fig. 10 Hair color transfer by modifying colors of sparse diffusion curves: (a) original image, (b, c) reference images, (d, e) hair color transfer, and (f–i) hair highlighting effects.

calculate IS-FLIP and FLIP for the facial regions, since the existing intrinsic image decomposition can only process the face regions. For vectorization, our method obtains similar accuracy to the conventional PVG vectorization method [25]. For color transfer, our method yields results with the highest quality. Visual results also confirm that our method can effectively keep the original illumination and facial details during color transfer.

To compare the efficiency of various methods, we ran them on a PC with a 2.80 GHz Intel i7 CPU. Table 3 breaks down the running time spent for reconstruction and update, as preprocessing, vectorization, recomputation, and OMT.

7.2 Semantic color transfer

Since our method encodes low-frequency colors in a set of sparse diffusion curves with semantic meanings, semantic color transfer becomes easy. For example, the user can change colors semantically in the base level such as facial and hair color transfer shown in Fig. 10, and in Fig. 15 later. Note that we can readily edit the targeted curves to generate hair highlighting effects thanks to our vector representation, which is challenging if only using deep learning based methods.

Table 3 Runtime breakdown of reconstruction and color transfer, into preprocessing, vectorization, recomputation, and OMT. Time reported is in second

Method	Prep.	Vec.	Recomp.	OMT
Raster	—	—	—	7.58
L_1 smoothing	327.82	—	—	7.55
Intrinsic raster	0.75	—	—	14.95
Intrinsic PVG	0.75	4.44	0.50	2.28
Conventional PVG	0.50	2.43	0.52	1.65
Hierarchical PVG	**1.28**	**0.97**	**0.50**	**1.39**

7.3 Light editing

Since our method separates illumination from color, the user can explicitly edit light using the PRs in the middle level. Figure 11 shows light editing results using the proposed blending functions.

7.4 Expression editing

Since there are only a few sparse diffusion curves in the base level, geometry editing becomes easy. Figure 12 shows examples of expression editing, in which the user changes the geometries of the diffusion curves. The fine details and high-frequency residuals are generated automatically by a deep generative model.

7.5 Comparison with DC vectorization

Existing diffusion curve based vectorization methods [1, 3, 4, 46] often produce a large number of vector primitives, making post-editing difficult. Our hierarchical PVG can significantly reduce the user's burden since the base and middle levels contain a small number (a few tens) of vector primitives. The user can edit the middle-level PRs by using

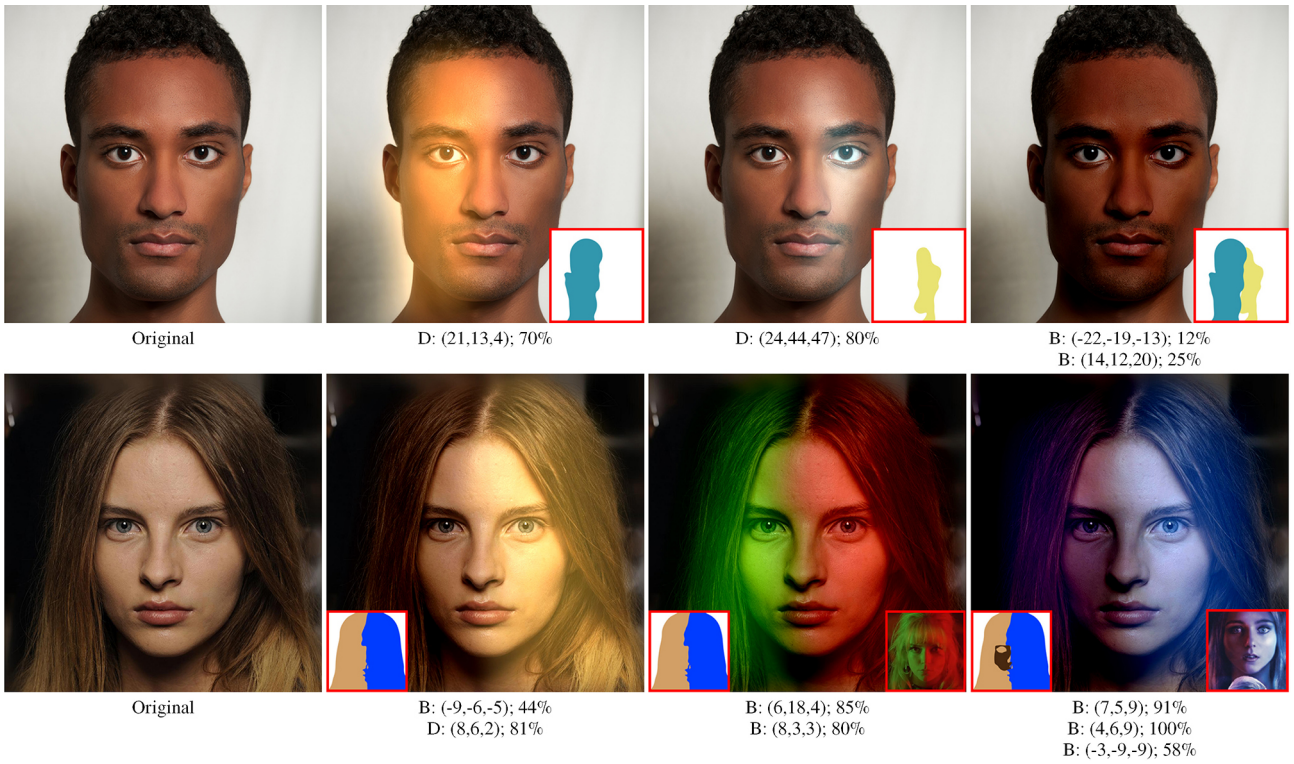


Fig. 11 Light editing. We apply linear dodge and linear burn to the middle-level PRs to modify highlights and shadows. D and B stand for linear dodge and linear burn, respectively. The 3-tuples are the integral of Laplacians of the PR and the percentage is the alpha value (opacity). We show the PR primitives and the references as insets.

Fig. 12 Expression editing by changing the geometries of DCs in the base level. The first column shows the original images, and others are editing results. We show the references and the original and updated DCs as insets.

linear blending to modify highlights and shadows (see Fig. 11). Though there are a large number of pixel-sized PRs in the top level, they are not meant for direct editing by the user. Turning off fPRs yields face retouching results. As Fig. 13 shows, our retouched results are of higher quality than an artist's manual

Fig. 13 Comparing our face retouching results with the L_1 smoothing method [45] and manual editing by professional artists [7]. Our method simply removes fPRs, and keeps other primitives unchanged. The L_1 smoothing method can keep illumination, but often over-smooths salient features. Our method can effectively preserve illumination, yielding more realistic results than the L_1 smoothing method. Our results are also better than manual retouching, as manual editing rarely reaches pixel-level precision and often compromises original features, such as highlights. Quantitative evaluations in terms of FLIP, IS-FLIP, and SSIM confirm this observation.

retouching [7] and L1-smoothing [45]. In geometry editing, when diffusion curves in the base level are changed, we adopt a deep neural network model to generate residual PRs automatically (see Fig. 12). Also applying the DC masks to the changed DCs, we can remove lights in the generated residual Poisson regions (see Fig. 8(e)).

7.6 Comparison to existing PVG vectorization

The PVG vectorization method of Ref. [25] converts a portrait photo into a PVG with only two levels: the base level consists of diffusion curves for the low-frequency skin and hair colors, and the top level contains everything else. Their 2-level decomposition works well for color transfer, which involves diffusion curves only. However, since their method does not separate illumination and facial details from skin color, it is difficult apply their method to illumination editing, geometry editing, and face retouching. While we adopt the same method for transferring boundary colors between diffusion curves as in Fu et al. [25], our color transfer results are better in terms of both a visual comparison and quantitative measures as shown later in Fig. 15. This is because our DC extraction can separate illumination from low-frequency colors better than Fu et al. [25].

7.7 Comparison to deep learning based methods

The smart contour method [28] uses a cascade of two deep neural networks, where the low-frequency network encodes a raster image as a sparse set of contours, which are associated with color gradients, for overall colors and geometry, and the high-frequency network generates residuals. Their method is flexible and efficient in that it allows the user to freely edit the contours, including adding and deleting contours and changing their geometry, and obtains photo-realistic results in near real time. Our method also allows the user to edit geometry by changing diffusion curves and then automatically generates residuals using deep learning. As Fig. 14 shows, both methods produce a similar number of vector primitives (diffusion curves and smart contours) in the base level. However, thanks to Poisson regions in the middle level, our method allows the user to edit illumination in an easy and intuitive manner, whereas their method cannot. We also observe that our method can better preserve fine details in image reconstruction thanks to fPRs in the top level.

Moreover, our hierarchical representation provides an integral framework for portrait coarse-to-fine editing, which conjoins the advantages of both vectorization and deep learning based methods. Apart from expression manipulation requiring high-frequency signals, the other presented editing procedures only involve vector primitives (curves or regions, i.e., DCs or PRs) modification and PVG

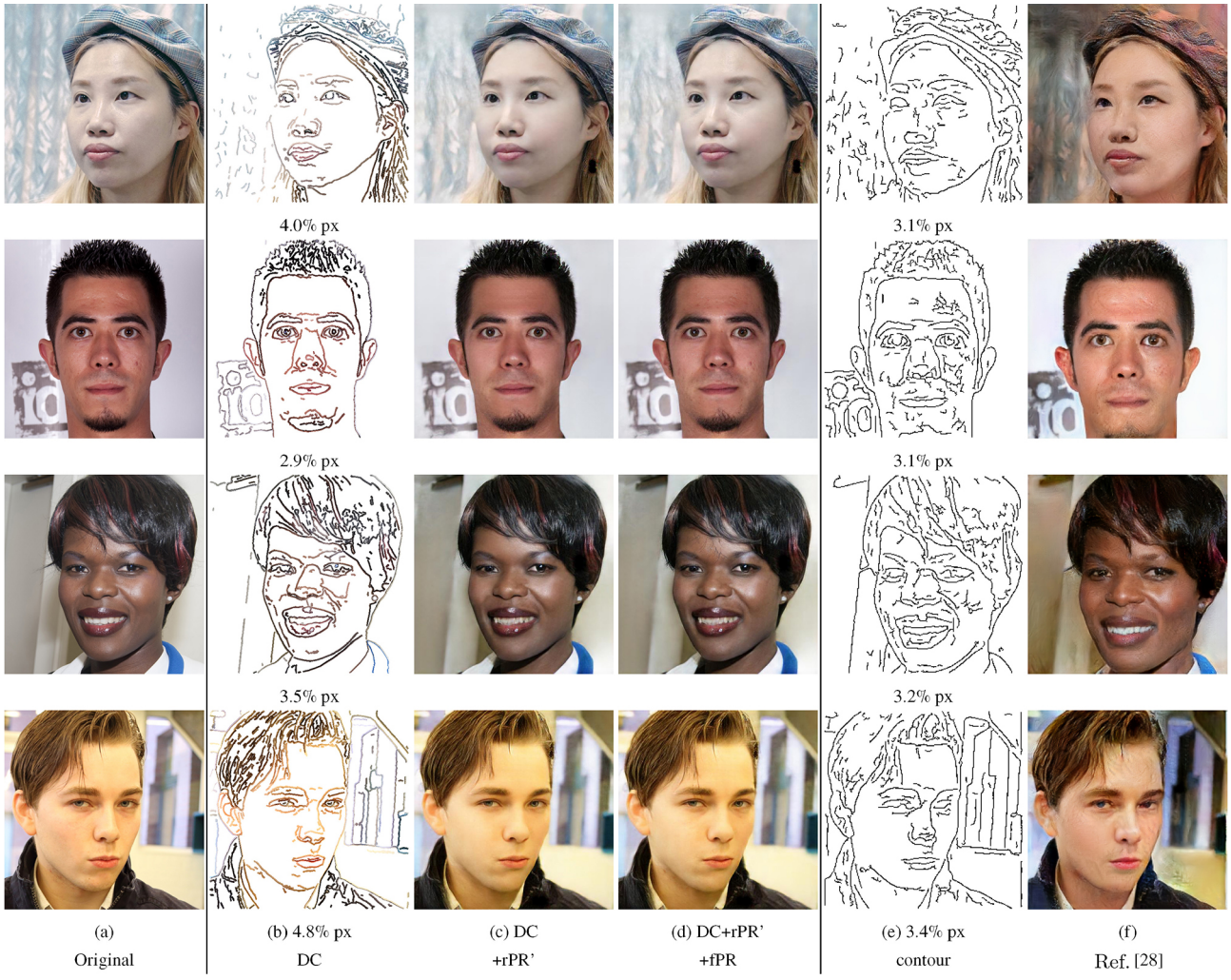


Fig. 14 Image reconstruction comparison with deep sparse, smart contours [28]. Our method produces a similar number of diffusion curves to smart contours. To make a fair comparison, we generated the residual Poisson regions rPR' using the deep neural network. We observe that our method can better preserve lights and colors than their method even without fPRs (compare (c) to (f)). The pixel ratios show the complexity of vector primitives (DC or contours).

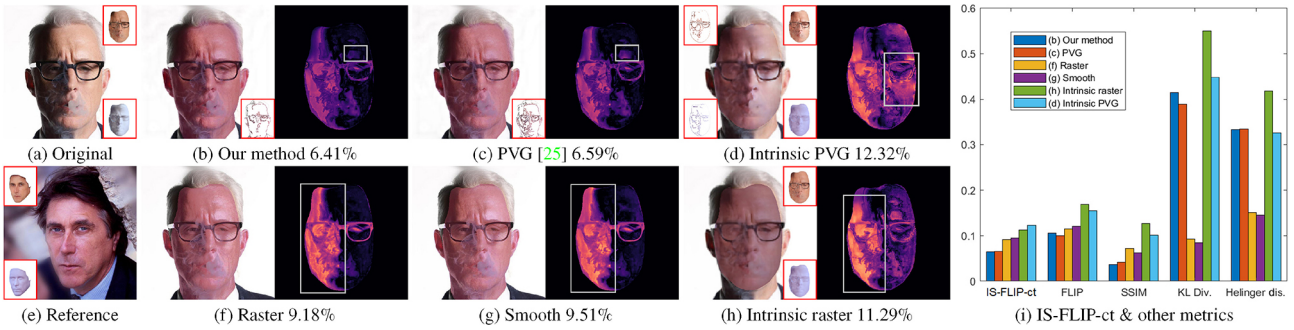


Fig. 15 Facial color transfer on a challenging case with a wide range of brightness. We show the results of (b) our method, (c) Fu et al.'s method [25], (d) intrinsic decomposition [11] applied to PVG [25], (f) raster image OMT, (g) OMT combined with L1-smoothing [45], and (h) OMT combined with intrinsic image [11]. We also visualize the IS-FLIP-ct error maps next to the results. The fewer facial details in the error maps, the higher the quality of color transfer. (i) Values of several metrics, including FLIP, SSIM, KL divergence, and Helinger distance. Both the quality metric IS-FLIP-ct and visual error maps confirm our method to be superior to other approaches.

rendering. Compared to deep learning methods, vectorization based editing can provide more subtle control. See, for example, Fig. 10: we can control exactly which hairlines are modified. It also has low computational and memory costs since the editing process does not involve loading or running any neural models during vector primitive modification and rendering.

7.8 Failures

Our method may fail to vectorize images if the input is blurred or has low contrast, as it cannot then extract sufficient DCs to represent salient facial features. Furthermore, when transferring colors for face or hair, it may lead to artifacts due to the poor face parsing. Figure 16 shows such examples.

7.9 Further results

Figure 17 shows how our method can handle different cases in one framework including different head poses and special cases including beards and glasses.

8 Conclusions

We have developed an automatic method for vectorizing portrait images. In contrast to existing methods that generate a large number of diffusion curves for reconstructing fine details of the input, our method produces a modest number of editable vector primitives without compromising accuracy. We organize the primitives into three levels. The base level is a set of sparse diffusion curves for representing low-frequency colors and salient facial features. The

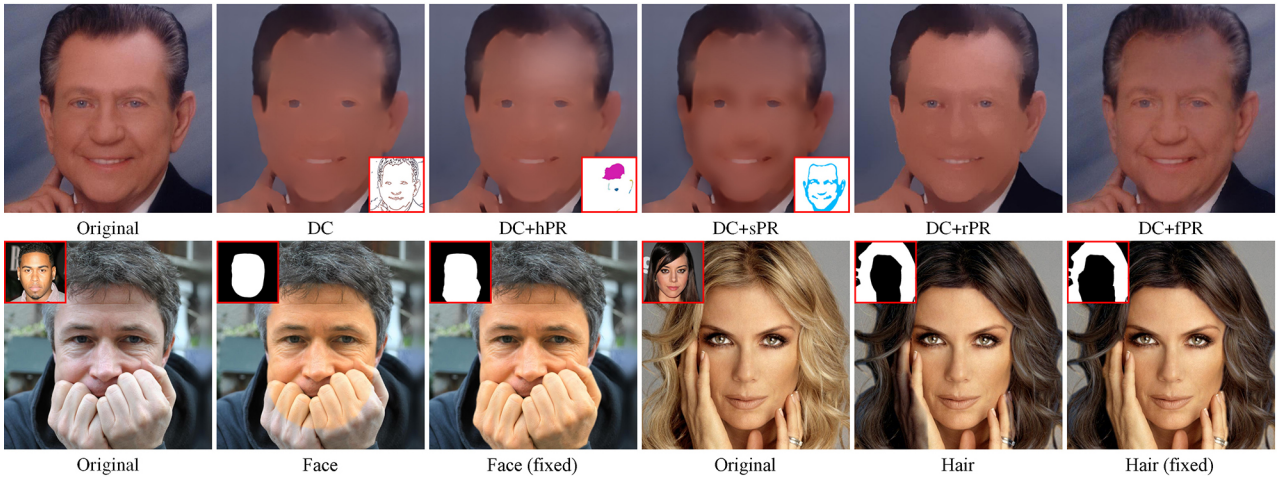


Fig. 16 Failures. Top: vectorization fails since the vector primitives in the base level are not able to capture facial features due to blurred input affecting edge detection. Bottom: color transfer fails for skin and hair due to poor segmentation results in face parsing; it also shows fixed results after improving the face or hair masks manually.

Fig. 17 More representative results for various cases including different head poses and special cases like bear and glasses. The updated result is obtained by transferring color (highlights) to the references shown in the small insets. Our method can handle various situations with one framework.

middle level consists of a few large Poisson regions for specular highlights and shadows. The top level contains pixel-sized Poisson regions for fine details and high-frequency residuals. We demonstrate that the hierarchical vector representation facilitates portrait editing, such as color transfer, lighting, and expression editing.

Although our proposed method has a range of desirable properties, such as its full automation and its high flexibility, there are also open issues that we aim to address in future. For example, we can further improve the hierarchical vectorization by adding other semantic levels for additional facial features, such as tattoos. We only provide 3 simple blending functions: alpha blending, linear dodge, and linear burn, in our current implementation. In future, we aim to add more linear and non-linear blending modes to further expand applicability of PVG. We also hope to extend our method from still images to videos by considering spatial and temporal coherence of diffusion curves and Poisson regions between frames. Last but not least, our current method is particularly designed for portrait images, thanks to the availability of various portrait-tailored computational tools for tasks such as segmentation and highlight removal. We believe that the proposed hierarchical PVG vectorization framework is general and can be extended to other images given proper pre-processing tools.

Appendix

A Training of portrait-tailored DexiNed model

As discussed in Section 4, we train the portrait-tailored DexiNed model on our manually annotated dataset for DC extraction, using the code from `https://github.com/xavysp/DexiNed.git`. We use the model trained on manually annotated data to annotate 500 randomly sampled portrait images, which is then combined with the manually annotated data to train the final model. The hyper-parameters used for training the model are summarized in Table 4.

Table 4 Hyper-parameters used to train our portrait-tailored DexiNed model with the device of 1 Nvidia Tesla v100 16 GB GPU

Hyper-parameter	Value
Learning rate	1E-4
Weight decay	1E-5
Optimizer	Adam
Optimizer betas	(0.9, 0.999)
No. of training epochs	25
Batch size	16
No. of outdoor images	250
No. of manually annotated portrait images	27
No. of model annotated portrait images	500

B Training details of the Pix2pixHD-based rPR generation model

As discussed in Section 5.3, we train a pix2pixHD model on our dataset, using code from `https://github.com/NVIDIA/pix2pixHD.git`, to generate the rPRs taking diffusion curve images as input. The pix2pixHD model adopts a conditional GAN architecture for image-to-image translation. We use the global generator network of the original pix2pixHD model as our generator, which consists of a convolutional front-end, 9 residual blocks, and a transposed convolutional back-end. As in the original pix2pixHD model, 2 multi-scale discriminators are used in our model. In addition to the discrimination losses, we also minimize the VGG perceptual loss and discriminator feature matching loss in the same way as Ref. [42]. See Ref. [42] for more details of the model's architecture. 10,000 images are sampled from the FFHQR dataset for model training. We follow the steps described in Sections 4 and 5.3 to generate the diffusion curve images and the rPRs, respectively. The pix2pixHD model is then trained using the diffusion curve images as inputs to construct the rPRs. The hyper-parameters used for training the model are summarized in Table 5.

Table 5 Hyper-parameters used to train our pix2pixHD-based rPR generation model using 4 Nvidia Tesla v100 16GB GPUs

Hyper-parameter	Value
Learning rate	2.5E-4
Optimizer	Adam
Optimizer betas	(0.9, 0.999)
No. of training epochs	80
Batch size	16
No. of input label channels	3
VGG loss enabled	true
Pretrained VGG	VGG19
No. of training images	10,000
Image size	512 × 512

Acknowledgements

This project was supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (RG20/20), the National Natural Science Foundation of China (61872347), and the Special Plan for the Development of Distinguished Young Scientists of ISCAS (Y8RC535018).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article. The author Ying He is the Associate Editor of this journal.

References

[1] Orzan, A.; Bousseau, A.; Winnemöller, H.; Barla, P.; Thollot, J.; Salesin, D. Diffusion curves. *ACM Transactions on Graphics* Vol. 27, No. 3, 1–8, 2008.

[2] Finch, M.; Snyder, J.; Hoppe, H. Freeform vector graphics with controlled thin-plate splines. *ACM Transactions on Graphics* Vol. 30, No. 6, 1–10, 2011.

[3] Xie, G. F.; Sun, X.; Tong, X.; Nowrouzezahrai, D. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Transactions on Graphics* Vol. 33, No. 6, Article No. 230, 2014.

[4] Zhao, S.; Durand, F.; Zheng, C. X. Inverse diffusion curves using shape optimization. *IEEE Transactions on Visualization and Computer Graphics* Vol. 24, No. 7, 2153–2166, 2018.

[5] Hou, F.; Sun, Q.; Fang, Z.; Liu, Y. J.; Hu, S. M.; Qin, H.; Hao, A. M.; He, Y. Poisson vector graphics (PVG). *IEEE Transactions on Visualization and Computer Graphics* Vol. 26, No. 2, 1361–1371, 2020.

[6] Lee, C. H.; Liu, Z. W.; Wu, L. Y.; Luo, P. MaskGAN: Towards diverse and interactive facial image manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5548–5557, 2020.

[7] Shafaei, A.; Little, J. J.; Schmidt, M. AutoRetouch: Automatic professional face retouching. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 989–997, 2021.

[8] Bell, S.; Bala, K.; Snavely, N. Intrinsic images in the wild. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 159, 2014.

[9] Cheng, Z. A.; Zheng, Y. Q.; You, S. D.; Sato, I. Non-local intrinsic decomposition with near-infrared priors. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2521–2530, 2019.

[10] Zhou, H.; Yu, X.; Jacobs, D. GLoSH: Global-local spherical harmonics for intrinsic image decomposition.

In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 7819–7828, 2019.

[11] Sengupta, S.; Kanazawa, A.; Castillo, C. D.; Jacobs, D. W. SfSNet: Learning shape, reflectance and illuminance of faces 'in the wild'. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 6296–6305, 2018.

[12] Shu, Z. X.; Yumer, E.; Hadap, S.; Sunkavalli, K.; Shechtman, E.; Samaras, D. Neural face editing with intrinsic image disentangling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5444–5453, 2017.

[13] Sun, J.; Liang, L.; Wen, F.; Shum, H. Y. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics* Vol. 26, No. 3, 11–es, 2007.

[14] Lai, Y. K.; Hu, S. M.; Martin, R. R. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics* Vol. 28, No. 3, Article No. 85, 2009.

[15] Chen, K. W.; Luo, Y. S.; Lai, Y. C.; Chen, Y. L.; Yao, C. Y.; Chu, H. K.; Lee, T. Y. Image vectorization with real-time thin-plate spline. *IEEE Transactions on Multimedia* Vol. 22, No. 1, 15–29, 2020.

[16] Liao, Z. C.; Hoppe, H.; Forsyth, D.; Yu, Y. Z. A subdivision-based representation for vector image editing. *IEEE Transactions on Visualization and Computer Graphics* Vol. 18, No. 11, 1858–1867, 2012.

[17] Zhou, H. L.; Zheng, J. M.; Wei, L. Representing images using curvilinear feature driven subdivision surfaces. *IEEE Transactions on Image Processing* Vol. 23, No. 8, 3268–3280, 2014.

[18] Zhang, S. H.; Chen, T.; Zhang, Y. F.; Hu, S. M.; Martin, R. R. Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics* Vol. 15, No. 4, 618–629, 2009.

[19] Boyé, S.; Barla, P.; Guennebaud, G. A vectorial solver for free-form vector gradients. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 173, 2012.

[20] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. PAMI-8, No. 6, 679–698, 1986.

[21] Lu, S. F.; Jiang, W.; Ding, X. F.; Kaplan, C. S.; Jin, X. G.; Gao, F.; Chen, J. Z. Depth-aware image vectorization and editing. *The Visual Computer* Vol. 35, Nos. 6–8, 1027–1039, 2019.

[22] Shu, Z. X.; Hadap, S.; Shechtman, E.; Sunkavalli, K.; Paris, S.; Samaras, D. Portrait lighting transfer using a mass transport approach. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 2, 2017.

[23] Zhou, H.; Hadap, S.; Sunkavalli, K.; Jacobs, D. Deep single-image portrait relighting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 7193–7201, 2019.

[24] Zhang, X. M.; Fanello, S.; Tsai, Y. T.; Sun, T. C.; Xue, T. F.; Pandey, R.; Orts-Escolano, S.; Davidson, P.; Rhemann, C.; Debevec, P.; et al. Neural light transport for relighting and view synthesis. *ACM Transactions on Graphics* Vol. 40, No. 1, Article No. 9, 2021.

[25] Fu, Q.; He, Y.; Hou, F.; Zhang, J. Y.; Zeng, A. X.; Liu, Y. J. Vectorization based color transfer for portrait images. *Computer-Aided Design* Vol. 115, 111–121, 2019.

[26] Liao, J.; Yao, Y.; Yuan, L.; Hua, G.; Kang, S. B. Visual attribute transfer through deep image analogy. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 120, 2017.

[27] Afifi, M.; Brubaker, M. A.; Brown, M. S. HistoGAN: Controlling colors of GAN-generated and real images via color histograms. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7937–7946, 2021.

[28] Dekel, T.; Gan, C.; Krishnan, D.; Liu, C.; Freeman, W. T. Sparse, smart contours to represent and edit images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3511–3520, 2018.

[29] Lu, Z. H.; Hu, T. H.; Song, L. X.; Zhang, Z. X.; He, R. Conditional expression synthesis with face parsing transformation. In: Proceedings of the 26th ACM International Conference on Multimedia, 1083–1091, 2018.

[30] Shih, Y.; Paris, S.; Barnes, C.; Freeman, W. T.; Durand, F. Style transfer for headshot portraits. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 148, 2014.

[31] Sheng, L.; Lin, Z. Y.; Shao, J.; Wang, X. G. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8242–8250, 2018.

[32] Portenier, T.; Hu, Q. Y.; Szabó, A.; Bigdeli, S. A.; Favaro, P.; Zwicker, M. Faceshop. *ACM Transactions on Graphics* Vol. 37, No. 4, Article No. 99, 2018.

[33] Jo, Y.; Park, J. SC-FEGAN: Face editing generative adversarial network with user's sketch and color. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 1745–1753, 2019.

[34] Chen, S. Y.; Liu, F. L.; Lai, Y. K.; Rosin, P. L.; Li, C. P.; Fu, H. B.; Gao, L. DeepFaceEditing: Deep face generation and editing with disentangled geometry and appearance control. *ACM Transactions on Graphics* Vol. 40, No. 4, Article No. 90, 2021.

[35] Thanh-Tung, H.; Tran, T. Catastrophic forgetting and mode collapse in GANs. In: Proceedings of the International Joint Conference on Neural Networks, 1–10, 2020.

[36] Bang, D.; Shim, H. MGGAN: Solving mode collapse using manifold-guided training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2347–2356, 2021.

[37] Shen, H. L.; Zheng, Z. H. Real-time highlight removal using intensity ratio. *Applied Optics* Vol. 52, No. 19, 4483, 2013.

[38] Leordeanu, M.; Sukthankar, R.; Sminchisescu, C. Efficient closed-form solution to generalized boundary detection. In: *Computer Vision – ECCV 2012. Lecture Notes in Computer Science, Vol. 7575*. Fitzgibbon, A.; Lazebnik, S.; Perona, P.; Sato, Y.; Schmid, C. Eds. Springer Berlin Heidelberg, 516–529, 2012.

[39] Soria, X.; Riba, E.; Sappa, A. Dense extreme inception network: Towards a robust CNN model for edge detection. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 1912–1921, 2020.

[40] Xie, Q. Z.; Luong, M. T.; Hovy, E.; Le, Q. V. Self-training with noisy student improves ImageNet classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10684–10695, 2020.

[41] Zoph, B.; Ghiasi, G.; Lin, T. Y.; Cui, Y.; Liu, H. X.; Cubuk, E. D.; Le, Q. V. Rethinking pre-training and self-training. In: Proceedings of the 34th International Conference on Neural Information Processing Systems, Article No. 323, 3833–3845, 2020.

[42] Wang, T. C.; Liu, M. Y.; Zhu, J. Y.; Tao, A.; Kautz, J.; Catanzaro, B. High-resolution image synthesis and semantic manipulation with conditional GANs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8798–8807, 2018.

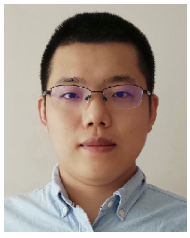
[43] Andersson, P.; Nilsson, J.; Akenine-Möller, T.; Oskarsson, M.; Åström, K.; Fairchild, M. FLIP: A difference evaluator for alternating images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* Vol. 3, No. 2, Article No. 15, 2020.

[44] Wang, Z.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* Vol. 13, No. 4, 600–612, 2004.

[45] Bi, S.; Han, X. G.; Yu, Y. Z. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics* Vol. 34, No. 4, Article No. 78, 2015.

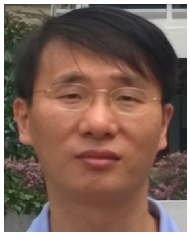
[46] Favreau, J. D.; Lafarge, F.; Bousseau, A. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Transactions on Graphics* Vol. 36, No. 6, Article No. 180, 2017.

Qian Fu received her B.S. and M.S. degrees in computer science & technology from Beijing Normal University, and her Ph.D. degree in computer science & engineering from Nanyang Technological University. She is currently a research scientist with Data61, Commonwealth Scientific and Industrial Research Organisation, Australia. Her research interests fall into the areas of computer graphics, computer vision, and computational geometry.



Linlin Liu received his B.S. degree in information systems from the National University of Singapore. He is currently a Ph.D. candidate at Interdisciplinary Graduate School, Nanyang Technological University, in the Joint Ph.D. Program between Alibaba and Nanyang Technological University. His research interests include image synthesis, representation learning, and natural language processing.

Fei Hou received his Ph.D. degree in computer science from Beihang University in 2012. He is currently a research associate professor of Institute of Software, Chinese Academy of Sciences. He was a postdoctoral researcher at Beihang University from 2012 to 2014 and a research fellow in the School of Computer Science and Engineering, Nanyang Technological University from 2014 to 2017. His research interests include geometry processing, image-based modeling, data vectorization, and medical image processing.



Ying He received his B.S. and M.S. degrees in electrical engineering from Tsinghua University, China, and his Ph.D. degree in computer science from Stony Brook University, USA. He is currently an associate professor in the School of Computer Science and Engineering, Nanyang Technological University. His research interests fall into the general areas

of visual computing and he is particularly interested in problems which require geometric analysis and computation.