On poisson vector graphics : displacement mapping, vectorization and color transfer

Fu, Qian

2021

Fu, Q. (2021). On poisson vector graphics : displacement mapping, vectorization and color transfer. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/145976>

<https://doi.org/10.32657/10356/145976>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 23:21:45 SGT

ON POISSON VECTOR GRAPHICS:
DISPLACEMENT MAPPING, VECTORIZATION AND
COLOR TRANSFER

Qian Fu

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy (Ph.D)

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

12 August 2020

.....
Date

.....
Qian Fu

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

12 August 2020

.....
Date

.....
Ying He

Authorship Attribution Statement

This thesis contains material from seven papers published in the following peer-reviewed journals in which I am listed as an author.

- Chapter 4 is published as **Qian Fu, Fei Hou, Qian Sun, Shi-Qing Xin, Yong-Jin Liu, Wencheng Wang, Hong Qin and Ying He**, “Decorating 3d models with poisson vector graphics,” *Computer-Aided Design*, vol. 102, pp. 1–11, 2018.
- Chapter 5 is published as **Qian Fu, Ying He, Fei Hou, Juyong Zhang, Anxiang Zeng, and Yong-Jin Liu**, “Vectorization based color transfer for portrait images,” *Computer-Aided Design*, vol. 115, pp. 111–121, 2019.
- Chapter 6 is published as **Qian Fu, Ying He, Fei Hou, Qian Sun, Anxiang Zeng, Zhenchuan Huang, Juyong Zhang, and Yong-Jin Liu**, “Poisson Vector Graphics (PVG)-Guided Face Color Transfer in Videos,” *Pending minor revision, IEEE computer graphics and applications*, 2020.

The contributions of the co-authors are as follows:

- Prof. Ying He provided the initial research directions.
- Prof. Ying He, Dr. Fei Hou, and Prof. Juyong Zhang contributed to discuss the construction of the technical framework.
- Dr. Shi-Qing Xin, Dr. Wensheng Wang, Mr. Anxiang Zeng and Mr. Zhenchuan Huang provided algorithmic association to implement some pre-process.
- Prof. Qian Sun offered artistic support as an artist in the experiments.
- I derived the relevant mathematical equations in the manuscripts with assistance from Prof. Ying He and Dr. Fei Hou.
- I wrote the MATLAB and C++ codes to build systems and obtain the results presented in the manuscripts.

- I prepared the manuscript drafts. The manuscripts were revised mainly by Prof. Ying He. Prof. Qin Hong and Prof. Yong-Jin Liu provided partial revision and proof reading.

12 August 2020

.....
Date

.....
Qian Fu

Abstract

Poisson Vector Graphics (PVG) generalize the popular Diffusion Curves (DC) by appending two new geometric primitives — Poisson Region (PR) and Poisson Curve (PC). It is an effective graphic designing tool which extends Laplace’s equation to Poisson’s equation. DC is a two-sided curve with colors defined on both sides that diffuse colors over the image to produce consistent color except for sharp features along the boundaries. PC is to produce discontinuous colors across the curve, whereas PR is to style smooth shading effect among the predefined region. But DC suffers from C^{-1} or C^0 continuity. In comparison, PR is C^1 continuous, that can establish smooth shading details. Furthermore, DC based vectorization works for both natural and art images quite well. But it usually requires dense curves for high-quality reconstruction, that makes post-editing difficult. Instead, PVG based vectorization converts raster image into more sparse curves, that results in easier and more convenient post-editing. PVG only extracts the curves of salient features as DCs to represent the low-frequency colors, and encode the high-frequency details in PRs, which allows to separately handle the low-frequency and high-frequency information with more flexibility. Inspired by PVG’s good characteristics, we apply PVG into a few relevant fields to optimize existing methods.

In the first part of this thesis, associated with the smooth nature provided by PR, we propose a novel approach of decorating 3D surfaces to composite smooth geometric details. Since PVG can produce an artistic work with sparse curves, user can directly edit synthetic geometric details and colors on the models. In our system, user can create discontinuous or sharp features by using DCs and PCs, and synthesize smooth details by PRs. PR is superior than DC since its smooth nature, and it also supports hierarchical level-of-detail editing. Comparing our method with other commonly-used technologies, their methods generally depend on global/local parameterization. Our method supports

parameterization-free decoration, meanwhile diminishing the overall computational cost. In order to render PVG effectively on models, we generalize the Poisson solver from 2D domain to 3D meshes. Our solver is a local solver that designed to flexibly and versatily render PVG. Our results manifest effectiveness of the proposed approach on both real-world and synthetic models.

In the second part of the thesis, based on the property of DC that represents the salient colors of vector images, we leverage PVG to vectorize portrait images and then transfer color between the source and reference. Hereby color transfer is calculated in sparse curves rather than the entire images. First, we extract facial masks through a pre-trained neural network, and vectorize each source or reference images with sparse DCs to encode the salient (low-frequency) colors, and then calculate PRs to encode the residual (high-frequency) details. Next, we transfer DCs' boundary color between the source and reference images by applying the optimal mass transport (OMT). Finally, we render the updated PVG. Compared with existing methods that require 3D coordinate positions/normals or assume dense correspondences and alike poses between input and reference images, our method does not require pixel-to-pixel dense correspondence or 3D information, and it can process portrait images with greatly different color and postures.

In the third part of the thesis, we present a simple yet effective algorithm for automatically transferring face colors in portrait videos that generalize the method of transferring color between portrait images. We extract the facial features and vectorize the faces in the input video using PVG. Then we transfer the face color of a reference image/video to the first frame of the input video by applying OMT between the boundary colors of DCs. Next the boundary color of the first frame is transferred to the subsequent frames by matching the curves. Finally, we render the video with the color-changed DCs. Thanks to our efficient DC matching algorithm, transferring colors for the vectorized video takes less than 1 millisecond per frame. Our method is particularly desired for frequent transfer from multiple references due to its information reuse nature. We demonstrate the efficacy of our method on image-to-video transfer and color swap in videos.

PVG is an effective mathematic tool that can be applied into various situations to inspire better resolutions, thanks to its good properties. In this thesis, we present three relevant works. Our future direction is to explore more significative applications of PVG, associated with state-to-art technologies like deep-learning techniques.

Acknowledgments

Instant, my four-year doctoral study is coming to an end. Looking back, I have experienced setbacks and joys, and I especially appreciate the people who have supported me and accompanied me on this journey.

First, I am pleased to express my genuine gratitude to my supervisor Prof. He Ying. I always feel that it is my greatest luck to be his student. At the beginning, I was unfamiliar with this new scientific field. It was Prof. He who gave me patient instructions on how to research and how to formulate an effective research plan, that made me gradually become a person who can independently undertake certain scientific research tasks. In normal times, Prof. He regularly meets with me every week to discuss the progress of my project and solve some of the problems I encounter in my research. When writing papers, Prof. He always helps me revise the papers, and puts forward valuable revision opinions, and at the same time teaches me how to write a qualified research paper in the progress. With my graduation quickly approaching, Prof. He helps me to analyze and plan my career in the future, so that I can calmly and actively face the turning point in my life.

Then, I would like to thank the colleagues in our laboratory. Dr. Hou Fei gives me guidance in the relevant theories through his rich knowledge, allowing me to quickly get started in the research field, and meanwhile helps me with some programming problems. Dr. Fang Zheng and Miss Du Jie, who are sitting beside me, have given me a lot of help and care in my daily life and work.

At last, I would also like to thank the Singapore government for supporting me with a Ph.D. scholarship that allows me to concentrate on my scientific research, and Nanyang Technological University for providing a good research environment in the School of Computer Science and Engineering. I also sincerely appreciate my parents for their constant trust and support over the past four years.

Contents

Abstract	v
Acknowledgments	vii
List of Figures	xi
List of Tables	xvii
List of Publications	xviii
1 Introduction	1
1.1 Background	1
1.2 Motivation and Objective	3
1.3 Contributions	6
1.4 Thesis Organization	7
2 Preliminaries	8
2.1 Poisson Vector Graphics	9
2.2 Harmonic B-splines	9
2.3 2D PVG Solver	11
2.4 3D PVG Mesh Solver	12
3 Related Work	15
3.1 Vector Graphics	15
3.2 Image Vectorization	16
3.3 Color Transfer	17
3.4 Style Transfer	18
3.5 Surface Decoration	19

4	Decorating 3D Models with Poisson Vector Graphics	21
4.1	Background	21
4.2	Algorithm	23
4.2.1	Overview	23
4.2.2	Discrete Green's Function	26
4.2.3	Geodesic Voronoi Tessellation	29
4.2.4	Control Coefficients	30
4.2.5	Basis Functions	30
4.3	Results	32
4.4	Summary	40
5	Vectorization based Color Transfer for Portrait Images	41
5.1	Background	41
5.2	Overview	43
5.3	Vectorization	43
5.4	Labelling Facial Features	46
5.5	Color Transfer with Optimal Mass Transport	48
5.6	Rendering	51
5.7	Results	52
5.8	Summary	59
6	Poisson Vector Graphics (PVG)-Guided Face Color Transfer in Videos	62
6.1	Background	63
6.2	Overview	64
6.3	Video Vectorization	65
6.3.1	Extracting Facial Features	67
6.3.2	Computing Boundary Color g for DCs	69
6.4	PVG-Guided Color Transfer	71
6.5	Results	72
6.6	Summary	77

7	Conclusions & Future Work	83
7.1	Conclusions	83
7.2	Future Work	85
	References	87

List of Figures

1.1	Poisson vector graphics: the primitives and corresponding rendering results.	2
1.2	Intersection examples of DCs and PRs on 3D models. (a)(c) PVG primitives. (b)(d) Normal map of the rendering results. The first row is two PRs intersecting with each other. The second row shows two DCs intersecting lead to non-smooth artificial effect, since their relative boundary conditions are usually “competing” against each other.	4
1.3	Poisson vector graphics guided color editing. The ice cream image (a) is vectorized into a set of diffusion curves (colored, solid lines) and Poisson regions (dotted regions) (b), where the former encodes the low-frequency color and the latter encodes the high-frequency details. To change the ice cream’s color, we only need to edit the boundary colors of the diffusion curves (d). Diffusing the new boundary colors produces an image with a totally new looking (c). Intuitively speaking, we formulate the region-based color editing into a problem of editing the color of its boundary. The only difference between (b) and (d) is the boundary color of DCs. . .	5
2.1	DC vs. PVG. PVG extends DC framework by appending two new primitives, PRs and PCs, to provide additional control on colors.	10
4.1	Transforming a cube model to a flag model with a simple set of PVG primitives involving 5 cDCs, 4 dDCs and 1 dPR. We use color map to visualize the displacement constraints.	24
4.2	DC vs. PR. We visualize f and g with the same color map as above, i.e., color and warm colors indicate negative and positive values respectively, green is zero.	25

4.3	On planar domain, thanks to the analytic solution of Green's function, PR (dashed curve) remains C^1 continuous along the boundary.	25
4.4	Illustration of our Poisson solver on a cube mesh $\mathcal{M}$. (a) Displacement constraints. (b) Knots $\{\mathbf{t}_i\}$ & Voronoi diagram $\{\mathcal{V}_i\}$. (c) Basis function $\psi_j(\mathbf{x})$. (d) Control points $\{\lambda_i\}$. (e) Harmonic B-spline function $u(\mathbf{x}) = \sum_{i=1}^n \lambda_i \psi_i(\mathbf{x})$	27
4.5	Decorating Bunny with three letters "SPM" and colorful patterns. With a set of DCs and PRs, we produce smooth and colorful reliefs. Owing to the <i>local</i> property of our solver, we can only build harmonic B-spline function and geodesic Voronoi tessellation in the computation regions as (d). . .	33
4.6	Decorating a stele model with convex patterns of dragon/auspicious clouds and concave Chinese calligraphy characters. By simply tuning Laplacians, we can control the height of the convex/concave reliefs. Comparing the reliefs produced by dDCs with those created by dPRs (i.e. the convex patterns of dragon/auspicious clouds and the concave Chinese calligraphy characters), it is obvious that our PRs can generate smooth features while DCs can only construct discontinues effects.	34
4.7	Decorating two vase models with smooth reliefs produced by dPRs. . . .	35
4.8	Decorating Sofa and Rhino with repeated reliefs.	36
4.9	Constructing a cactus from a sphere. We define PVG on two separate layers, where the first layer models the stems and the second layer the sharp spines. Since PR is C^1 continuous, the two layers can be seamlessly combined together. Note that it is hard to apply only DCs for level-of-detail modeling, since it is discontinuous along the curves.	37
4.10	Decorating and editing a Piggy Bank model. (a) shows the input model. (b) shows the first decoration result by using PVGs. (c) is the second result after changing color whose Voronoi diagram remains unchanged. (d) is the final result after changing dDC's shape of two clouds whose Voronoi diagram is locally updated with some additional knots and only the vertices $\mathbf{x}'$ inside these new Voronoi cells are recomputed by using equation $u(\mathbf{x}') = \sum_{\mathcal{V}_j \in N(\mathbf{x}')} \lambda_j \psi_j(\mathbf{x}')$. There are 1,073 knots in (g), and 214 additional knots are added in (h).	38

4.11	Modeling a tree with multi-level PVG. Given a rectangular domain (a), user sketches a circular dPR enclosed by a dDC whose Dirichlet boundary condition is 0. The resulting PVG is the tree's trunk (b). Then user sketches two other sets of DC and PR (c) to define two branches (d). Repeating this procedure a few times, we obtain the tree (e). Similar to other figures, we use color map to visualize the values of f and g , i.e., red for positive value and green for 0.	39
5.1	Examples of skin and hair color transfer with our method. The original images are shown in the first column of the first/third row and the reference images are shown in small insets. Note that our method does not require a dense correspondence between the input and the reference, therefore it can handle portraits with different postures.	44
5.2	Our algorithmic pipeline that taking skin color transfer as an example. The labeled DCs fall into three classes: skin (red), hair (blue) and background (green).	45
5.3	Vectorizing a portrait image. (a) The original image. (b) Extracted DCs with boundary colors. (c) The DCs represent only the low-frequency colors. (d) The PRs are pixel-level regions encoding high-frequency details. (e) The combination of DCs and PRs can produce a well-reconstructed vector image with high accuracy. (f) Label the DCs as 3 types: background (green), hair (blue) and skin (red).	46
5.4	A portrait image is segmented with 3 semantic labels: background, skin, and hair.	47
5.5	Color transfer for skin and hair with different iterations N . The first column of the first/third row show the input images and the last column of the second/last row are the references. The middle columns (from front to back) are the results with $N = 1, 5, 10, 20, 30, 50$ respectively.	50
5.6	Skin color transfer results. The red frames indicate original images, and the others are the transferred results.	53
5.7	Skin color transfer between oil/watercolor paintings and real portraits. The first column of the first two rows and the mini images are original. .	55

5.8	Color transfer between portraits of different head positions. The images in red frames are original.	56
5.9	Our method can maintain original details such as sunglasses, freckles, and tattoos.	57
5.10	Comparison with the state-of-the-art related works [1] and [2].	58
5.11	Hair color transfer for photos and paintings. The first column is the inputs, the second and fourth rows are the references.	59
5.12	Color transfer for mouths and eyes. Left: the original image; middle and right: the transferred results.	60
5.13	Our method fails on portraits with significantly different color distributions. Note the input image has a very low contrast, whereas the reference image has a high contrast. Tuning the PRs can reduce the artifacts. . .	61
6.1	Algorithmic pipeline. Thanks to its information reuse nature, our algorithm is highly desired for frequent color transfer from multiple reference images.	65
6.2	Face vectorization. (a) shows two frames of an input video. We detect 106 feature points for each frame (b) and obtain a face mask by extracting skin colors (c). We use the feature points and the boundary of the mask as diffusion curves (d). We compute their boundary colors (e) by solving a linear least square problem. To improve the performance, we use harmonic B-spline solver which is built upon a quad-tree data structure. DCs encode the low-frequency (LF) colors (g) and PRs encode the high-frequency (HF) details. (f) shows the color-coded PRs. Using DCs and PRs, we can faithfully reconstruct the input video (h). Moreover, by changing the coefficients of PRs, we can edit contrast and highlight (i-j).	68
6.3	DC boundary color matching. The boundary color of the first frame g_1 is computed by Eqn. (6.3). With the boundary color g_i for the i -th frame and the correspondence map $\pi_i : \gamma_i \rightarrow \gamma_{i+1}$, we compute the boundary color g_{i+1} by the inverse map π_i^{-1}	70

6.4	Iterative computation of the optimal mass transport map $\mathcal{T}$. We show a few representative intermediate results $I'_1 _k$ of the first frame in the top row and their corresponding histograms $H_{\mathcal{T}^{(k)}(\gamma_1)}$ in the bottom row. The iterative algorithm converges quickly and produces the final result in only 20 iterations.	72
6.5	Transferred results of a celebrity with tanned skin color. The person is transferred from 4 references. We observe that the transferred colors are highly consistent and show the first, the second, the middle and the last frames as the representative frames. Note that the first frame is computed via OMT and the subsequent frames are obtained via DC matching. . . .	74
6.6	Transferred results of a celebrity with fair skin color. The person is transferred from 4 references.	75
6.7	Transferred results of a celebrity with dark skin color. The person is transferred from 4 references.	76
6.8	Transferring face colors to person wearing glasses. We extract the facial features to form the diffusion curves and then compute their boundary colors by solving a linear least square for the first frame. DCs encode the low-frequency skin colors and PRs encode the high-frequency details (including glasses). Given a vectorized reference image (with or without wearing glasses), we only require to transfer the boundary colors of DCs and retain PRs unchanged.	77
6.9	The transferred results of persons wearing glasses.	78
6.10	Face color swap. Row 1 shows two frames of the original video. Row 2 shows the results by swapping the boundary colors of DCs and keep the PRs unchanged. Row 3 shows the results by updating PRs with coefficients 0.8 (Lemon) and 1.2 (Cuomo), respectively. We observe that the results with updated PR coefficients have more natural contrast and highlight. . .	79

6.11	Quality analysis. We transfer Reference A and Reference B to the first and second half of the input video and compute the difference of PSNR between the original video and the transferred video. The more constant the difference, the more consistence with the input video, hence the less dependence of the frame index and the less flickering artifacts. The x -axis shows the frame index and the y -axis shows the squared difference of PSNR between the original video and the transferred video. Both our curve (red) and the curve of the image-based OMT (green) are almost constant, indicating the comparable quality of the results. However, our method runs significantly faster. We observe that simply applying Fu's method [3] to the video frames produces results with flickering artifacts. The large variation of the blue curve confirms Fu's results are unstable. The PSNR deviations of our method, the image-based OMT and Fu's method are 0.0656, 0.0838, and 0.6248, respectively.	80
6.12	A failed case due to wrong skin extraction. The skin color clustering algorithm computes wrong masks (green) which include part of her hair. As a result, skin colors are also transferred to the hair. The artifacts are highlighted in the red boxes.	82
7.1	The overall architecture of our system.	86

List of Tables

2.1	Notations	8
4.1	Statistics of results. We report the number of DCs and PRs used. The prefixes “c” and “d” distinguish the primitives for color and displacement.	39
6.1	Statistics. The timings were measured in milliseconds.	73

List of Publications

1. Qian Fu, Fei Hou, Qian Sun, Shi-Qing Xin, Yong-Jin Liu, Wencheng Wang, Hong Qin and Ying He, “Decorating 3d models with poisson vector graphics,” *Computer-Aided Design*, vol. 102, pp. 1–11, 2018.
2. Qian Fu, Ying He, Fei Hou, Juyong Zhang, Anxiang Zeng, and Yong-Jin Liu, “Vectorization based color transfer for portrait images,” *Computer-Aided Design*, vol. 115, pp. 111–121, 2019.
3. Qian Fu, Ying He, Fei Hou, Qian Sun, Anxiang Zeng, Zhenchuan Huang, Juyong Zhang, and Yong-Jin Liu, “Poisson Vector Graphics (PVG)-Guided Face Color Transfer in Videos,” Pending minor revision, *IEEE computer graphics and applications*, 2020.

Chapter 1

Introduction

1.1 Background

Poisson vector graphics (PVG) is a useful authoring tool that provides intuitive and simple editing for the users. This representation utilizes three different types of primitives, which are diffusion curves (DCs) for rendering low-frequency colors, Poisson curves (PCs) for styling discontinuous effect, and Poisson regions (PRs) for creating smooth shading. Fig. 1.1 shows the PVG primitives and their rendering results. Intuitively, motivated by these good properties of PVG, we are interested to use them into the relevant areas as alternatives with an improvement over other existing methods in some ways.

In the modern world, taking portrait and self-portrait photographs and videos has become a cultural phenomenon that more and more people have turned to photo sharing on social media. There are a lot of programs and apps allowing the user to retouch the photos, such as editing eyes, brightening skin, removing wrinkles, reshaping face, etc. Since it is a time-consuming work to manually tune colors even for skilled users, color transfer provides a pragmatic approach to tune colors of an input image (video) to harmonize with tone of the reference images (videos). Color transfer has been studied extensively since 2000. Reinhard et al. [4] pioneered the color transfer method by matching the mean and standard deviation of color distributions of the sources and references. Hwang et al. [5] applied probabilistic moving least square for color transfer between images with the same scene but different camera settings and illumination conditions. Shih et al. [2] presented a good work for portraits' style transferring which matched the global lighting and the local contrast by transferring the local statistics. Their method can

produce visually pleasing results, however it requires a dense correspondence between the source and the reference image. Shu et al. [1] developed a method to transfer both lighting and colors for portrait photos. They applied shape from shading technique to estimate the face normal and then formulated the transfer problem as a 8-dimensional optimal mass transport (OMT), considering colors, pixel coordinates and face normals. Though their method produces excellent results, it takes a few minutes to process an HD image, since its formulation is conducted in 8-dimensional space. And such a high computational cost also diminishes its application to video color transfer. Here we apply OMT for sparse curves other than the whole raster images to efficiently transfer colors. Due to the information reuse property of DCs, this method is preferred especially for frequently transferring color from multiple references.

Fig. 1.1: Poisson vector graphics: the primitives and corresponding rendering results.

Furthermore, in the past decade, new techniques have been developed for 3D graphics modelling, decorating 3D meshes of any topological pattern remains an open problem in computer graphics and geometric modelling areas. As for enhancing the realism of 3D

models, except for the most popular technique — texture mapping, bump mapping is an alternative frequently used method. According to the textures’ local features, Sander et al. [6] reassigned the parametrization of the object to locally produce more details on the model. On the basis of the current viewpoint, Stamminger and Drettakis [7] used a perspective shadow map to convert the texture space that can produce a more realistic shadow effect near the camera. Dachsbacher and Stamminger [8] generated procedural terrain by warping geometric image. However, since these mapping based methods require local/global parameterization, it is usually difficult to produce a satisfactory result for 3D models of complex geometry and may causes visual distortion. A practical method is to use vector graphics to build color/displacement functions on 3D meshes directly. Jeschke et al. [9] presented to use diffusion curves [10] to produce carved pattern details on 3D models. For diffusion curves’ solution suffers from C^0 continuity, they are not able to build smooth features. Meditated by their work, we propose a PVG based 3D mesh decoration algorithm in order to synthesize smooth geometric details.

1.2 Motivation and Objective

Fully appreciating the good demonstration of PVG representation, we apply PVG into relevant fields to optimize existing procedures, such as presented in this thesis, displacement mapping and color transfer in images/videos.

First, inspired by the existing methods of decorating 3D models, we propose a 3D mesh decoration algorithm by leveraging PVG’s effective properties in order to generate smooth relief details. DCs based decoration algorithm suffers from C^{-1} or C^0 continuity, and DCs cannot allow intersection between each other as shown in Fig. 1.2. In comparison, as Poisson region is C^1 continuous and allows intersection, Poisson regions are sufficient to establish complex smooth details. And to render PVG on 3D meshes effectively, we extend PVG from planar domain to 3D mesh by building geodesic Voronoi tessellation on the models. Moreover, PVG is a simple good 2D graphic authoring tool. With sketching a few sparse curves, the user can easily design both smooth and discontinuous relief details on 3D models.

Then, we aim at proposing an easy yet effective approach for converting color between portrait images based on PVG vectorization. Fig. 1.3 illustrates the idea of PVG

Fig. 1.2: Intersection examples of DCs and PRs on 3D models. (a)(c) PVG primitives. (b)(d) Normal map of the rendering results. The first row is two PRs intersecting with each other. The second row shows two DCs intersecting lead to non-smooth artificial effect, since their relative boundary conditions are usually “competing” against each other.

guided color editing. The image of ice cream is vectorized into diffusion curves for low-frequency colors and Poisson regions for high-frequency details. To change the color, we simply modify the boundary colors of the diffusion curves and keep all Poisson regions unchanged. We apply the idea to transfer colors of portrait images. We first extract the face mask through a pre-trained neural network, and vectorize each source or reference images by utilizing several sparse DCs to encode the low-frequency colors, and then calculate PRs to encode the high-frequency details. Next, we transfer DCs’ boundary color between the source and reference images by applying the optimal mass transport. Finally, we use an efficient random-access Poisson solver to render the final PVG consisting of the color-changed DCs and the original or modified PRs. Compared with existing approaches that require 3D information or assume dense correspondences and alike poses between reference and source images, our method requires no pixel-to-pixel dense correspondence or 3D information to process portraits of different poses (including side views). Since our method conducts color transfer technique on only sparse curves (DCs), it also can transfer skin color of particular facial textures such as tattoo and freckle, and serves well for not only photographs but also paintings. In addition, our system provide support for PRs tuning to obtain special artistic styles like oil-painting or watercolor effects.

(a) Origin

(b) DCs & PRs

(c) Result

(d) DCs w/ new boundary colors

Fig. 1.3: Poisson vector graphics guided color editing. The ice cream image (a) is vectorized into a set of diffusion curves (colored, solid lines) and Poisson regions (dotted regions) (b), where the former encodes the low-frequency color and the latter encodes the high-frequency details. To change the ice cream’s color, we only need to edit the boundary colors of the diffusion curves (d). Diffusing the new boundary colors produces an image with a totally new looking (c). Intuitively speaking, we formulate the region-based color editing into a problem of editing the color of its boundary. The only difference between (b) and (d) is the boundary color of DCs.

The above has proposed a vectorization-based method that reduces the region color transfer problem to a boundary color transfer, hereby it is efficient. However, since their results are highly sensitive to the PDE’s boundary conditions, directly applying this method to video leads to unpleasing visual artifacts, such as flickering. To overcome the challenges of the existing works in the field of color transfer, we only convert the face color of the given reference image/video to the first frame of the input video by applying OMT between the boundary colors of DCs. Next, the DC boundary color of the first frame is transferred to the subsequent frames by matching the diffusion curves. Finally, with the original or modified PRs, we render the video with the Poisson solver. Thanks to our highly efficient DC matching algorithm, transferring colors for the vectorized video takes less than 1 millisecond per frame. Since we only apply OMT to the first frame, our method is computationally efficient and stable. For a single transfer from one reference image, the absolute performance of our algorithm, including pre-computation (i.e., vectorization) and post-computation (i.e., rendering) is comparable to the conventional raster image based method, which simply applies OMT to all frames. For frequent transfer from multiple reference images, our method runs one or two orders of magnitude faster than the image OMT method, due to its information reuse nature.

1.3 Contributions

Our main contributions of this thesis are concluded as follows:

1. We generalize Poisson solver from 2D domain to 3D models. It is a local solution to PVG rendering with more efficiency and flexibility. It allows the users to create displacement details on 3D models directly. Our 3D decorating system provides an easy and intuitive way to design geometric details on 3D meshes. PVG supplies the users with a simple set of curves to generate detailed relief patterns. In particular, comparing to the conventional DC that can only produce discontinues features, PR is superior for its smooth nature, which can support smooth 3D decoration and hierarchical level-of-detail editing on complicated models.
2. We propose an automatic vectorization based color transfer method for portrait image (video), which deduces time-consuming comparing to traditional method

of raster image, especially for scenes where the color of the given image (video) is transferred for multiple times. Because the information is full reused in our method.

3. Considering the continuity of video space, rather than processing each frame separately, we correspondingly match colors for subsequent frames. As a result, this strategy not only further improves the speed, but also ensures the anti-flicker of the video simply and effectively.

1.4 Thesis Organization

The remaining of this thesis is organized as follows. Chapter 2 introduces the preliminary knowledge of PVG and Chapter 3 reviews the relative literature. Chapter 4 describes 3D PVG solver for displacement mapping on 3D meshes. Chapter 5 presents PVG vectorization based color transfer for portrait images, and Chapter 6 extends to transfer face Color in Videos. Finally, Chapter 7 concludes the thesis and discusses the future directions.

Chapter 2

Preliminaries

We start by introducing a few concepts relevant to diffusion curve (DC), Poisson vector graphics (PVG) and harmonic B-spline.¹

For readability, we have listed the main symbols in Table 2.1 as below.

Item	Description
Ω	2D domain
$\mathcal{M}$	Input 3D triangle mesh
$U \subset \mathcal{M}$	A simply connected region on the mesh
$\mathbf{x}, \mathbf{y}$	Vertices of Ω or $\mathcal{M}$
$\mathbf{n}$	Normal vector
$\mathbf{t}_i$	The i -th knot of Ω or $\mathcal{M}$
$\bigcup_i \mathcal{V}_i$	Voronoi diagram
$N_1(\mathcal{V}_i)$	1-ring neighboring cells of $\mathcal{V}_i$
n	Number of knots (Voronoi cells)
$A_{\mathcal{M}}$	The total area of mesh $\mathcal{M}$
$A_{\mathcal{V}_i}$	The area of Voronoi cell $\mathcal{V}_i$
$A_{\mathbf{x}}$	The area of Voronoi region of vertex $\mathbf{x}$
e_{ij}	Voronoi edge between cells $\mathcal{V}_i$ and $\mathcal{V}_j$
d_{ij}	Geodesic distance between $\mathbf{t}_i$ and $\mathbf{t}_j$
f	Laplace constraints of PCs and PRs
g	The Dirichlet boundary condition
$u(\mathbf{x})$	The solution of Poisson equation
Δ	The Laplace-Beltrami operator
$\phi_{\mathbf{y}}(\mathbf{x})$	Green's function centered at vertex $\mathbf{y}$

Table 2.1: Notations

¹The work in this chapter has been published in [11].

2.1 Poisson Vector Graphics

Orzan et al. [10] proposed diffusion curves that are two-sided curve with colors defined on both sides. It can diffuse colors over the image to produce consistently shaded result except for sharp features along the boundaries (see Fig. 2.1(a)(b)). Mathematically speaking, diffusion curve image is the solution to Laplace equation

$$\Delta u(\mathbf{x}) = 0, \quad u(\mathbf{x})|_{\partial\Omega} = g, \quad (2.1)$$

where g is user-specified boundary color, and u represents the color of the whole region. Since the solution to Laplace equation is a harmonic function, by the maximum principle that a non-constant harmonic function can only remain a maximum (minimum) on the boundary of its domain, it implies that u cannot achieve local extrema. Therefore, DCs cannot provide enough degrees of freedom to control color gradient.

Poisson vector graphics [12] complements diffusion curves by appending two primitives — Poisson Curve (PC) and Poisson Region (PR). PC is to produce discontinuous colors across curves, whereas PR is to style smooth shading effect among the predefined regions. PVG solves the Poisson equation with piecewise constant function f , hereby taking DC as a special case with $f \equiv 0$. There are four unique advantages of extending the zero Laplacian to piecewise constant Laplacians f : (1) Users can simply manipulate the color minima and maxima not only on the curves (for DC/PC) but also inside the regions (for PR). (2) Users can explicitly manipulate geodesic shapes of the local/global shading by controlling f (which is a scalar for each color channel). (3) PVG natively supports seamless cloning. (4) PVG allows intersection among the geometric primitives. Even though PVG allows an arbitrary number of PRs/PCs, it requires at least one DC to serve as Dirichlet boundary condition g . Hou et al. [12] illustrated that PVG is an effective and easy-to-use graphic authoring tool that can create photo-realistic effect from sketches. A simple example of PVG is shown in Fig. 2.1(c)(d).

2.2 Harmonic B-splines

Let $\mathcal{T} = \{\mathbf{t}_i | \mathbf{t}_i \in \Omega\}_{i=1}^n$ a set of knots and $\Omega \subset \mathbb{R}^2$ be a 2D compact domain. Taking $\{\mathbf{t}_i\}$ as the generators, we build a Voronoi tessellation $\Omega = \bigcup_{i=1}^n \mathcal{V}_i$, where $\mathcal{V}_i$ is the Voronoi cell of knot $\mathbf{t}_i$.

Fig. 2.1: DC vs. PVG. PVG extends DC framework by appending two new primitives, PRs and PCs, to provide additional control on colors.

For any point $\mathbf{x}, \mathbf{y} \in \Omega$, Green's function of the Laplace operator Δ satisfies

$$\Delta \phi_{\mathbf{y}}(\mathbf{x}) = \delta_{\mathbf{y}}(\mathbf{x}), \quad (2.2)$$

where $\delta_{\mathbf{y}}(\mathbf{x})$ is the Dirac delta function centered at $\mathbf{y}$. In planar domain, an analytic solution of Eqn. (2.2) is

$$\phi_{\mathbf{y}}(\mathbf{x}) = \frac{1}{2\pi} \log(\|\mathbf{x} - \mathbf{y}\|).$$

For a Voronoi cell $\mathcal{V}_j$, using Green's theorem to (2.2) obtains

$$\int_{\mathcal{V}_j} \Delta \phi_{\mathbf{y}}(\mathbf{x}) \, d\sigma = \int_{\partial \mathcal{V}_j} \frac{\partial \Delta \phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} \, ds, \quad (2.3)$$

where $\mathbf{n}$ is the outward unit normal to the boundary $\partial \mathcal{V}_j$, $d\sigma$ and ds are the area and line integral elements, respectively.

Next, we define a function ψ_j for each Voronoi cell $\mathcal{V}_j$ as

$$\psi_j(\mathbf{x}) = \sum_i w_{ij} \phi_{\mathbf{t}_i}(\mathbf{x}), \quad (2.4)$$

where w_{ij} is the discrete Laplacian weight and $\sum_i w_{ij} \phi_{\mathbf{t}_i}(\mathbf{x})$ is a boundary sum that approximates the line integral on the right hand side of Eqn. (2.3).

Feng and Warren [13] demonstrated that the function ψ_j is *approximately* nonnegative, local, and satisfying partition of unity, inheriting many features of the basis functions of the popular B-splines, thus, they called it *harmonic* B-spline. There are two notable

properties to make it different from the conventional B-splines: (1) It is parameterization free, because calculating Green's function $\phi_{\mathbf{y}}(\mathbf{x})$ requires the distance $\|\mathbf{x} - \mathbf{y}\|$, which can be obtained by a coordinate-free measurement. (2) There is no restriction on knots, even completely irregular knots can construct splines effectively.

2.3 2D PVG Solver

To render PVG, we solve the Poisson equation below

$$\Delta u(\mathbf{x}) = f, \quad u(\mathbf{x})|_{\mathbf{x} \in \partial\Omega} = g, \quad (2.5)$$

where f is the Laplacian constraint of shading represented by PCs/PRs, and g is the Dirichlet boundary condition of colors assigned by DCs. Recently, Hou et al. [12] presented a closed-form solution to this equation on the basis of harmonic B-spline [13].

Green's third identity offers an analytical solution to Laplace/Poisson equation in integral form

$$\begin{aligned} u(\mathbf{x}) &= \iint_{\Omega} \phi_{\mathbf{y}}(\mathbf{x}) \Delta u(\mathbf{y}) d\sigma_{\mathbf{y}} \\ &+ \oint_{\partial\Omega} \left(u(\mathbf{y}) \frac{\partial \phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} - \phi_{\mathbf{y}}(\mathbf{x}) \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}} \right) dl_{\mathbf{y}}, \end{aligned} \quad (2.6)$$

where dl and $d\sigma$ and the line and surface elements, $\mathbf{n}$ is the outward pointing unit normal of dl .

Hou et al. [12] rasterized the image domain Ω according to the resolution specified by user and then discretized the PVG primitives with quad-tree structure. Taking the quad-tree nodes as generators, they constructed a Voronoi tessellation to partition the domain $\Omega = \bigcup_i \mathcal{V}_i$. For any Voronoi cell $\mathcal{V}_i$, they established a harmonic B-spline basis function $\psi_i(\mathbf{x})$, whose knots are the generators of $\mathcal{V}_i$ and its 1-ring neighboring cells. For basis function of a boundary Voronoi cell $\mathcal{V}_i$, they used the Dirichlet boundary condition g to determine its control point λ_i . For basis functions of internal Voronoi cells, they solved a sparse linear function of size $k \times k$ to calculate their control points, where k is the number of interior quad-tree nodes. Finally, the solution was obtained by $u(\mathbf{x}) = \sum_{i=1} \lambda_i \psi_i(\mathbf{x})$. Because harmonic B-spline basis functions are approximately local, evaluating $u(\mathbf{x})$ only contains the basis functions close to $\mathbf{x}$.

2.4 3D PVG Mesh Solver

Let $\mathcal{M}$ be a closed manifold mesh and $U \subset \mathcal{M}$ a simply connected region enclosing the PVG primitives. To render PVG on 3D models, we leverage the harmonic B-spline based solution similar to [12]. For a closed mesh $\mathcal{M}$, Green's function of the Laplace-Beltrami operator is

$$\Delta\phi_{\mathbf{y}}(\mathbf{x}) = \delta_{\mathbf{y}}(\mathbf{x}) - \frac{1}{A_{\mathcal{M}}}, \quad (2.7)$$

where $A_{\mathcal{M}}$ is the area of $\mathcal{M}$. The additional area term is necessary to guarantee the existence of a solution $\mathbf{y}(\mathbf{x})$ in the compact case. According to Green's third identity on closed manifold, the solution $u(\mathbf{x})$ can be written as

$$\begin{aligned} u(\mathbf{x}) = & \frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma + \iint_U \Phi_{\mathbf{y}}(\mathbf{x}) \Delta u(\mathbf{x}) d\sigma \\ & + \oint_{\partial U} \left(u(\mathbf{x}) \frac{\partial \Phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} - \Phi_{\mathbf{y}}(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial \mathbf{n}} \right) dl. \end{aligned} \quad (2.8)$$

In contrast to 2D domain case (Eqn. (2.6)), we add a term $\frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma$ that is the average of $u(\mathbf{x})$ in region U . Such a term reveals the difference of Green's function defined on open and closed domains.

We choose a set of knots $\{\mathbf{t}_j\}_{j=1}^n$ which is a subset of mesh vertices and construct a Voronoi tessellation on $\mathcal{M}$. The knot selection criteria are as follows: (1) the outer boundary of $\mathcal{V}$ coincides with the boundary of U . (2) the regions on the boundary (i.e., $\partial\mathcal{V}_j \cap \partial U \neq \emptyset$) are small enough. (3) the function f of each $\mathcal{V}_j$ is constant. Similar to [14], we discretize $u(\mathbf{x})$ as

$$\begin{aligned} u(\mathbf{x}) \approx & \frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma + \underbrace{\sum_{\mathcal{V}_j \in \mathcal{I}} A_{\mathcal{V}_j} f|_{\mathcal{V}_j} \bar{\phi}_{\mathcal{V}_j}(\mathbf{x})}_{\text{internal cells}} \\ & + \underbrace{\sum_{\mathcal{V}_j \in \mathcal{B}} \bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) \int_{\partial\mathcal{V}_j \setminus \partial U} \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}} dl_{\mathbf{y}}}_{\text{inner boundaries}} \\ & + \underbrace{\sum_{\mathcal{V}_j \in \mathcal{B}} u(\mathcal{V}_j) \int_{\partial\mathcal{V}_j \cap \partial U} \frac{\partial \phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} dl_{\mathbf{y}}}_{\text{outer boundaries}}, \end{aligned} \quad (2.9)$$

Feng and Warren [13] defined the basis function for Voronoi cell $\mathcal{V}_j$ as

$$\psi_j(\mathbf{x}) = \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x}))$$

where the coefficient $a_{ij} = -\frac{\|e_{ij}\|}{d_{ij}}$, and the Voronoi edges e_{ij} are divided into three disjoint categories in the form of geometry: interior edges $\mathcal{E}_{ie}$, outer boundaries $\mathcal{E}_{ob}$ and inner boundaries $\mathcal{E}_{ib}$.

Next, we can yield

$$\begin{aligned} & \sum_{\mathcal{V}_j \in \mathcal{I} \cup \mathcal{B}} \lambda_j \psi_j \\ &= \sum_{\mathcal{V}_j \in \mathcal{I} \cup \mathcal{B}} \lambda_j \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{e_{ij} \in \mathcal{E}_{ob}} a_{ij}(\bar{\phi}_{\mathcal{V}_i} - \bar{\phi}_{\mathcal{V}_j}) \lambda_j \\ & \quad + \sum_{e_{ij} \in \mathcal{E}_{ib} \cup \mathcal{E}_{ie}} \left(a_{ij}(\bar{\phi}_{\mathcal{V}_j} - \bar{\phi}_{\mathcal{V}_i}) \lambda_i + a_{ij}(\bar{\phi}_{\mathcal{V}_i} - \bar{\phi}_{\mathcal{V}_j}) \lambda_j \right) \\ &= \sum_{\mathcal{V}_j \in \mathcal{I} \cup \mathcal{B}} \lambda_j \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_j \in \mathcal{I}} \left(\bar{\phi}_{\mathcal{V}_j} \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\lambda_i - \lambda_j) \right) \\ & \quad + \sum_{\mathcal{V}_j \in \mathcal{B}} \left(\bar{\phi}_{\mathcal{V}_j} \sum_{e_{ij} \in \mathcal{E}_{ib}} a_{ij}(\lambda_i - \lambda_j) \right) + \sum_{\mathcal{V}_j \in \mathcal{B}} \left(\lambda_j \sum_{e_{ij} \in \mathcal{E}_{ob}} a_{ij}(\bar{\phi}_{\mathcal{V}_i} - \bar{\phi}_{\mathcal{V}_j}) \right). \end{aligned} \tag{2.10}$$

For λ_j of $\mathcal{V}_j \in \mathcal{B}$, we simply make $\lambda_j = g|_{\mathcal{V}_j}$. To calculate the coefficients of $\bar{\phi}_{\mathcal{V}_j}$ ($\mathcal{V}_j \in \mathcal{I}$) satisfying $A_{\mathcal{V}_j} f|_{\mathcal{V}_j}$, we solve the equation below

$$\mathbf{L}^{\mathcal{I}} \boldsymbol{\lambda}^{\mathcal{I}} = \mathbf{b} - \mathbf{L}^{\mathcal{B}} \boldsymbol{\lambda}^{\mathcal{B}}, \tag{2.11}$$

where $b_j = A_{\mathcal{V}_j} f|_{\mathcal{V}_j}$. Obvious, λ_j is the average color (displacement) in $\mathcal{V}_j$. It is easy to verify that the third and fourth terms of Eqn. (2.10) are identical to inner boundaries and outer boundaries of Eqn. (2.9). Since λ_j is the average color (displacement) of $\mathcal{V}_j$, $\sum_{\mathcal{V}_j \in \mathcal{I} \cup \mathcal{B}} \lambda_j \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}}$ is the average color (displacement) of region U . Therefore, Eqn. (2.10) is the solution to the above Poisson equation (2.5).

Since the basis function $\psi_j(\mathbf{x})$ decays to zero quickly, we evaluate the $u(\mathbf{x}) = \sum_{\mathcal{V}_j \in N(\mathbf{x})} \lambda_j \psi_j(\mathbf{x})$ using only the basis functions in a local neighborhood of $\mathbf{x}$.

Remark. Our solver can be extended to process open meshes. By removing the constant term and adding the boundary integral to Eqn. (2.8), we can rewrite the solution as

$$\begin{aligned} u(\mathbf{x}) = & \frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma + \iint_U \Phi_{\mathbf{y}}(\mathbf{x}) \Delta u(\mathbf{x}) d\sigma \\ & + \oint_{\partial U} \left(u(\mathbf{x}) \frac{\partial \Phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} - \Phi_{\mathbf{y}}(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial \mathbf{n}} \right) dl. \end{aligned} \quad (2.12)$$

The basis function for Voronoi cell $\mathcal{V}_j$ is

$$\psi_j(\mathbf{x}) = \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij} (\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x})),$$

thus, we obtain the solution $u(\mathbf{x}) = \sum_{\mathcal{V}_j \in \mathcal{N}(\mathbf{x})} \lambda_j \psi_j(\mathbf{x})$ as a similar form as the case of closed meshes.

Chapter 3

Related Work

We present an overview of literature that related to the topic of this thesis in this chapter.

3.1 Vector Graphics

Comparing to raster images, vector graphics provide many advantages such as compact, arbitrary resolution and editable.

In 2008, Orzan et al. [10] proposed diffusion curve that is two-sided curve with colors defined on both sides. It can diffuse colors over the image to produce consistently shaded result except for sharp features along the boundaries. The maximum principle claims that a non-constant harmonic function can only achieve minima/maxima on the boundary of its domain rather than anywhere in the interior region, though, the solution to diffusion curves is based on harmonic functions. Therefore, the upward research on this topic focuses on increasing degrees of freedom to control color gradient. Bezerra et al. [15] developed diffusion barrier, diffusion anisotropy, and spatially changing color intensity to impact the diffusing procedure. Their method can generate great non-photorealistic results by diffusing colors and normals. Nonetheless, normal maps cannot be diffused with open diffusion curves, and it is non-straightforward to assign the normals' boundary condition. Finch et al. [16] provided smooth interpolation by extending diffusion curves through utilizing thin-plate splines. The thin-plate splines can simulate smooth effect, whereas it may produce unpredicted local results since it violates the maximal principle of harmonic functions. Furthermore, compared with solving a Laplace's equation, solving the bi-Laplace's equation increases the computational complexity. Lieng et

al. [17] presented shading curves by associating shading profiles to diffusion curves' both sides. The shading profile can express the color gradient out from its associated curve, and it allows manual manipulation. Recently, Hou et al. [18] proposed Poisson vector graphics by extending diffusion curves with non-zero Laplacians. It appends two new primitives called Poisson curves and Poisson regions, which can generate photo-realistic images effectively.

3.2 Image Vectorization

To vectorize raster images efficiently, some such approaches have appealed already.

Price and Barrett [19] presented a subdivision-based vectorization method, which hierarchically subdivides objects in an image with a recursive graph cut when the error in this patch is above threshold. Each patch is represented as a Bezier patch, so makes this vectorization editable. But this vectorization method may bring many tiny patches on the highlight regions. On the basis of a two-level variational parametric segmentation algorithm, Lecot and Levy [20] proposed a vectorization approach by minimizing Mumford and Shah's energy function. It operates on the intermediate triangulation, since this structure is very suitable for the calculation considering the characteristics of the images. But in general, the segmentations in the vector result is too dense to edit. However, the vector units are difficult to edit, owing to the density of segmentations.

Sun et al. [21] developed an optimized gradient mesh consisting of Ferguson patches [22] whose four boundaries are represented with Bezier splines. By minimizing reconstruction error, they could obtain a simpler gradient mesh structure than the result generated from commercial softwares such as Adobe "Live Trace". However, the vectorization process needed user to assign some lines to guide the mesh generation. Based on slit map [23], Lai et al. [24] automatically converted a raster image to a high-quality gradient mesh of non-trivial topology by applying holomorphic 1-form. Gradient mesh provides a hierarchical expression for vectorizing object-based images accurately. But it cannot deliver satisfying results for images with chaos backgrounds.

With the development of artificial intelligence techniques, Song et al. [25] developed an image compression approach by using a deep neural network. Their representation

supports GPU texture mapping and compactly approximate any point sampling. Later, Dekel et al. [26] proposed a convolutional neural network based vectorization algorithm for portrait images with the input of sparse diffusion curves. Their network was cascading two U-Net networks — one for low frequency reconstruction and another for high frequency reconstruction. Since they stored colors and details in the sparse curves, they can edit the geometry of the faces by selecting and dragging these feature curves.

Diffusion curve vectorization aims at converting a raster image into a set of curve based vector primitives. Orazan et al. [10] extracted diffusion curves using Canny edges and fitted their boundary colors by a least square method. Xie et al. [27] developed an automatic method for vectorizing raster images with hierarchical diffusion curves. Their method is efficient, accurate and robust, working for both art and natural images. However, it often extracts many curves, making post-editing difficult.

The proposed PVG vectorization only takes the salient feature curves as diffusion curves, which present the low-frequency skin colors only. Furthermore, our vectorization method designed for videos is also efficient, since we only need to vectorize the first frame of the input video. The boundary colors of diffusion curves for the subsequent frames are obtained by a simple DC matching algorithm.

3.3 Color Transfer

There exists an extensive literature on the topic of color transfer since 2000.

Reinhard et al. [4] pioneered the color transfer method that matches the mean and standard deviation of color distributions of the source and reference images. Pitie et al. [28] proposed a N -dimensional probability density function that can match two distribution in an exact manner. Hwang et al. [5] applied probabilistic moving least square for color transfer between images with the same scene but different camera settings and illumination conditions. To decrease the artifacts of the transferring results, Lee et al. [29] proposed an example-based color conversion method by implementing tone mapping, color regularization and illumination adjustment in combination. Shih et al. [2] presented a good strategy of portraits' style transferring that matched the global lighting and the local contrast by transferring the local statistics. This technique can generate

reasonable results, but it involves complex pre-processing to calculate a dense correspondence between the source and the reference images.

With respect to the subject of portrait color editing, there is also a large body of work. Wu et al. [30] first implemented facial feature segmentation to extract semantic regions for portraits, and then respectively converted colors between these corresponding semantic regions. Yang et al. [31] transferred colors for portraits semantically. It requires manual facial region extraction and transfers colors between the corresponding regions using [4]. Li et al. [32] applied a melanin and hemoglobin based model to represent the diffuse color variations in face skin and remove specular highlights. Zhou et al. [33] developed a CNN model to regress lighting parameters from a single face image. Sun et al. [34] trained a novel neural network that can predict the illumination from a single image and hereby be used for relighting. Shu et al. [1] developed a relighting and color transfer method for headshots by solving OMT for 8D data including rgb colors, pixel positions, and normals. There are also works that apply OMT for raster image color transfer [35,36].

In this thesis, we conduct OMT for sparse curves other than the whole raster images to efficiently transfer colors. Due to the information reuse property of DCs, this method is preferred especially for frequently transferring color from multiple references.

3.4 Style Transfer

Style transfer is also a closely related problem. Recent years have witnessed great success of deep learning techniques.

Style transfer like [37,38] is usually applied to photons, sketches and paintings. Even between photos, it is easy to cause distortion, resulting in the effect of painting. Then, Luan et al. [39] improved the neural style method by adopting an energy equation, that obtained a local affine transfer to produce the photo conversion effect rather than like painting. Liao et al. [40] presented that the transfer between human face images obtained by [39] is not so ideal, and their method required a dense correspondence between the input images with similar scenes, otherwise it may lead to failure case. Similarly, He et al. [41] realized color transfer between images with perceptually similar semantic structures

by matching semantically-meaningful dense correspondence and made object-to-object color transferring results more precise. Li et al. [42] proposed a photorealistic image stylization by first adopting a neural style transform algorithm and then smoothing the result where existed inconsistent stylization in semantically similar regions based on a manifold ranking algorithm. However, these methods either focus on non-photorealistic style or require a dense correspondence map between the source and the reference. To our knowledge, there is no deep learning approaches designed for face color transfer in videos.

Generative adversarial network (GAN) [43] is a powerful generative model. A GAN trained on portrait photographs can generate new photographs that look authentic to human observers, having many realistic characteristics [44–46]. Karras et. al. [47] proposed a style-based generator that uses face color as a high-level attribute and transfers it to target images. StarGAN [48] and AttnGAN [49] allow facial attribute editing. The GAN approaches do not require semantic segmentation and can generate realistic images. However, training a GAN is difficult due to large amount of training data and some model-related issues, such as diminished gradient, non-convergence and mode collapse. Though some approaches, such as CycleGAN [50, 51], can be used for transfer color between images by taking color as an attribute, it is unclear whether they can be used for videos.

Our method, built upon PVG vectorization and optimal mass transport, is conceptually simple and easy to implement.

3.5 Surface Decoration

Besides the most popular technique — texture mapping, there exists other alternatives to decorate 3D models.

Bump mapping based approaches such as [52–54] perturb normals of 3D model to simulate shading effect, thus, its geometry remain the same as before. Comparing with bump mapping which can only change normal maps, parallax mapping [55, 56] can control the texture coordinates to achieve mesostructured normals along its viewing direction and approximately change colors based on its depth information. Relief texture mapping

[57, 58] leverages depth values to enhance the texture realism, which can produce visual detailed geometries on flat polygons and generate pixel shader by locating the intersection between the ray and the height field. Comparing with the above methods, displacement mapping [59–61] actually changes geometries of the models by lifting vertex’s location along its normal direction.

Furthermore, by combining displacement mapping and texture mapping, the geometric texture mapping methods [62, 63] can synthesize complicated geometric details on the surface of an arbitrary model. Bhat et al. [64] provided an example-based method for volumetric models to produce geometric textures. Their method can obtain some certain effects like grooves and pits by generalizing the neighbourhood-based texture composition algorithm. By using local stitching and deformation, Zhou et al. [65] presented a mesh quilting method to produce a geometric texture on a thin shell, which seamlessly wrapped around the original model. De Toledo et al. [66] used a group of geometry textures to represent highly subdivided meshes by reconstructing fine-scale detailed geometries on a simple proxy of the source mesh. Taking a user-specified lateral line as input to control the orientation and distribution, Landreneau and Schaefer [67] provided an effective approach to generate scale pattern or scale-like structure repetitively on polygon mesh objects.

In this thesis, we generalize PVG from 2D domain to 3D meshes that leads to the user’s artistic decorating directly onto models. Our PVG guided decoration approach can easily and intuitively produce smooth relief geometric details on meshes.

Chapter 4

Decorating 3D Models with Poisson Vector Graphics

This chapter addresses a novel approach of decorating 3D surfaces to composite smooth relief geometries. Since Poisson Vector Graphics (PVG) can produce an artistic work with sparse curves, the user can directly edit synthetic geometric details and colors on the models. Note that DC/PC is useful to generate discontinuous or sharp results, moreover, PR can create smooth details. PR is superior for its smooth nature, which can support hierarchical level-of-detail editing on the models. By comparing our method with other commonly-used existing technologies, we find that their methods generally depend on global/local parameterization. Our method supports parameterization-free decoration for any model, meanwhile diminishing the overall computational cost. So this is an ideal method to generate smooth relief details. In order to render PVG effectively on models, we generalize the Poisson solver on 2D domain to 3D meshes. The 3D Poisson solver is based on harmonic B-spline curves constructed with the help of geodesic Voronoi tessellation. It is a local solver that is designed to flexibly and versatily render PVG. Our results manifest effectiveness of the proposed approach on both real-world and synthetic models.¹

4.1 Background

Even though in the past decade, new techniques has been developed for 3D graphics modelling, decorating 3D meshes of any topological pattern remains an open problem

¹The work in this chapter has been published in [11].

in computer graphics and geometric modelling areas. As for enhancing the realism of 3D models, except for the most popular technique — texture mapping, bump mapping like [6–8, 68] is an alternative frequently used method. However, since these mapping based methods require local/global parameterization, it is usually difficult to produce a satisfactory result for 3D models of complex geometry and may causes visual distortion.

A practical method is to use vector graphics to build color/displacement functions on 3D meshes directly. Jeschke et al. [9] presented to use diffusion curves [10] to produce carved pattern details on 3D models. For diffusion curves' solution suffers from C^0 continuity, they are not able to build smooth features.

Strongly motivated by the existing methods of decorating 3D models, we propose a 3D mesh decoration algorithm by leveraging PVG's effective properties in order to generate smooth relief details. PVG [12] naturally generalize the popular Diffusion Curves (DC) by appending two primitives — Poisson Region (PR) and Poisson Curve (PC). It is an effective 2D graphic designing tool which extends Laplace's equation to Poisson's equation. In this chapter, we extend PVG from 2D domain to 3D mesh decorating. DC based decorating approaches suffer from C^{-1}/C^0 continuity. In comparison, as PR is C^1 continuous, PRs are sufficient to establish smooth shading effect.

To render vector graphics, Orzan et al. [10] developed a multi-grid based solution to utilize a fine version of the domain as the high-frequency component and a coarse version to obtain the low-frequency component. Jeschke et al. [69] proposed finite difference with varied step length to speed up its convergence process. On the basis of bi-Laplacian solver, Boye et al. [70] proposed a finite element method that is able to handle Laplacian constraints directly. It is worth mentioning that, only faces involved in the decorating area require further processing, so a *global* solver is unnecessary for mesh rendering. In the light of 2D PVG solver, we develop a *local* Poisson solver to efficiently render displacement functions on 3D meshes. Instead of processing global details, it works on only the vertices contributing to the decoration region.

4.2 Algorithm

4.2.1 Overview

PVG contains three different primitives: DC, PC and PR. DC is a two-sided curve with colors/displacements defined on both sides that diffuse colors over the image to produce consistent color except for sharp features along the boundaries. And as the boundary condition, it ensures a unique solution of Poisson equation. PC is a two-sided curve with reverse Laplacians to produce hard shading along the curve. PR is a region with opposite Laplacians defined on inner region and outer boundary to create a controllable soft shading within the region. In brief, DC assigns colors or heights as boundary condition, PC and PR control shading effect by tuning Laplacians. In particular, PR can effectively create smooth features such as highlights and smooth relief details.

In this chapter, we use PVG to create colors and displacements on meshes. To distinguish the different functions, we add prefixes “c” and “d” to the primitives. For instance, dPR is displacements produced by PR, and cDC is colors assigned by DC. Fig. 4.1 gives a simple example that transform a cube model to a flag model with a simple set of PVG primitives. Moreover, Fig. 4.1(e)(g) illustrates that PRs make a smoother decoration than conventional DCs.

Fig. 4.2 illustrates a detailed comparison between DC and PR. DC is assigned Dirichlet boundary condition g , whereas PR is associated with Laplacians g . To calculate the displacement (color) field, We solve the Poisson function as below

$$\Delta u = g, u|_{\partial\Omega} = f.$$

DC can create smooth displacement inside the domain Ω , but is not continuous across the boundary $\partial\Omega$. And the only exception is $f \equiv 0$, which obtains a constant function $u \equiv 0$ (see Fig. 4.2(a)(b)). Associating with DCs and PRs, we can represent displacement of C^1 continuity both along the boundary $\partial\Omega$ and inside the domain Ω (see Fig. 4.2(c)(d)). In short, DC can only create sharp features, however, combining DC and PR together is able to produce smooth reliefs.

In addition, owing to the analytic solution of Green’s function of 2D Laplace operator, PR remains C^1 -continuous along its boundary in planar domain (see Fig. 4.3). However,

Fig. 4.1: Transforming a cube model to a flag model with a simple set of PVG primitives involving 5 cDCs, 4 dDCs and 1 dPR. We use color map to visualize the displacement constraints.

Fig. 4.2: DC vs. PR. We visualize f and g with the same color map as above, i.e., color and warm colors indicate negative and positive values respectively, green is zero.

Fig. 4.3: On planar domain, thanks to the analytic solution of Green's function, PR (dashed curve) remains C^1 continuous along the boundary.

for general 3D models, PR cannot hold C^1 -continuity any more. Still, PR is more flexible to use in contrast to DCs. Because DCs cannot allow intersection between each other, whereas PRs can (see Fig. 1.2). It is a desired feature for 3D graphic authoring.

Fig. 4.4 illustrates the pipeline of our 3D mesh decorating algorithm: (a) shows the domain U (since dDCs are open, $U = \mathcal{M}$) and the displacement constraints including 1 dPR (red star) and 4 dDCs (another two on the bottom) along edges of the cube model, the dPR specifies Laplacians f and dDCs determine Dirichlet boundary condition g . (b) First, to satisfy that the Laplacians in any Voronoi cell is constant, sample n knots including all the pivotal vertices in dPR/dDCs and some other vertices evenly distributed with a given knot density. Next, take these boundary and interior knots as generators to establish geodesic Voronoi tessellation, and U is segmented into a set of disjoint sub-regions, $U = \bigcup_{i=1}^n \mathcal{V}_i$. (c) According to the Green's function of the Laplacian operator on $\mathcal{M}$, for a Voronoi cell $\mathcal{V}_j$, define a harmonic B-spline basis function $\psi_j(\mathbf{x})$, whose knot $\mathbf{t}_j$ is the generator of $\mathcal{V}_j$. (d) If $\mathcal{V}_j$ is a boundary Voronoi cell, simply assign its control point λ_j with the given boundary condition g . If $\mathcal{V}_j$ is an inner Voronoi cell, calculate its control point λ_j by solving a sparse linear system, whose size is much smaller than the number of mesh's vertices. (e) After calculating basis functions and control coefficients, establish harmonic B-spline function $u(\mathbf{x}) = \sum_{i=1}^n \lambda_i \psi_i(\mathbf{x})$ to obtain the solution of Poisson function. As shown in (c) that the basis functions decay to zero quickly, we can evaluate the $u(\mathbf{x}) = \sum_{\mathcal{V}_i \in \mathcal{N}(\mathbf{x})} \lambda_i \psi_i(\mathbf{x})$ only with the basis in a neighborhood around $\mathbf{x}$.

Taking the user-sketched curves as input, we take a subset of mesh vertices as knots and then establish a geodesic Voronoi tessellation. Next, for each Voronoi cell, we build a harmonic B-spline basis function ψ . By solving a small sparse linear system, we achieve the control coefficients λ and finally present the color/displacement function as $\sum_i \lambda_i \psi_i$.

4.2.2 Discrete Green's Function

Define $g_{ij} = \phi_{\mathbf{v}_i}(\mathbf{v}_j)$ the Green's function centered at vertex $\mathbf{v}_i$ evaluating at vertex $\mathbf{v}_j$. We denote by $\mathbf{G} = (g_{ij})_{m \times m}$ the discrete Green's function on closed mesh M , where m is the number of vertices. Lipman et al. [71] demonstrated that $\mathbf{A}^{-1} \mathbf{L} \mathbf{G} = \mathbf{I} - \frac{1}{n} \mathbf{1} \mathbf{1}^T$, where $\mathbf{1}$ is the column vector of all ones. Denote by $\mathbf{M}_j$ the j -th column of matrix $\mathbf{M}$. They showed that $\mathbf{G}_j = \mathbf{x} - \frac{\mathbf{1}^T \mathbf{x}}{\mathbf{1}^T \mathbf{1}} \mathbf{1}$, where $\mathbf{x}$ is a particular solution to $\mathbf{L} \mathbf{A}^{-1} \mathbf{L} \mathbf{x} = (\mathbf{I} - \frac{1}{n} \mathbf{1} \mathbf{1}^T)_j$.

Fig. 4.4: Illustration of our Poisson solver on a cube mesh $\mathcal{M}$. (a) Displacement constraints. (b) Knots $\{\mathbf{t}_i\}$ & Voronoi diagram $\{\mathcal{V}_i\}$. (c) Basis function $\psi_j(\mathbf{x})$. (d) Control points $\{\lambda_i\}$. (e) Harmonic B-spline function $u(\mathbf{x}) = \sum_{i=1}^n \lambda_i \psi_i(\mathbf{x})$.

Algorithm 1 Decorating 3D meshes with PVG

Require: The triangle mesh $\mathcal{M}$, the PVG primitives (i.e., DCs and PRs) associated with color and/or displacement constraints

Ensure: The mesh with colors and reliefs decorated by PVG

- 1: Calculate discrete Green's function $\phi_{\mathbf{y}}(\mathbf{x})$ by harmonic form of [71] Eqn. (10)
 - 2: Select knots $\{\mathbf{t}_i\}_{i=1}^n$ from vertices of the mesh
 - 3: Build geodesic Voronoi tessellation on $\mathcal{M}$, and $\mathcal{M}$ is divided into a set of sub-regions $\mathcal{M} = \bigcup_{i=1}^n \mathcal{V}_i$
 - 4: // Construct the harmonic B-spline basis function $\{\psi_i\}_{i=1}^n$
 - 5: **for** each Voronoi cell $\mathcal{V}_j$ **do**
 - 6: $\psi_j(\mathbf{x}) = \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x}))$
 - 7: $\bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) = \frac{1}{A_{\mathcal{V}_j}} \sum_{\mathbf{y} \in \mathcal{V}_j} A_{\mathbf{y}} \phi_{\mathbf{y}}(\mathbf{x})$
 - 8: **end for**
 - 9: // Compute the control points $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1}^n$
 - 10: **for** each Voronoi cell $\mathcal{V}_i$ **do**
 - 11: Construct i -th row of sparse matrix $\mathbf{L} = \{a_{ij}\}_{i,j=1}^n$
 - 12: **for** every neighbor $\mathcal{V}_j \in N_1(\mathcal{V}_i)$ **do**
 - 13: $a_{ij} = -\frac{\|e_{ij}\|}{d_{ij}}$
 - 14: **end for**
 - 15: $a_{ii} = \sum_{\mathcal{V}_j \in N_1(\mathcal{V}_i)} a_{ij}$
 - 16: $b_i = A_{\mathcal{V}_i} f|_{\mathcal{V}_i}$
 - 17: **end for**
 - 18: Solve $\mathbf{L}\boldsymbol{\lambda} = \mathbf{b}$
 - 19: // Render the decorated mesh
 - 20: **for** each vertex $\mathbf{x} \in \mathcal{V}_i$ **do**
 - 21: $u(\mathbf{x}) = \sum_{\mathcal{V}_j \in N(\mathbf{x})} \lambda_j \psi_j(\mathbf{x})$
 - 22: **end for**
-

To get the particular solution, replace the first row and the first column of $\mathbf{A}^{-1}\mathbf{L}$ by zeros and set the diagonal entry at their intersection to 1. Also replace the first row of $(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)_j$ by 0. As mentioned in [71], this kind of linear system can be solved very efficiently by first performing Cholesky factorization of $\mathbf{L}\mathbf{A}^{-1}\mathbf{L}$ and then performing two backward substitutions for every given vector on the right hand side. Green's functions on open mesh can be similarly solved by $\mathbf{A}^{-1}\mathbf{L}\mathbf{G} = \mathbf{I}$.

4.2.3 Geodesic Voronoi Tessellation

When user draw DCs and PRs' boundaries on the surface of 3D model, each curve assigned with two-side color (height) or Laplacian are represented by a B-spline curve. According to the work in [72], De Casteljau's algorithm are extended into 2D manifold which replaced the linear interpolation with the smooth geodesic interpolation. Based on this framework, we represent PVG primitives by B-spline curves on 3D meshes. Actually, there requires massive calculation of pairwise geodesic path. Hereby we apply an effective method [73] to query approximate geodesic path between any two points on meshes.

According to user-sketched curves (DCs/PRs), we assign the vertices across these curves with the given Laplacian or color constraints, and take the vertices as a subset of knots. To meet the requirement that Laplacian in each Voronoi cell is constant, we additionally select some extra vertices as knots, that are evenly distributed on the surface with a given density. The selected knots are then used as generators to build geometric Voronoi tessellation on the mesh.

Voronoi tessellation in $\mathbb{R}^2$ can be well constructed by the existing algorithms, e.g., dual Delaunay triangulation. However, in $\mathbb{R}^3$, it is more complex and one of the biggest differences is that the bisector is a point trace on the mesh which have equal geodesic distance to two knot neighbors instead of a straight line. First, by using the improved CH algorithm [74], we establish multi-generator exact geodesic distance field on the mesh with taking the selected knots as generators. Then, we assign each vertex a geodesic distance to its closest knot. After that, we mark the triangle edges whose endpoints have different closest knots. And when the incident triangle face of a marked edge contains three marked edges, it includes a branch point inside, otherwise it is crossed through by a segment of bisector. Finally, by using the marching algorithm [75], we construct the

geodesic Voronoi tessellation (Fig. 4.4(b)) on the mesh based on the marked edges and faces.

4.2.4 Control Coefficients

Associated with the established geodesic Voronoi tessellation and the color/displacement constraints defined on the vertices, we calculate the control coefficients λ of harmonic B-spline function by solving a sparse linear system as follows

$$\mathbf{L}\lambda = \mathbf{b} \quad (4.1)$$

where $\mathbf{L} = \{a_{ij}\}$ is a sparse symmetric matrix of size $n \times n$ (n is the number of knots $\{\mathbf{t}_i\}$), $\mathbf{b}$ is a n -dimensional vector defined as each knot's area multiplying its Laplacian constraint. According to Voronoi tessellation, we can actually obtain them by

$$a_{ij} = \begin{cases} -\frac{\|e_{ij}\|}{d_{ij}}, & \text{if } i \neq j \text{ and } \{t_i, t_j\} \text{ are neighbors} \\ 0, & \text{if } i \neq j \text{ and } \{t_i, t_j\} \text{ are not neighbors} \\ \sum_{\mathcal{V}_j \in N_1(\mathcal{V}_i)} a_{ij}, & \text{if } i = j; i, j \in [1, n] \end{cases} \quad (4.2)$$

$$b_i = A_{\mathcal{V}_i} f|_{\mathcal{V}_i}, i \in [1, n] \quad (4.3)$$

here a_{ij} is only related to the i -th knot $\mathbf{t}_i$ and its 1-ring Voronoi neighbourhood.

Note that each knot has six neighbors on average (because the average number of adjacent Voronoi cells is six), thus, there are generally seven non-zero values in each row of $\mathbf{L}$. In other words, the average color (displacement) of each Voronoi cell (see Fig. 4.4(d)) will serve as the control coefficients of harmonic B-spline function. In addition, since $\text{rank}(\mathbf{L}) = (n - 1)$, the linear system Eqn. (4.1) has infinite solutions. Here we use Dirichlet boundary condition to obtain a unique solution as Eqn. (2.11). That is, PVG allows any number of its primitives but at least one DC to ensure a unique solution of Poisson equation.

4.2.5 Basis Functions

We aim at supplying an effective decoration approach for 3D meshes to control both colors and geometric details of the vertices' features. In order to enable the Poisson solver on

all vertices of the model, obviously, we use a denser mesh (the same shape tessellated with more vertices) that we can obtain a higher resolution result. In other words, to produce good rendering results that satisfy the resolution requirements, we require the input mesh has sufficient vertex density. Otherwise, in our system, we automatically increase the input model's density by using midpoint subdivision algorithm.

As a result, the fundamental problem of our PVG solver is to reduce the computational expense, which is significant in the calculation process, especially for the high-resolution meshes we require. Harmonic B-spline function provides the local computation for solving the Poisson equation on the mesh instead of a global solution covering all vertices as the conventional finite element method does. Hereby, based on harmonic B-spline function, we develop a local solver of the Poisson equation on dense meshes.

With the obtained control coefficients, we can construct harmonic B-spline function on the mesh $\mathcal{M}$. Based on Green's third identity theory and previous derivation, we can define the j -th basis function of harmonic B-spline on the open boundary mesh by

$$\psi_j(\mathbf{x}) = \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x})), j \in [1, n] \quad (4.4)$$

and which is on the closed mesh:

$$\psi_j(\mathbf{x}) = \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x})), j \in [1, n] \quad (4.5)$$

where $\mathcal{V}_i$ is 1-ring neighbour of $\mathcal{V}_j$, $A_{\mathcal{M}}$ is the total area of the mesh, $A_{\mathcal{V}_j}$ is the area of j -th Voronoi cell $\mathcal{V}_j$, and is the average value of Green's function ϕ in Voronoi cell $\mathcal{V}_j$:

$$\bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) = \frac{1}{A_{\mathcal{V}_j}} \int_{\mathcal{V}_j} \phi(\mathbf{x}, \mathbf{y}) d\delta_{\mathbf{y}} \quad (4.6)$$

whose discrete form can be written as

$$\bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) = \frac{1}{A_{\mathcal{V}_j}} \sum_{\mathbf{y} \in \mathcal{V}_j} A_{\mathbf{y}} \phi_{\mathbf{y}}(\mathbf{x}) \quad (4.7)$$

Here $\phi_{\mathbf{y}}(\mathbf{x})$ is the Green's function on the mesh, that can be calculated by the approach proposed in [71]. Green's function on a specific mesh only requires one-time calculation, which is obtained in the preprocessing step in our system.

Next, we can derive the harmonic B-spline function as below:

$$u(\mathbf{x}) = \sum_{j=1}^n \lambda_j \psi_j(\mathbf{x}) \quad (4.8)$$

where $u(\mathbf{x})$ is each vertex's color (displacement), except these $\mathbf{x}$ belonging to the knot group that $u(\mathbf{x})$ are already known (assigned by DCs) or obtained (by Eqn. (4.1)), the other $u(\mathbf{x})$ remains unknown and requires further calculation. Note that λ_j is the control coefficient obtained by Eqn. (4.1).

Since the basis function $\psi(\mathbf{x}, \mathbf{y})$ decays to zero quickly, we can evaluate Eqn. (4.8) by only using the basis of a neighbourhood around $\mathbf{x}$:

$$u(\mathbf{x}) = \sum_{\mathbf{v}_j \in N(\mathbf{x})} \lambda_j \psi_j(\mathbf{x}) \quad (4.9)$$

After solving this harmonic B-spline function Eqn. (4.9), each vertex's color (displacement) can be attained eventually (Fig. 4.4(e)). Furthermore, thanks to the random-access solver, we can flexibly and explicitly determine the computation region, hereby reducing the computational expense. For example, adopting the visibility algorithms like Z-buffer [76] or back-face culling [77], we can detect if a point $\mathbf{x}$ is visible. If invisible, there requires no computation for $u(\mathbf{x})$ at all.

4.3 Results

We implemented our algorithm in C++ and evaluated it on a laptop with an Intel i7 CPU 2.80GHz. Our method allows user to decorate 3D meshes in three ways: (1) user sketches PVG primitives (DCs/PCs/PRs) directly on the surface, and then tunes the Dirichlet boundary condition g and Laplacians of color (displacement) f ; (2) user attaches a predefined 2D PVG to the target region on the 3D model using *local* parameterization; and (3) user tessellates a predefined 2D PVG to the entire 3D model using global parameterization.

Fig. 4.5 shows the result of decorating the Bunny model with three letters "SPM" and a set of colorful patterns. PVG based mesh decorating tool is easy to use, since users can directly sketch arbitrary curves on 3D mesh and then tune the colors (displacements)

by assigning the associated functions g and f . Moreover, as the input covering a small region (the computation region) of the mesh, our Poisson solver can just process the computation region in a local manner, that greatly reduce computational expense in actual situations. As shown in Fig. 4.5(d), according to the input, we first obtain a connected region that requires further process. After constructing a geodesic Voronoi tessellation with the n selected knots from that field, we can compute n control points of harmonic B-spline function by using equation $\mathbf{L}\boldsymbol{\lambda} = \mathbf{b}$. Nonetheless, only vertices whose Voronoi cells involving input field (the green regions) will be actually computed their $u(\mathbf{x})$.

Fig. 4.5: Decorating Bunny with three letters “SPM” and colorful patterns. With a set of DCs and PRs, we produce smooth and colorful reliefs. Owing to the *local* property of our solver, we can only build harmonic B-spline function and geodesic Voronoi tessellation in the computation regions as (d).

Fig. 4.6 is to decorate a stele model with convex patterns of dragon/auspicious clouds and concave Chinese calligraphy characters. In the decorating procedure, we first

produce an overall concave effect of the middle plaque with 2 rectangular dDCs. Second, we slightly lift the entire shape of the dragon also with 2 dDCs. Third, we use dPRs to create smooth convex effect for the auspicious clouds and the middle part of the dragon. Finally, we use more dPRs to generate smooth concave effect for the Chinese calligraphy characters. By flexibly applying different types of PVG, we can create a variety of interesting effects in both displacement and color aspects.

Fig. 4.6: Decorating a stele model with convex patterns of dragon/auspicious clouds and concave Chinese calligraphy characters. By simply tuning Laplacians, we can control the height of the convex/concave reliefs. Comparing the reliefs produced by dDCs with those created by dPRs (i.e. the convex patterns of dragon/auspicious clouds and the concave Chinese calligraphy characters), it is obvious that our PRs can generate smooth features while DCs can only construct discontinues effects.

Fig. 4.7 shows the results of decorating two different vase models with smooth reliefs generated by dPRs and constant colors by cDCs. In the RGB field, we first decorate the vase with blue and white porcelain patterns (red and green painted patterns) by using cDCs. Then, we create smooth convex reliefs for all the petals (flowers and vines) on the

surface. From this example, we can see that our approach is still available and powerful even for an exquisite and delicate PVG input.

Fig. 4.7: Decorating two vase models with smooth reliefs produced by dPRs.

Considering that sometimes massive repeated patterns are required for mesh decoration, our system allows user to attach predefined 2D PVG on the mesh by using global parametrization (see Fig.4.8), which can highly simplify user's sketching process. Besides, combining sketching with global parametrization, we can create various interesting decoration effects (see Fig. 4.9).

In our current implementation, the bottleneck is the preprocess step, involving calculating the Green's function and establishing the geodesic Voronoi tessellation. For the

Fig. 4.8: Decorating Sofa and Rhino with repeated reliefs.

Fig. 4.9: Constructing a cactus from a sphere. We define PVG on two separate layers, where the first layer models the stems and the second layer the sharp spines. Since PR is C^1 continuous, the two layers can be seamlessly combined together. Note that it is hard to apply only DCs for level-of-detail modeling, since it is discontinuous along the curves.

Piggy Bank model with roughly 40 PVG primitives, preprocess takes 138.4s (see Fig. 4.10). Local editing/updating PVG is much more efficient than re-computing for the entire model, since Voronoi diagram can be updated locally and we only requires to re-compute the basis functions for the affected Voronoi cells. Fig. 4.10 shows that changing colors takes only 0.758s, and editing the geometries of PRs takes 2.889s.

Fig. 4.10: Decorating and editing a Piggy Bank model. (a) shows the input model. (b) shows the first decoration result by using PVGs. (c) is the second result after changing color whose Voronoi diagram remains unchanged. (d) is the final result after changing dDC's shape of two clouds whose Voronoi diagram is locally updated with some additional knots and only the vertices $\mathbf{x}'$ inside these new Voronoi cells are recomputed by using equation $u(\mathbf{x}') = \sum_{\mathbf{v}_j \in N(\mathbf{x}')} \lambda_j \psi_j(\mathbf{x}')$. There are 1,073 knots in (g), and 214 additional knots are added in (h).

Thanks to the smooth features provided by PRs, our method can also allow multi-level modeling for 3D mesh (see Fig. 4.11).

Our decoration results illustrate that PVG can create rich and attractive effects on 3D meshes in both color and geometry aspects by sketching and global parametrization. Particularly in the field of synthesizing 3D detailed geometry, in contrast to the sharp displacement produced by only using DCs, our PRs can create smooth reliefs and allows

Fig. 4.11: Modeling a tree with multi-level PVG. Given a rectangular domain (a), user sketches a circular dPR enclosed by a dDC whose Dirichlet boundary condition is 0. The resulting PVG is the tree’s trunk (b). Then user sketches two other sets of DC and PR (c) to define two branches (d). Repeating this procedure a few times, we obtain the tree (e). Similar to other figures, we use color map to visualize the values of f and g , i.e., red for positive value and green for 0.

level-to-detail editing. In addition, Table. 4.1 gives statistics of these results in this section.

Triangle mesh	#cDC	#cPR	#dDC	#dPR
Bunny (Fig. 4.5)	82	0	82	81
Stele (Fig. 4.6)	0	0	22	31
Vase 1 (Fig. 4.7)	385	0	155	155
Vase 2 (Fig. 4.7)	264	0	264	264
Rhino (Fig. 4.8)	0	0	3038	3038
Sofa (Fig. 4.8)	27144	0	27144	27144
Cactus (Fig. 4.9)	854	0	122+854	122+854
Tree (Fig. 4.11)	$1 + 119 \times 2$	$1 + 119 \times 2$	$1 + 119 \times 2$	$1 + 119 \times 2$
Piggy bank (Fig. 4.10)	17(+3)	66(+15)	17(+3)	17(+3)

Table 4.1: Statistics of results. We report the number of DCs and PRs used. The prefixes “c” and “d” distinguish the primitives for color and displacement.

Jeschke et al. [9] presented a geometric detail decoration approach with DCs. Since the C^0 continuity of Laplacian/Poisson’s equation on boundary, the details are inevitable with sharp boundaries. Though the feature is pleasing sometimes, such as inlay ornaments on

vase, smooth feature is pleasing for some other occasions, such as Figure 4.11. Our C^1 continuous PR provides a natural multi-level tool for editing complicated decoration.

Finite element method [70] just evaluates C^0 continuity solution of Poisson equation. Though high-order elements could result in smooth solution, the computational expense increases dramatically.

4.4 Summary

Poisson vector graphics (PVG) is a powerful extension from the commonly-used Diffusion Curves (DC) by appending another two primitives — Poisson region (PR) and Poisson curve (PC).

This chapter has advocated a novel decoration technique working directly on arbitrary topological meshes guided by PVG primitives. DCs are leveraged to define height and color on 3D models as the boundary condition. Meanwhile, both PRs and PCs could produce the shading effect with offsetting Laplacians. By utilizing them together, we design a parameterization-free framework. It allows the users to create their own artistic decorations on 3D models directly, by drawing only a few sparse curves associated with PVG representation. In terms of the algorithms, we extend Poisson solver from planar domain to 3D meshes with harmonic B-spline based on the geodesic Voronoi diagram. Comparing with the existing methods supported by DCs only, our method can synthesize geometric detailing including both colors and the smooth embossing/carving effects. Thanks to the smoothness effect supported by PRs, we provide hierarchical level-of-detail editing on models. And for the Poisson solver’s localization, our method lessens the computational load, while providing more versatility like disregarding computation for the invisible positions out of sight.

To render PVG with our mesh Poisson solver, one significant step is to compute Green’s function for the vertices of 3D meshes, and then approximately calculate the function value for other points within the triangular faces. Therefore, we assume the source models are of high resolution and good tessellation quality. It is highly desirable to compute Green’s functions for non-vertex points in a more precise way. In addition, the scalar functions is adopted to define the displacement area. We tend to generalize our approach to vector-valued functions, thus, it can synthesize non-height function based displacement features.

Chapter 5

Vectorization based Color Transfer for Portrait Images

This chapter presents a Poisson vector graphics (PVG) based approach to vectorize portrait images and then transfer color between source and reference. First, we extract the face mask through a pre-trained neural network, and vectorize each source or reference images utilizing several sparse DCs to encode the dominant (low-frequency) colors, and then calculate PRs to encode the residual (high-frequency) details. Next, we transfer DCs' boundary color between the source and reference images by applying the optimal mass transport. Finally, we use an efficient random-access Poisson solver to render the final PVG consisting of the color-changed DCs and the original or modified PRs. Compared with existing methods that require 3D coordinate positions/normals or assume dense correspondences and alike poses between reference and source images, our method does not require pixel-to-pixel dense correspondence or 3D information, and it can process portrait images with greatly different color and postures.¹

5.1 Background

In the modern world, taking portrait and self-portrait photographs and videos has become a cultural phenomenon that more and more people have turned to photo sharing on social media. There are a lot of programs and apps allowing the user to retouch the photos, such as editing eyes, brightening skin, removing wrinkles, reshaping face, etc. Since it

¹The work in this chapter has been published in [3].

is a time-consuming work to manually tune colors even for skilled users, color transfer provides a pragmatic approach to tune a source image's colors to harmonize with the reference images' tone.

Color transfer has been studied extensively since 2000. Reinhard et al. [4] pioneered the color transfer method that matched the mean and standard deviation of color distributions of the source and reference images. Hwang et al. [5] applied probabilistic moving least square for color transfer between images with the same scene but different camera settings and illumination conditions. Shih et al. [2] presented a good framework for transferring styles between portraits, which matched the global lighting and the local contrast by transferring the local statistics. Their method can produce visually pleasing results, however it requires a dense correspondence between the source and the reference image. Shu et al. [1] developed a method to transfer both lighting and colors for portrait photos. They applied shape from shading technique to estimate the face normal and then formulated the transfer problem as a 8-dimensional optimal mass transport (OMT), considering colors, pixel coordinates and face normals. Though their method produces excellent results, since its formulation is conducted in 8D space, it is computationally expensive to process a high-definition image.

In this chapter, we aim at proposing an easy and effective approach to transfer color for portrait images. We first extract the face mask through a pre-trained neural network, and vectorize each source or reference images by utilizing several sparse DCs to encode the low-frequency colors, and then calculate PRs to encode the high-frequency details. Then we transfer DCs' boundary color between the source and reference images by applying the optimal mass transport. Finally, we use an efficient random-access Poisson solver to render the final PVG consisting of the color-changed DCs and the original or modified PRs. Thanks to our local Poisson solver, the updated colors is able to be rendered locally. Compared with existing approaches that require 3D information or assume dense correspondences and alike poses between reference and source images, our method requires no pixel-to-pixel dense correspondence or 3D information to process portraits of different poses.

Since our method conducts optimal mass transport on a simple set of sparse curves (DCs), it also can transfer skin color of particular facial textures such as tattoo and

freckle, and serves well for not only photographs but also paintings. In addition, our system provide support for PRs tuning to obtain special artistic styles like oil-painting or watercolor effects. Figure 5.1 gives a few of our results that transferring skin/hair colors.

5.2 Overview

Our approach mainly includes four steps. First, we vectorize the source and reference images by using DCs and PRs. DCs are a simple set of two-sided curves with colors defined on both sides, that encode the salient colors. Figuratively, vectorization is like to “push” the low-frequency colors of the non-salient field into the boundary colors of the extracted DCs. PRs are the Laplacians of color residual, which represent the high-frequency details. Second, by adopting a pre-trained neural network, we extract facial masks and use them to label the DCs as 3 types: background, hair and skin. Third, we formulate the problem of color transfer as an optimal transport between the labeled DCs, and adopt the sliced Wasserstein distance algorithm to solve it. Finally, to efficiently render the updated PVGs, we obtain the retouched portrait with a local Poisson solver.

Figure 5.2 shows the pipeline of our color transfer approach and the corresponding pseudo-code is provided in Algorithm 1.

Algorithm 2 Vectorization based color transfer on portrait images

Require: The input image I , the reference image R

Ensure: The output image O

- 1: Vectorize I and R with diffusion curves DC and Poisson regions PR (Section 5.3)
 - 2: Label the DCs as background, hair and skin (Section 5.4)
 - 3: Transfer the boundary colors of DC_R to DC_I using optimal mass transport (Section 5.5)
 - 4: Render the updated PVGs with new boundary colors DC'_I and the original or modified Poisson regions PR_I by using a local Poisson solver (Section 5.6)
-

5.3 Vectorization

Since only using DCs to vectorize portrait images, it requires a large number of DCs to well reconstruct the original images, that basically makes no difference between the method conducting on raster images. To decrease the number of DCs and meanwhile

Fig. 5.1: Examples of skin and hair color transfer with our method. The original images are shown in the first column of the first/third row and the reference images are shown in small insets. Note that our method does not require a dense correspondence between the input and the reference, therefore it can handle portraits with different postures.

Fig. 5.2: Our algorithmic pipeline that taking skin color transfer as an example. The labeled DCs fall into three classes: skin (red), hair (blue) and background (green).

maintain detailed features, we separate raster images into two levels. One is the low-frequency features like the salient colors that stored in DCs, and the other is the high-frequency details that encoded in PRs. Specifically, we apply edge detection method to extract sparse DCs, Laplace operator to compute PRs, and efficiently render and update vector images by using a local Poisson solver. Figure 5.3 gives an example of portrait image vectorization.

First, we apply the edge detection approach [78] to extract DCs as boundary condition. Any detected edge of non-zero probability is deemed as a DC candidate. And we preserve the edges as DCs, whose gradients exceed a predefined threshold. Because color leap always appear where there is a large gradient and comparative colors represent salient colors around to the greatest extent. For our color transfer algorithm, only the edges with sharp color variations are required, so our DC extraction approach is reliable for this application.

Then, we use 2D Laplace operator to compute PRs as Laplacian constraints. In discrete cases, Laplacians can be simply obtained based on the composition image that appending the extracted DCs to the original image. For each pixel p_i , let p_j be the top, bottom, left, and right neighbor of p_i in the composition image, we can obtain the

Fig. 5.3: Vectorizing a portrait image. (a) The original image. (b) Extracted DCs with boundary colors. (c) The DCs represent only the low-frequency colors. (d) The PRs are pixel-level regions encoding high-frequency details. (e) The combination of DCs and PRs can produce a well-reconstructed vector image with high accuracy. (f) Label the DCs as 3 types: background (green), hair (blue) and skin (red).

Laplacian constraint of p_i through RGB channels:

$$Lap(p_i) = \sum_{j=1}^4 (RGB(p_j) - RGB(p_i)). \quad (5.1)$$

Finally, combining boundary condition g and Laplacian constraints f , we render and update vector images by using our local Poisson solver to solve the Poisson equation (5.5) (see Section 5.6).

5.4 Labelling Facial Features

To segment facial features from portrait images automatically, we apply a deep learning based approach presented in [79], which is one of the state-of-the-art face parsing methods.

The network structure is the FCN-8s [80] that is commonly used for semantic image segmentation. Here we train the labeling CNN model on HELEN dataset [81], which includes 2330 facial images of arbitrary size, and each pixel of the images is labeled as 11 types (right eye, left eye, right eyebrow, left eyebrow, nose, upper lip, inner mouth, lower lip, facial skin, hair, and background). Figure 5.4 shows 3 major labels of the above portrait in Figure 5.3, i.e., background, skin, and hair.

Fig. 5.4: A portrait image is segmented with 3 semantic labels: background, skin, and hair.

After vectorization, DCs use a set of curves to encode the salient colors of original images, and then color transfer can be efficiently implemented in these sparse curves. In general, only a subset of DCs belongs to the target labels such as hair or skin, so we automatically label DCs by using the above facial segmentation. Specifically, we use the segmentation masks of skin (the combination of facial skin, mouths, nose, eyes and brows), hair and background to label each control point of DCs, since some curves are too long to always fall in only one category. And we find that DCs are sparsely distributed and the number of DCs is small. As a result, even if the segmentation is not so accurate, the labeling result is still relatively good.

Nonetheless, the labeling result cannot always satisfy our requirements, especially for skin color transfer, such as that the neck area is labeled as background rather than skin. Because the skin segmentation only contains facial skin that does not include other areas like neck and hands. Moreover, it cannot separate facial skin and background completely accurately.

Though, the semantic segmentation can provide help to correct the recognition process of the extracted DCs. Take skin labeling as an example, we first sort the colors of DCs labeled as facial skin, and break off both ends to remove the superfluous parts brought by inaccurate recognition. Next we obtain a range of skin color to measure other

DCs whether they belong to skin or not. In actuality, this strategy is applicable when skin tones can be significantly distinguished from background and hair. Otherwise if their color is similar, this method cannot improve the accuracy of automatical labeling. Therefore, our system allows user to specify and change labels where necessary. In our practical implementation, the perfect labels are not necessary, especially for the reference images.

5.5 Color Transfer with Optimal Mass Transport

Optimal mass transport is an effective approach for color transfer by matching the color histograms of the source and reference images. In short, it provides a solution to transporting the color histogram of the source to the reference with minimal effort.

Specifically, let $I = \{c_i | c_i \in \mathbb{R}^{m \times 3}\}_{i=1}^m$, $R = \{c_j | c_j \in \mathbb{R}^{n \times 3}\}_{j=1}^n$ be the RGB color of the source and reference images, and H_I and H_R be their normalized color histograms. A transport plan is $\sigma: \{c_i\} \rightarrow \{c_j\}$, $\sigma = \{\sigma_{ij}\}$, σ_{ij} is the number of pixels of color c_i transferred to color c_j . Then the optimal mass transport plan can be defined as:

$$\arg \min_{\sigma} \sum_i \sum_j \sigma_{ij} \|c_i - c_j\|^2 \quad (5.2)$$

with constraints:

$$\sigma_{ij} \geq 0, \quad \sum_{j=1}^n \sigma_{ij} = H_{c_i}, \quad \sum_{i=1}^m \sigma_{ij} = H_{c_j}.$$

This problem is known as the Kantorovich's formulation [82], and it can be solved by linear programming method. Nonetheless, this solution may transfer some pixels of the same color to different color sets, which is an unsatisfactory function for the color transfer application.

The Monge's formulation [83] is to find a transport map $T: \{c_i\} \rightarrow \{c_j\}$ that realizes

$$\arg \min_T \sum_{i=1}^m H_{c_i} \|c_i - T(c_i)\|^2 \quad (5.3)$$

with constraints:

$$H_{T(I)} = H_R.$$

It provides a resolution that ensure all pixels of a same color transferred to a single color set. However, the Monge's formulation may have no solution, if there is no T satisfying $H_{T(I)} = H_R$, for instance, the number of color I is not equal to color R .

Hereby, we adopt the sliced Wasserstein distance algorithm [84] to solve the transport problem. This algorithm utilizes an iteration approach to obtain an approximate solution as $H_{T(I)} \approx H_R$:

$$T^{k+1}(I) = (1 - \alpha)T^k(I) + \alpha T(T^k(I)), \quad k = 1, 2, \dots, N. \quad (5.4)$$

with initial condition:

$$T^1(I) = I$$

where α is the linear parameter, and N is the number of iterations. This algorithm guarantees a certain solution of the transport problem, meanwhile, the solution transfers all pixels of the same color to a single color.

In 1D space, the sliced Wasserstein distance algorithm can simply attain the resolution based on cumulative distribution function of the color differences of the source and reference images. Moreover, in multi-dimensional space (e.g. 3D RGB space), this algorithm uses its 1D algorithm in each projection of multi-dimensional data with a random basis and then iterates to attain a final approximate result. Comparing with directly adopting a multi-dimensional transport algorithm, the sliced algorithm is more efficient and reduces computational expense. In our experiment, we assign $\alpha = 0.2$ as in [85] and $N = 30$ whose results satisfy the visualization requirements. Figure 5.5 gives the transfer results of different iterations.

To reduce color leap of the skin color transfer result, we add Gaussian noises to the color set of the source and reference images as in [1]. As a result, each color can be transferred to a relatively similar color. In my experiment, these noises increase the original data by five times, but in fact it has a small impact on the running time, since the data is just some sparse curves and it leads to smoother and more natural results. According to experimental observation, it requires no additional noises for hair color transfer to obtain a satisfying result, since skin color of one person often remains a single tone, whereas the hair may have various vibrant colors.

After classifying DCs as skin, hair, and background, the objective of color transfer becomes labeled DCs from one class, e.g. hair, and the target of optimal mass transport is a set of sparse curves. Then, we use the Sliced Wasserstein Distance algorithm to transfer color between DCs of the same label of source and reference images. Finally,

Fig. 5.5: Color transfer for skin and hair with different iterations N . The first column of the first/third row show the input images and the last column of the second/last row are the references. The middle columns (from front to back) are the results with $N = 1, 5, 10, 20, 30, 50$ respectively.

we render the updated vector graphics to attain the color transfer result. Actually, this process changes the boundary condition and PRs remains unchanged. In other words, the detailed texture of source keeps the same, while the color is changed.

5.6 Rendering

To update and render vector graphics, we solve the Poisson function

$$\Delta u(\mathbf{x}) = f, \quad u(\mathbf{x})|_{\mathbf{x} \in \partial\Omega} = g, \quad (5.5)$$

where f is the Laplacian constraints, and g is the Dirichlet boundary condition of colors. We construct a harmonic B-spline [13] based local solver for (5.5).

Let $\Omega \subset \mathbb{R}^2$ be a 2D compact domain and $\mathcal{T} = \{\mathbf{t}_i | \mathbf{t}_i \in \Omega\}_{i=1}^n$ a set of knots. Taking $\{\mathbf{t}_i\}$ as the generators, we build a Voronoi tessellation $\Omega = \bigcup_{i=1}^n \mathcal{V}_i$, where $\mathcal{V}_i$ is the Voronoi cell of knot $\mathbf{t}_i$. For any point $\mathbf{x}, \mathbf{y} \in \Omega$, Green's function of the Laplace operator Δ satisfies

$$\Delta \phi_{\mathbf{y}}(\mathbf{x}) = \delta_{\mathbf{y}}(\mathbf{x}), \quad (5.6)$$

where $\delta_{\mathbf{y}}(\mathbf{x})$ is the Dirac delta function centered at $\mathbf{y}$. In 2D space, we obtain the analytic solution

$$\phi_{\mathbf{y}}(\mathbf{x}) = \frac{1}{2\pi} \log(\|\mathbf{x} - \mathbf{y}\|).$$

For a Voronoi cell $\mathcal{V}_j$, using Green's theorem to Eqn. (5.6) derives

$$\int_{\mathcal{V}_j} \Delta \phi_{\mathbf{y}}(\mathbf{x}) \, d\sigma = \int_{\partial\mathcal{V}_j} \frac{\partial \Delta \phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} \, ds, \quad (5.7)$$

where $\mathbf{n}$ is the outward unit normal to the boundary $\partial\mathcal{V}_j$, $d\sigma$ and ds are the area and line integral elements, respectively.

We define a basis function ψ_j for each Voronoi cell $\mathcal{V}_j$ as

$$\psi_j(\mathbf{x}) = \sum_i w_{ij} \phi_{\mathbf{t}_i}(\mathbf{x}), \quad (5.8)$$

where w_{ij} is the discrete Laplacian weight and $\sum_i w_{ij} \phi_{\mathbf{t}_i}(\mathbf{x})$ is a boundary sum that approximates the line integral on the right hand side of Eqn. (5.7). Feng and Warren [13] demonstrated that the function ψ_j is nonnegative, approximately local, and satisfying partition of unity.

Green's third identity gives an analytical solution of Poisson function in integral form

$$\begin{aligned}
 u(\mathbf{x}) &= \iint_{\Omega} \phi_{\mathbf{y}}(\mathbf{x}) \Delta u(\mathbf{y}) d\sigma_{\mathbf{y}} \\
 &+ \oint_{\partial\Omega} \left(u(\mathbf{y}) \frac{\partial \phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} - \phi_{\mathbf{y}}(\mathbf{x}) \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}} \right) dl_{\mathbf{y}},
 \end{aligned} \tag{5.9}$$

where dl and $d\sigma$ are the line and surface elements, $\mathbf{n}$ is the outward pointing unit normal of dl .

Applying harmonic B-spline [14], we calculate an approximate solution by

$$u(\mathbf{x}) = \sum \lambda_j \psi_j(\mathbf{x})$$

where if λ_j is a boundary, assign it with the Dirichlet boundary condition g ; otherwise, compute it by solving a sparse linear system.

By using this Poisson solver on vector images, we are able to update and render them locally.

5.7 Results

Our approach has obtained usually satisfactory color transfer results for portrait images. For skin color transfer, we can generate natural and consistent skin color with basically remaining other features or regions unchanged. Figure 5.6 gives the results of skin color transfer between six real portrait photos of different skin colors, which are ordered from fair to dark.

Except for real portrait photos, our approach is able to transfer skin color between painting-to-photo and painting-to-painting images. In Figure 5.7, the first two rows show the skin color of watercolor/oil paintings are transferred according to real photons, the other two of the first column show the transferred result between these two paintings, and the last six of the third row is transferring photos according to watercolor painting (the first three) and oil painting (the last three). To simulate the style of watercolor/oil painting (the last row), we half/double their PRs based on the color transfer results of the above row.

Furthermore, the portrait lighting approach presented in [1] can also be applied to transferring skin color. To transfer color and lighting of portrait images, they use a mass

Fig. 5.6: Skin color transfer results. The red frames indicate original images, and the others are the transferred results.

transport method on 8D data of facial regions involving RGB colors, 2D coordinates, and 3D normals. So they implemented the transport in the entire facial regions, whose data volume is much more than sparse curves as ours. Since they required to solve a high dimension problem, their approach is computationally expensive. It took about 3 minutes to transfer a portrait image of size 500×600 . In our approach, vectorization is the bottleneck, which takes 30 to 60 seconds. After that, labelling DCs and transferring color are highly efficient, taking only a few seconds. Rendering the updated vector graphics with the local solver also takes a few seconds. Thus, our approach is more efficient than Shu et al.’s approach [1]. In addition, since our result is represented by vector graphics (a set of sparse curves), it is relatively easy for downstream applications, e.g., editing the geometry of facial features, seamless cloning, etc.

Rather than front faces, our approach can also generate satisfactory transfer results for other postures, e.g., side face, head down, head up (see Figure 5.8). Vectorization segments raster images into two levels where the color change is in the low-frequency signals, thus, our approach can retain the texture and details of original images furthest. As shown in Figure 5.9, whether tattoos or freckles on faces or partially covered by some objects such as sunglasses, we can still attain a good result preserving facial features.

It is worth mentioning that Shih et al.’s approach [2] requires a dense correspondence between the source and reference images, thus, it can only handle portrait photos with similar facial features and head postures. In contrast, our approach does not require such a correspondence, so that it is flexible and can work for portraits with different facial features and head positions (see Figure 5.8). In addition, our approach can also process non-photorealistic images, e.g., paintings and cartoons.

Furthermore, we compare our approach with [1] and [2]. As shown in Figure 5.10, our approach can obtain comparable results by maintaining facial highlights and skin textures. Relatively speaking, these three approaches have slightly different objectives: our method targets skin and hair colors, Shu et al.’s approach [1] works for lighting transfer and Shih et al.’s approach [2] focuses on skin colors and textures. Our approach is more computationally efficient than the other methods and it does not require dense correspondences between the source-reference image pair.

Except for skin color, our system works well for hair color transfer too. Figure 5.11 shows the results of transferring two girls’ hair color (the first column) to various hair

Fig. 5.7: Skin color transfer between oil/watercolor paintings and real portraits. The first column of the first two rows and the mini images are original.

Fig. 5.8: Color transfer between portraits of different head positions. The images in red frames are original.

Fig. 5.9: Our method can maintain original details such as sunglasses, freckles, and tattoos.

Fig. 5.10: Comparison with the state-of-the-art related works [1] and [2].

colors, regardless of whether they have similar hair color or not. And we obtain satisfactory results, transferring not only the hair color with a single color but also with multiple colors. Besides, our system also accepts paintings as the references (the last column), rather than real photos.

Fig. 5.11: Hair color transfer for photos and paintings. The first column is the inputs, the second and fourth rows are the references.

Our current labels for portraits fall into three main classes including skin, hair, and background. As shown in Figure 5.12, by labelling DCs of the regions of mouths and eyes, we can also transfer the colors.

5.8 Summary

This chapter presents an effective but simply-implemented approach for transferring hair and skin color between portrait images. By conducting PVG vectorization to the source

Fig. 5.12: Color transfer for mouths and eyes. Left: the original image; middle and right: the transferred results.

and reference images, the low frequency is expressed by sparse DCs containing the main colors, and the high frequency is presented by PRs including residual details. Optimal mass transportation is then leveraged to change DCs' boundary colors between the source reference images. The experimental outcomes provide some results of color transfer on hair and portraits skin, but actually, color transfer of smaller regions like eyes or mouth is transferable in a similar way. Moreover, our approach can also be utilized to transfer colors for landscapes or other objects.

Nonetheless, our method has limitations. If the source and reference images have massively distinct colors, the color transfer outcomes may be artificial, even though tuning their PRs can slightly enhance the results as shown in Figure 5.13. The facial segmentation is not precise enough, and we could completely avoid labeling by the users if we had precise segmentation masks in the future. Furthermore, we do not provide quantitative/numerical measurements to evaluate the effectiveness of our method except for the visual results. We tend to develop a quantitative evaluation for the color change results, and it can be also utilized to automatically locate the most fitting reference images.

Fig. 5.13: Our method fails on portraits with significantly different color distributions. Note the input image has a very low contrast, whereas the reference image has a high contrast. Tuning the PRs can reduce the artifacts.

Chapter 6

Poisson Vector Graphics (PVG)-Guided Face Color Transfer in Videos

This chapter presents a simple yet effective algorithm for automatically transferring face colors in portrait videos that extending the method conducted for portrait images as described in the above chapter. We extract the facial features and vectorize the faces in the input video using Poisson vector graphics, which encodes the low-frequency colors as the boundary colors of diffusion curves, and the high-frequency colors as Poisson regions. Then we transfer the face color of a reference image/video to the first frame of the input video by applying optimal mass transport between the boundary colors of diffusion curves. Next the boundary color of the first frame is transferred to the subsequent frames by matching the curves. Finally, with the original or modified Poisson regions, we render the video using an efficient random-access Poisson solver. Thanks to our efficient diffusion curve matching algorithm, transferring colors for the vectorized video takes less than 1 millisecond per frame. Our method is particularly desired for frequent transfer from multiple references due to its information reuse nature. The simple DC matching also greatly improves the performance of video vectorization, since we only need to solve an optimization problem for the first frame. Since our method does not require correspondence between the reference image/video and the input video, it is flexible and robust to handle faces with significantly different geometries and postures, which often pose challenges to the existing methods. Moreover, by manipulating Poisson regions, we can enhance or reduce the highlight and contrast so that the reference color

can fit into the input video naturally. We demonstrate the efficacy of our method on image-to-video transfer and color swap in videos.¹

6.1 Background

In the digital age, portrait and self-portrait photographs and videos are extremely popular and have spread to every corner of the world. There are many programs and apps allowing the user to retouch the photos, such as editing eyes, brightening skin, removing wrinkles, reshaping face, etc. However, editing videos is still a challenging task.

In this chapter, we are interested in automatically transferring face color from a reference image/video to an input video. Though color transfer has been studied in the computer vision and graphics community for almost two decades, there are only a few works for portrait photos. Shu et al. [1] developed a method to transfer both lighting and colors for portrait photos. They applied shape from shading technique to estimate the face normal and then formulated the transfer problem as a 8-dimensional optimal mass transport (OMT), considering colors, pixel coordinates and face normals. Though their method produces excellent results, it takes a few minutes to process an HD image. Such a high computational cost diminishes its application to video color transfer. The above chapter [3] proposed a vectorization-based method that reduces the region color transfer problem to a boundary color transfer, hereby is efficient. They solved a Poisson’s equation to render the vector graphics. However, since their results are highly sensitive to the PDE’s boundary conditions, directly applying their method to video leads to unpleasing visual artifacts, such as flickering.

To overcome the challenges of the existing work, we propose a simple yet effective method for transferring face color in videos. We extract the facial features and vectorize the face(s) of the input video using Poisson vector graphics, which encodes the low-frequency colors as the boundary colors of diffusion curves (DCs), and the high-frequency colors in Poisson regions (PRs). Then we transfer the face color of a reference image/video to the first frame of the input video by applying OMT between the boundary colors of DCs. Next the DC boundary color of the first frame is transferred to the subsequent

¹The work in this chapter is derived from [86], which has been submitted for publication.

frames by matching the diffusion curves. Finally, with the original or modified PRs, we render the video using an efficient Poisson solver. Thanks to our highly efficient DC matching algorithm, transferring colors for the vectorized video takes less than 1 millisecond per frame. Since we only apply OMT to the first frame, our method is computationally efficient and stable. For a single transfer from one reference image, the absolute performance of our algorithm, including pre-computation (i.e., vectorization) and post-computation (i.e., rendering) is comparable to the conventional raster image based method, which simply applies OMT to all frames. For frequent transfer from multiple reference images, our method runs one or two orders of magnitude faster than the image OMT method, due to its information reuse nature. The simple DC matching algorithm also greatly improves the performance of video vectorization, since we only need to solve an optimization problem for the first frame. Since our method does not require correspondence between the reference image/video and the input video, it is flexible and robust to handle faces with significantly different geometries and postures, which often pose challenges to the existing methods. Moreover, by manipulating the PRs, we can enhance or reduce the highlight and contrast so that the reference color can fit into the input video naturally. We demonstrate the efficacy of our method on image-to-video transfer and color swap in videos.

6.2 Overview

As a pre-computation method, our method first vectorizes the input video and then performs color transfer for arbitrary reference image/video.

To vectorize the video, we first detect the facial landmarks, and obtain a face mask by extracting the skin region for each frame. Next we use the extracted facial landmarks and the boundary of the face mask as diffusion curves (DCs) [10], whose boundary colors are computed by solving a linear least square problem. We encode the low-frequency skin colors using diffusion curves and the high-frequency details using Poisson regions (PRs) [18]. We refer the reader to the supplementary material [10, 18] for more details about diffusion curves and Poisson regions. The reference image is also vectorized using Poisson vector graphics.

To transfer face colors, we adopt the optimal mass transport model [28] that matches the color distribution of the source and the reference in an exact manner. Note that naïvely applying OMT to all frames is obviously time consuming. In our approach, we apply OMT only to the first frame. Then, taking advantage of the vectorized video, we transfer the DC boundary colors to the subsequent frames by our DC matching algorithm, which is efficient and can also guarantee the smoothness of the DC boundary colors in-between frames, hereby effectively reducing flickering artifacts.

Figure 6.1 shows the pipeline of our color transfer method and Algorithm 1 presents the pseudo-code. The most distinctive feature of our method is information reuse. For frequent transfer from multiple reference images, our method runs one or two orders of magnitude faster than the image OMT method.

Fig. 6.1: Algorithmic pipeline. Thanks to its information reuse nature, our algorithm is highly desired for frequent color transfer from multiple reference images.

6.3 Video Vectorization

Vectorization plays a critical role in our method. A naïve method is to simply apply the image vectorization method (e.g., [3]) to every frame. However, it would be very time

Algorithm 3 PVG-Guided Face Color Transfer for Videos

Require: The input video with n frames $I = \{I_i\}_{i=1}^n$, a reference image R , and (optional) $\lambda \in [0.7, 1.5]$ the PR coefficient.

Ensure: The output video $I' = \{I'_i\}_{i=1}^n$ with transferred face color

► Pre-computation: vectorize I

for $i = 1 : n$ **do**

 Extract facial features and skin mask for I_i

 Construct diffusion curves γ_i using the extracted features

if $i > 1$ **then**

 Compute $\pi_{i-1} : \gamma_{i-1} \rightarrow \gamma_i$

end if

end for

Compute g_1 for I_1 by solving Eqn. (6.3)

Compute Poisson regions f_1 for I_1 using Eqn. (6.4)

for $i = 2 : n$ **do**

$g_i = g_{i-1} (\pi_{i-1}^{-1}(\gamma_i))$

 Compute Poisson regions f_i for I_i using Eqn. (6.4)

end for

► Transfer DC boundary colors

Transfer color to I_1 using OMT $g'_1 = \psi(g_1, g_r)$

for $i = 2 : n$ **do**

 Transfer color to I_i using $g'_i = g'_{i-1} (\pi_{i-1}^{-1}(\gamma_i))$

end for

► (Optional) Edit contrast and highlight

for $i = 1 : n$ **do**

 Update Poisson regions for I_i by $f'_i = \lambda f_i$

end for

► Render vector primitives

for $i = 1 : n$ **do**

 Compute I'_i by solving Eqn. (6.1) with f'_i and g'_i

end for

consuming, since one had to solve a least square problem n times. In this chapter, we develop a simple yet effective method to vectorize face videos. Our key idea is to solve the optimization problem only for the *first* frame and then transfer the boundary colors of DC to the subsequent frames by matching the curves. Since the computational cost of DC matching is significantly less than that of solving the least square problem, our method is much faster than the naïve image vectorization method.

6.3.1 Extracting Facial Features

Facial landmark detection is to locate a set of predefined Fiducial points on 2D faces. As a core component of various face applications, it has been studied extensively in the last decade. In our work, we implement a simple yet effective end-to-end deep neural network for robust facial landmark localization. We adopt MobileNetV2 [87] as the feature extractor with regression output layers. Instead of the L_2 loss, we adopt the wing loss [88] which focuses more on small- and medium-range errors. To deal with imbalanced facial pose datasets, we use online hard example learning [89]. To further improve the accuracy, we adopt the knowledge distillation method [90] that migrates knowledge from the high-precision model to the real-time model by collecting a large amount of unlabeled data. Our model is compact with a size of 2.2 Mb and can run 125 fps per face for input size 160×160 on iPhone 7 Plus. Figure 6.2(b) shows the detected 106 facial landmarks. Using the detected facial landmarks, we crop the input video and vectorize only the region of interest.

To extract the face mask, we adopt the YCbCr color space, where Y is the luma component and Cb and Cr are the blue-difference and red-difference chroma components. As pointed out by Hsu et al. [91], skin colors are concentrated in the YCbCr space. Specifically, the pixels in the CrCb sub-space follow an approximately elliptical distribution, and the YCbCr color space does not depend on luminance. Based on this observation, we set a range $[140, 160]$ for the Cr component to extract the domain of skin. Since the domain may be disconnected and contain non-facial part, we adopt a simple heuristic by taking the largest connected component. Then we take its boundary as the face mask (see Fig. 6.2(c)). Furthermore, the probability edge extraction [92] can help us obtain more robust skin mask from some cases whose skin color is not located in the Cr range

Fig. 6.2: Face vectorization. (a) shows two frames of an input video. We detect 106 feature points for each frame (b) and obtain a face mask by extracting skin colors (c). We use the feature points and the boundary of the mask as diffusion curves (d). We compute their boundary colors (e) by solving a linear least square problem. To improve the performance, we use harmonic B-spline solver which is built upon a quad-tree data structure. DCs encode the low-frequency (LF) colors (g) and PRs encode the high-frequency (HF) details. (f) shows the color-coded PRs. Using DCs and PRs, we can faithfully reconstruct the input video (h). Moreover, by changing the coefficients of PRs, we can edit contrast and highlight (i-j).

but clearly distinguished from the background, e.g. Fig. 6.7. We observe this simple technique works very well for face videos with various colors and postures.

6.3.2 Computing Boundary Color g for DCs

Given an image I and the specified feature curves consisting of the facial landmarks and the boundary of the skin mask, our algorithm assigns colors of DCs and evaluates PRs automatically. The DCs represent significant boundaries containing the main color derived from the image boundaries. The PRs represent highlights/shadows and geometric details derived from image regions (skin mask in our implementation) with pixel Laplacians. Figure 6.2 illustrates the face vectorization algorithm.

To render and update the vector images, we solve the Poisson's equation

$$\begin{cases} \Delta u(\mathbf{x}) = f, & \mathbf{x} \in \Omega \setminus \partial\Omega \\ u(\mathbf{x})|_{\partial\Omega} = g, & \mathbf{x} \in \partial\Omega \end{cases} \quad (6.1)$$

where f is the Laplacian constraints (PRs), g is the Dirichlet boundary condition of colors (the boundary colors are defined on the DCs) and Ω is a 2D compact domain.

And by using harmonic B-splines [13], we can solve the Poisson's equation with a closed form as

$$u(\mathbf{x}) = \sum_{i=1}^n \lambda_i \phi_i(\mathbf{x}) \quad (6.2)$$

where λ_i is the discrete Laplacian weight, and $\phi_i(\mathbf{x})$ is the basis function of harmonic B-splines.

To determine color g of boundaries γ , we build a quad tree to tessellate the skin region and then construct a harmonic B-spline whose basis functions $\phi_i(\mathbf{x})$ are defined on the nodes of the quad tree. Then, we compute the control coefficients λ_i by fitting the input image I in a linear least square

$$\arg \min_{\lambda_j} \left(\sum \lambda_j \phi_j(\mathbf{x}) - I \right)^2. \quad (6.3)$$

We obtain the boundary color g by evaluating the color function u on the boundaries $g = u|_{\gamma}$.

After that, we compute the Laplacian constraints f by applying the Laplacian operator to all interior pixels

$$f = \Delta \left(I - \sum \lambda_i \phi_i(\mathbf{x}) \right), \quad \forall \mathbf{x} \in \Omega \setminus \partial\Omega. \quad (6.4)$$

In the previous chapter, we apply color transfer to portrait images. The image is vectorized into diffusion curves for low-frequency colors and Poisson regions for high-frequency details. To change the color, we transfer the boundary colors of the diffusion curves between the source and reference images by using optimal mass transport.

We apply the above method only to the first frame of the input video and obtain $g_1 : \gamma_1 \rightarrow \mathbb{R}^3$. For the subsequent frames, we do not need to compute the boundary colors g_i , $i > 1$, for the diffusion curves. Instead, we match the diffusion curves and then transfer g from the first frame to the remaining frames. This simple strategy not only reduces computational cost significantly, but also ensures the skin colors are consistent in all frames.

The 106 facial landmarks are semantically labelled, hence providing a one-to-one correspondence between frames. Denote by $\pi_i : \gamma_i \rightarrow \gamma_{i+1}$ the map between two consecutive frames. Given the boundary color $g_i : \gamma_i \rightarrow \mathbb{R}^3$ of the i -th frame, we compute the boundary color of the $(i + 1)$ -th frame by

$$g_{i+1} = g_i \left(\pi_i^{-1}(\gamma_{i+1}) \right) \quad (6.5)$$

where $i = 1, 2, \dots, n$. Fig. 6.3 illustrates the above method for matching DCs' boundary color.

Fig. 6.3: DC boundary color matching. The boundary color of the first frame g_1 is computed by Eqn. (6.3). With the boundary color g_i for the i -th frame and the correspondence map $\pi_i : \gamma_i \rightarrow \gamma_{i+1}$, we compute the boundary color g_{i+1} by the inverse map π_i^{-1} .

After rendering by the Poisson solver, we get the vector frames and their residual ξ by calculating the difference between current rendering result and the origin frame.

6.4 PVG-Guided Color Transfer

Optimal mass transport is a powerful technique for image-to-image color transfer. Denote by I and J the input and reference images with N_I and N_J pixels respectively. It computes a transport map between the input color distribution $I = \{c_i | c_i \in \mathbb{R}^{3 \times 1}\}_{i=1}^{N_I}$ and the reference color distribution $J = \{c_j | c_j \in \mathbb{R}^{3 \times 1}\}_{j=1}^{N_J}$ with a minimum cost, where c_i and c_j are the pixel colors. To ensure all the instances of a color in the input are transferred to a single color, we adopt the Monge's formulation [83] to compute the transport $\mathcal{T} : \{c_i\} \rightarrow \{c_j\}$:

$$\arg \min_{\mathcal{T}} \sum_{i=1}^{N_I} H_{c_i} \|c_i - \mathcal{T}(c_i)\|^2 \quad (6.6)$$

with constraints $H_{\mathcal{T}(I)} = H_J$, where H_I and H_J are the normalized histograms of I and J .

Since the Monge's formulation may not always produce a solution, we adopt the sliced Wasserstein distance algorithm [84] to obtain an approximate solution $\psi(I, J)$ such that $H_{\mathcal{T}(I)} \approx H_J$:

$$\mathcal{T}^{(k+1)}(I) = (1 - \beta)\mathcal{T}^{(k)}(I) + \beta\theta M(\theta^{-1}\mathcal{T}^{(k)}(I), \theta^{-1}J) \quad (6.7)$$

with initial condition $\mathcal{T}^{(1)}(I) = I$, where $k = 1, 2, \dots, N$ is the iteration count, $\beta \in [0, 1]$ is a linear parameter balancing convergence speed and stability of the result, $\theta \in \mathbb{R}^{3 \times 3}$ is a randomly generated orthogonal matrix, and $M : \mathbb{R}^{3 \times N_I} \times \mathbb{R}^{3 \times N_J} \rightarrow \mathbb{R}^{3 \times N_I}$ is the histogram matching operator.

A naïve implementation is to simply compute the OMT map between each frame and the reference image. This is obviously time consuming. In our method, we compute the OMT map only for the diffusion curves of the first frame as follows:

$$\mathcal{T}^{(k+1)}(\gamma_1) = (1 - \beta)\mathcal{T}^{(k)}(\gamma_1) + \beta\theta M(\theta^{-1}\mathcal{T}^{(k)}(\gamma_1), \theta^{-1}\gamma_r) \quad (6.8)$$

with initial condition $\mathcal{T}^{(1)}(\gamma_1) = \gamma_1$, where γ_1 and γ_r respectively represent the DCs of the first frame I_1 and the reference image R . In our experiment, we set the coefficient $\beta = 0.2$ and the iterations $N = 20$ which is sufficient to produce visually pleasing result (see Fig. 6.4). Then we compute the transferred boundary color g_1 as $g_1 = \psi(\gamma_1, \gamma_r) \triangleq \mathcal{T}^{(N)}(\gamma_1)$.

The DC boundary colors of the next frame are obtained by transferring the boundary colors from the previous frame $g_i = g_{i-1}(\pi_{i-1}^{-1}(\gamma_i))$, $2 \leq i \leq n$.

Fig. 6.4: Iterative computation of the optimal mass transport map $\mathcal{T}$. We show a few representative intermediate results $I'_1|_k$ of the first frame in the top row and their corresponding histograms $H_{\mathcal{T}^{(k)}(\gamma_1)}$ in the bottom row. The iterative algorithm converges quickly and produces the final result in only 20 iterations.

6.5 Results

We implemented our algorithm in C++ and evaluated it on a PC with an Intel i7 CPU 2.80 GHz. Table 6.1 reports the running time of our method and the raster image based method.

We examine the time complexity of our method, the naïve image-based OMT that computes OMT for all frames, and the extended Fu et al.’s method [3] that applies their vectorization-based method for all frames. Denote by n the number of frames, m the number of references, T_{R-OMT} the time for computing image-based OMT for one frame, T_V the time for vectorizing one frame, T_{V-OMT} the time for computing vectorization-based OMT for one frame, T_{DC-M} the time for matching DCs between adjacent frames, T_R the time for rendering PVG.

The image-based OMT method takes mnT_{R-OMT} time, since it computes the OMT map for every frame and every reference. The extended Fu’s method consists of three steps: vectorization takes nT_V time, since it solves the linear least square problem n times; color transfer takes mnT_{V-OMT} time, since it computes an OMT map for every frame and every reference; PVG rendering takes mnT_R time.

Video	Size	T_{R-OMT}	T_V	T_{V-OMT}	T_{DC-M}	T_R
200×274	3262	3501	339	< 1	78
206×307	2858	3482	348	< 1	93
386×503	6640	5975	426	< 1	197
316×457	4876	4976	353	< 1	172
202×390	3412	3460	364	< 1	109
322×392	4622	4559	356	< 1	141
340×447	5640	5315	360	< 1	172

Table 6.1: Statistics. The timings were measured in milliseconds.

Our method also consists of three steps: vectorization takes only $T_V + (n - 1)T_{DC-M}$ time, since we only solve the linear square problem for the first frame; color transfer takes $mT_{V-OMT} + m(n - 1)T_{DC-M}$, since we compute the OMT map only for the first frame; PVG rendering takes mnT_R time. Since $T_{DC-M} \ll T_{R-OMT}$, $T_{DC-M} \ll T_{V-OMT}$ and $T_R \ll T_{R-OMT}$, our method has significant advantage for long videos (i.e., with large n) and multiple references (i.e., $m > 1$).

We evaluate our method on various face videos. Figures 6.5, 6.6 and 6.7 show our results on several celebrities with different skin colors.

Since our method does not require correspondence between the input video and the reference, it is flexible and robust to handle persons wearing glasses (see Figures 6.8 and 6.9). Figure 6.10 shows the color swap result in a video.

Now we examine the quality of the results. We observe that both our method and the image-based OMT obtain results of similar quality (which is hard to distinguish by naked eyes), but our method runs significantly faster. The extended Fu’s method has a similar run-time performance as ours for a single transfer, but their method produces results with flickering artifacts, since it computes the DC boundary color g separately and there is not guarantee of smoothness between adjacent frames. Our method does not

Fig. 6.5: Transferred results of a celebrity with tanned skin color. The person is transferred from 4 references. We observe that the transferred colors are highly consistent and show the first, the second, the middle and the last frames as the representative frames. Note that the first frame is computed via OMT and the subsequent frames are obtained via DC matching.

Fig. 6.6: Transferred results of a celebrity with fair skin color. The person is transferred from 4 references.

Fig. 6.7: Transferred results of a celebrity with dark skin color. The person is transferred from 4 references.

compute g for frames $i \geq 2$, instead it simply transfers the g from the previous frame. As a result, our method is not only stable but also efficient.

To quantitatively evaluate the quality, we compute the PSNR metric P for each frame of the original video and the transferred video. Then we measure their difference as shown in Fig. 6.11. We observe that the difference curve of our method is almost a constant, meaning that our result consistently mimics the input, differing only by a constant which is independent of the frame index. In contrast, we observe the extended Fu’s method has large changes frame by frame, implying that their result is not stable at all.

Fig. 6.8: Transferring face colors to person wearing glasses. We extract the facial features to form the diffusion curves and then compute their boundary colors by solving a linear least square for the first frame. DCs encode the low-frequency skin colors and PRs encode the high-frequency details (including glasses). Given a vectorized reference image (with or without wearing glasses), we only require to transfer the boundary colors of DCs and retain PRs unchanged.

6.6 Summary

We developed a simple yet effective method for automatically transferring face colors in videos. Taking advantage of PVG guided vectorization, our method computes only an optimal mass transport map for the first frame and then transfers the DC boundary colors for the subsequent frames by matching the DCs. Thanks to its information reuse

Fig. 6.9: The transferred results of persons wearing glasses.

Fig. 6.10: Face color swap. Row 1 shows two frames of the original video. Row 2 shows the results by swapping the boundary colors of DCs and keep the PRs unchanged. Row 3 shows the results by updating PRs with coefficients 0.8 (Lemon) and 1.2 (Cuomo), respectively. We observe that the results with updated PR coefficients have more natural contrast and highlight.

Fig. 6.11: Quality analysis. We transfer Reference A and Reference B to the first and second half of the input video and compute the difference of PSNR between the original video and the transferred video. The more constant the difference, the more consistence with the input video, hence the less dependence of the frame index and the less flickering artifacts. The x -axis shows the frame index and the y -axis shows the squared difference of PSNR between the original video and the transferred video. Both our curve (red) and the curve of the image-based OMT (green) are almost constant, indicating the comparable quality of the results. However, our method runs significantly faster. We observe that simply applying Fu’s method [3] to the video frames produces results with flickering artifacts. The large variation of the blue curve confirms Fu’s results are unstable. The PSNR deviations of our method, the image-based OMT and Fu’s method are 0.0656, 0.0838, and 0.6248, respectively.

nature, our method is highly desired in frequent color transfer from multiple references. Since our approach require no dense correspondence between the source video and the reference image/video, it is flexible and robust to handle faces with significantly different geometries and postures. Moreover, it also allows the user to edit the highlight and contrast by changing the coefficient of Poisson regions. Experimental results demonstrate the effectiveness of our method.

Limitations. Our method requires an accurate skin mask. In our implementation, we adopted Hsu et al.’s algorithm [91] for skin extraction. We observed that Hsu’s method works well if skin colors are clearly different from background. However, this method often computes wrong masks if the assumption does not hold. Figure 6.12 shows a failed example where skin and hair have similar colors. In the future, we tend to adopt state-of-the-art deep learning techniques (e.g., [93]) to improve the accuracy of skin extraction.

Fig. 6.12: A failed case due to wrong skin extraction. The skin color clustering algorithm computes wrong masks (green) which include part of her hair. As a result, skin colors are also transferred to the hair. The artifacts are highlighted in the red boxes.

Chapter 7

Conclusions & Future Work

7.1 Conclusions

Poisson vector graphics (PVG) is a powerful extension from the commonly-used Diffusion Curves (DC) by appending another two primitives — Poisson region (PR) and Poisson curve (PC). It is a beneficial graphic designing tool that extends Laplace’s equation to Poisson’s equation. DC suffers from C^{-1} continuity or C^0 . In comparison, because of PR’s C^1 continuity, PR is adequate to establish smooth shading features. Furthermore, PVG only takes the salient feature curves as DCs to express the low-frequency colors, and otherwise the high-frequency attributes are encoded in PRs. It results in respectively processing for the high-frequency and low-frequency information with great efficiency and flexibility. These beneficial characteristics of PVG motivate us to explore diverse applications promoting theoretical and applied development in relevant fields such as the applications presented in this thesis.

In the first part of the thesis, we propose a novel decoration technique working directly on arbitrary topological meshes guided by PVG primitives. DCs are leveraged to define height and color on 3D models as the boundary condition. Meanwhile, both PRs and PCs could produce the shading effect with offsetting Laplacians. By utilizing them together, we design a parameterization-free framework. It allows the users to create their own artistic decorations on 3D models directly, by drawing only a few sparse curves associated with PVG representation. In addition, to provide a more convenient control, we also allow local and global parameterization to create complex patterns such as repetitive patterns. In terms of the algorithms, we extend Poisson solver from planar domain to 3D meshes

with harmonic B-spline. Since harmonic B-splines support the local calculation, it is free to determine which locations of the model required to be computed. Comparing with the existing methods supported by DCs only, our method can synthesize geometric detailing including both colors and the smooth embossing/carving effects. In addition, thanks to the smoothness effect supported by PRs, we provide hierarchical level-of-detail editing on models. For our Poisson solver’s localization, our method lessens the computational load, while providing more versatility like disregarding computation for the invisible positions out of sight.

In the second part of the thesis, we present an effective but simply-implemented approach for transferring hair and skin color between portrait images. By conducting PVG vectorization to the source and reference images, the low frequency is expressed by sparse DCs containing the main colors, and the high frequency is presented by PRs including residual details. Optimal mass transportation is then leveraged to change DCs’ boundary colors between the source reference images. The experimental outcomes provide some results of color transfer on hair and portraits skin, but actually, color transfer of smaller regions like eyes or mouth is transferable in a similar way. Moreover, our approach can also be utilized to transfer colors for landscapes or other objects. Nonetheless, our method has limitations. If the source and reference images have massively distinct colors, the color transfer outcomes may be artificial, even though tuning their PRs can slightly enhance the results.

In the third part of the thesis, we developed an efficient method for automatically transferring face colors in videos. Taking advantage of PVG guided vectorization, our method computes only an optimal mass transport map for the first frame and then transfers the DC boundary colors for the subsequent frames by matching the DCs. Thanks to its information reuse nature, our method is highly desired in frequent color transfer from multiple references. Since our approach requires no dense correspondence between the source video and the reference image/video, it is flexible and robust to handle faces with significantly different geometries and postures. Moreover, it also allows the user to edit the highlight and contrast by changing the coefficient of PRs. Our method requires accurate skin masks. In our implementation, our adopted skin extraction algorithm works well if skin colors are clearly different from background. However, this method often

computes wrong masks if the assumption does not hold. Note that some more sophisticated algorithms of skin extraction could be integrated in our framework to improve the results.

7.2 Future Work

In this thesis, we introduce the details of applying PVG to various related fields by leveraging its different good properties. In fact, the content shown in this thesis is not enough to fully explore the value of PVG.

In recent related works, artificial intelligence techniques have become a research focus, especially in computer vision. And in the computer graphics field, it is also necessary to leverage learning based strategy to overcome the shortcomings of traditional methods or explore new application prospects. So far, the related works are largely deficient. So in the future, we intend to combine PVG and AI techniques to explore more valuable works, and improve the traditional heuristic algorithms. In our plan, we would like to use the deep neural networks to improve the traditional PVG vectorization at the first step. In the current process of PVG vectorization, we first perform edge extraction on the original images to obtain the probability of whether each pixel is an edge. If the probability is greater, it means the edge is more obvious. Then a threshold is specified to determine edges, serving as DC candidates. On both sides of these DC candidates, the color with the maximum gradient is adapted as the color of current DC candidate. And the pixel-level Laplacian (PR) can be calculated by the Laplace operator. Finally, we obtain PVG primitives, that DCs render the low-frequency colors and PRs present more other details.

The heuristic method can achieve acceptable vectorization results, but in some aspects, it still needs to be improved further. For example, for the DC extraction, the selection of threshold has a great influence on the quantity and quality of DCs. Now, for portraits, we give an empirical threshold to produce reasonable results, which is enough in our current color transfer framework. But for other goals, this empirical threshold may no longer meet the needs. Furthermore, it may be unacceptable to determine DCs in other applications by using the edge probability threshold alone. Sometimes, it cannot

guarantee the semantic integrity of DCs. For example, assuming that when a portrait is vectorized with PVG, if the color of face is similar to the background, the edge probability there is supposed to be too small to be distinguished as DCs. As a result, we cannot obtain a complete DC not only expressing the colors but also containing the facial contour.

Till now, we are trying to design an end-to-end network on the basis of deep learning techniques to improve the vectorization of PVG. It can not only simplify human intervention (such as the threshold setting) to realize a fully automatic vectorization system, but also ensure the semantic integrity of these PVG primitives, laying a foundation for its broader development prospects. Regarding to its applications, PVG vectorization utilizes sparse curves to express hue and tone separately, thus it can naturally be used for animation stylization. In addition, PVG is very easy to edit, which can also be applied to geometric editing, such as face deformation or expression editing, etc.

Specifically, the deep learning based PVG vectorization is the beginning of experimental stage. Figure 7.1 shows our preliminary design of the system structure. The system combines a face image processing network and an edge detection network to automatically extract features (i.e. DCs, PCs and PRs) for PVG vectorization. First, we are going to adopt SfsNet [94] to decompose face image into shape, albedo and shading. Then, taking the decomposed albedo and shading as the inputs, we are planning to extend DexiNed [95] to complete the tasks of extracting DCs (PCs) and fitting PRs.

Fig. 7.1: The overall architecture of our system.

References

- [1] Z. Shu, S. Hadap, E. Shechtman, K. Sunkavalli, S. Paris, and D. Samaras, “Portrait lighting transfer using a mass transport approach,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–15, 2018.
- [2] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand, “Style transfer for headshot portraits,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 148–171, 2014.
- [3] Q. Fu, Y. He, F. Hou, J. Zhang, A. Zeng, and Y.-J. Liu, “Vectorization based color transfer for portrait images,” *Computer-Aided Design*, vol. 115, pp. 111–121, 2019.
- [4] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, “Color transfer between images,” *IEEE Computer graphics and applications*, vol. 21, no. 5, pp. 34–41, 2001.
- [5] Y. Hwang, J.-Y. Lee, I. So Kweon, and S. Joo Kim, “Color transfer using probabilistic moving least squares,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3342–3349.
- [6] P. V. Sander, H. Hoppe, S. Gortler, and J. Snyder, “Signal-specialized parameterization,” in *13th Eurographics Workshop on Rendering*. Eurographics Association/Association for Computing Machinery, 2002, pp. 87–98.
- [7] M. Stamminger and G. Drettakis, “Perspective shadow maps,” in *ACM transactions on graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 557–562.
- [8] C. Dachsbacher and M. Stamminger, “Rendering procedural terrain by geometry image warping,” in *Rendering Techniques*, 2004, pp. 103–110.
- [9] S. Jeschke, D. Cline, and P. Wonka, “Rendering surface details with diffusion curves,” in *ACM SIGGRAPH Asia 2009 Papers*, ser. SIGGRAPH Asia ’09. New York, NY, USA: ACM, 2009, pp. 117:1–117:8.
- [10] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin, “Diffusion curves: A vector representation for smooth-shaded images,” *ACM Trans. Graph.*, vol. 27, no. 3, pp. 92:1–92:8, 2008.
- [11] Q. Fu, F. Hou, Q. Sun, S.-Q. Xin, Y.-J. Liu, W. Wang, H. Qin, and Y. He, “Decorating 3d models with poisson vector graphics,” *Computer-Aided Design*, vol. 102, pp. 1–11, 2018.
- [12] F. Hou, Q. Sun, Z. Fang, Y.-J. Liu, S.-M. Hu, H. Qin, A. Hao, and Y. He, “Poisson vector graphics (pvg) and its closed-form solver,” *arXiv:1701.04303*, 2017.
- [13] P. Feng and J. Warren, “Discrete bi-laplacians and biharmonic b-splines,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 115:1–115:11, Jul. 2012.
- [14] F. Hou, Y. He, H. Qin, and A. Hao, “Knot optimization for biharmonic b-splines on manifold triangle meshes,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 9, pp. 2082–2095, Sept 2017.
- [15] H. Bezerra, E. Eisemann, D. DeCarlo, and J. Thollot, “Diffusion constraints for vector graphics,” in *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR ’10. New York, NY, USA: ACM, 2010, pp. 35–42.
- [16] M. Finch, J. Snyder, and H. Hoppe, “Freeform vector graphics with controlled thin-plate splines,” *ACM Trans. Graph.*, vol. 30, no. 6, pp. 166:1–166:10, 2011.

REFERENCES

- [17] H. Lieng, F. P. Tasse, J. Kosinka, and N. A. Dodgson, “Shading curves: Vector-based drawing with explicit gradient control,” *Comput. Graph. Forum*, vol. 34, no. 6, pp. 228–239, 2015.
- [18] F. Hou, Q. Sun, Z. Fang, Y.-J. Liu, S.-M. Hu, H. Qin, A. Hao, and Y. He, “Poisson vector graphics (pvg),” *IEEE Transactions on Visualization and Computer Graphics*, p. 1, 2018. [Online]. Available: doi.ieeecomputersociety.org/10.1109/TVCG.2018.2867478
- [19] B. Price and W. Barrett, “Object-based vectorization for interactive image editing,” *The Visual Computer*, vol. 22, no. 9-11, pp. 661–670, 2006.
- [20] G. Lecot and B. Levy, “Ardeco: automatic region detection and conversion,” in *17th Eurographics Symposium on Rendering-EGSR’06*, 2006, pp. 349–360.
- [21] J. Sun, L. Liang, F. Wen, and H.-Y. Shum, “Image vectorization using optimized gradient meshes,” in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 11.
- [22] J. Ferguson, “Multivariable curve interpolation,” *Journal of the ACM (JACM)*, vol. 11, no. 2, pp. 221–228, 1964.
- [23] X. Yin, J. Dai, S. Yau, and X. Gu, “Slit map: Conformal parameterization for multiply connected surfaces,” in *Advances in Geometric Modeling and Processing, 5th International Conference, GMP 2008, Hangzhou, China, April 23-25, 2008. Proceedings*, 2008, pp. 410–422. [Online]. Available: https://doi.org/10.1007/978-3-540-79246-8_31
- [24] Y.-K. Lai, S.-M. Hu, and R. R. Martin, “Automatic and topology-preserving gradient mesh generation for image vectorization,” in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, pp. 85–93.
- [25] Y. Song, J. Wang, L.-Y. Wei, and W. Wang, “Vector regression functions for texture compression,” *ACM Trans. Graph.*, vol. 35, no. 1, pp. 5:1–5:10, Dec. 2015.
- [26] T. Dekel, C. Gan, D. Krishnan, C. Liu, and W. T. Freeman, “Sparse, smart contours to represent and edit images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3511–3520.
- [27] G. Xie, X. Sun, X. Tong, and D. Nowrouzezahrai, “Hierarchical diffusion curves for accurate automatic image vectorization,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, p. 230, 2014.
- [28] F. Pitie, A. C. Kokaram, and R. Dahiya, “N-dimensional probability density function transfer and its application to colour transfer,” in *Proceedings of ICCV ’05*. IEEE, 2005, pp. 1434–1439.
- [29] J.-Y. Lee, K. Sunkavalli, Z. Lin, X. Shen, and I. So Kweon, “Automatic content-aware color and tone stylization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2470–2478.
- [30] F. Wu, W. Dong, Y. Kong, X. Mei, J.-C. Paul, and X. Zhang, “Content-based colour transfer,” in *Computer Graphics Forum*, vol. 32, no. 1. Wiley Online Library, 2013, pp. 190–203.
- [31] Y. Yang, H. Zhao, L. You, R. Tu, X. Wu, and X. Jin, “Semantic portrait color transfer with internet images,” *Multimedia Tools and Applications*, vol. 76, no. 1, pp. 523–541, 2017.
- [32] C. Li, S. Lin, K. Zhou, and K. Ikeuchi, “Specular highlight removal in facial images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3107–3116.
- [33] H. Zhou, J. Sun, Y. Yacoob, and D. W. Jacobs, “Label denoising adversarial network (ldan) for inverse lighting of faces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6238–6247.
- [34] T. Sun, J. T. Barron, Y.-T. Tsai, Z. Xu, X. Yu, G. Fyffe, C. Rhemann, J. Busch, P. Debevec, and R. Ramamoorthi, “Single image portrait relighting,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 79, 2019.
- [35] O. Frigo, N. Sabater, V. Demoulin, and P. Hellier, “Optimal transportation for example-guided color transfer,” in *Asian Conference on Computer Vision*. Springer, 2014, pp. 655–670.
- [36] E. K. Ryu, Y. Chen, W. Li, and S. Osher, “Vector and matrix optimal mass transport: theory, algorithm, and applications,” *SIAM Journal on Scientific Computing*, vol. 40, no. 5, pp. A3675–A3698, 2018.
- [37] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

REFERENCES

- [38] J. Fišer, O. Jamriška, D. Simons, E. Shechtman, J. Lu, P. Asente, M. Lukáč, and D. Šykora, “Example-based synthesis of stylized facial animations,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 155, 2017.
- [39] F. Luan, S. Paris, E. Shechtman, and K. Bala, “Deep photo style transfer,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4990–4998.
- [40] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, “Visual attribute transfer through deep image analogy,” *arXiv preprint arXiv:1705.01088*, 2017.
- [41] M. He, J. Liao, D. Chen, L. Yuan, and P. V. Sander, “Progressive color transfer with dense semantic correspondences,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 2, p. 13, 2019.
- [42] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, “A closed-form solution to photorealistic image stylization,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 453–468.
- [43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [44] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” in *Advances in neural information processing systems*, 2016, pp. 469–477.
- [45] J. Choe, S. Park, K. Kim, J. Hyun Park, D. Kim, and H. Shim, “Face generation for low-shot learning using generative adversarial networks,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1940–1948.
- [46] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, “Cvae-gan: fine-grained image generation through asymmetric training,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2745–2754.
- [47] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.
- [48] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.
- [49] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, “Attgan: Facial attribute editing by only changing what you want,” *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5464–5478, 2019.
- [50] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
- [51] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, “Multimodal unsupervised image-to-image translation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 172–189.
- [52] M. Peercy, J. Airey, and B. Cabral, “Efficient bump mapping hardware,” in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 303–306.
- [53] N. L. Max, “Horizon mapping: shadows for bump-mapped surfaces,” *The Visual Computer*, vol. 4, no. 2, pp. 109–117, 1988.
- [54] M. J. Kilgard, “A practical and robust bump-mapping technique for today’s GPUs,” in *Game Developers Conference 2000*, 2000, pp. 1–39.
- [55] T. Kaneko, T. Takahei, M. Inami, N. Kawakami, Y. Yanagida, T. Maeda, and S. Tachi, “Detailed shape representation with parallax mapping,” in *Proceedings of ICAT*, vol. 2001, 2001, pp. 205–208.
- [56] N. Tatarchuk, “Dynamic parallax occlusion mapping with approximate soft shadows,” in *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 2006, pp. 63–69.
- [57] M. M. Oliveira, G. Bishop, and D. McAllister, “Relief texture mapping,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 359–368.
- [58] F. Policarpo, M. M. Oliveira, and J. L. Comba, “Real-time relief mapping on arbitrary polygonal surfaces,” in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. ACM, 2005, pp. 155–162.

REFERENCES

- [59] R. L. Cook, “Shade trees,” *ACM Siggraph Computer Graphics*, vol. 18, no. 3, pp. 223–231, 1984.
- [60] R. L. Cook, L. Carpenter, and E. Catmull, “The reyes image rendering architecture,” in *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 95–102.
- [61] L. Szirmay-Kalos and T. Umenhoffer, “Displacement mapping on the GPU—state of the art,” in *Computer Graphics Forum*, vol. 27, no. 6. Wiley Online Library, 2008, pp. 1567–1592.
- [62] G. Elber, “Geometric texture modeling,” *IEEE Computer Graphics and Applications*, vol. 25, no. 4, pp. 66–76, 2005.
- [63] Y.-K. Lai, S.-M. Hu, D. Gu, and R. R. Martin, “Geometric texture synthesis and transfer via geometry images,” in *Proceedings of the 2005 ACM symposium on Solid and physical modeling*. ACM, 2005, pp. 15–26.
- [64] P. Bhat, S. Ingram, and G. Turk, “Geometric texture synthesis by example,” in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, 2004, pp. 41–44.
- [65] K. Zhou, X. Huang, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.-Y. Shum, “Mesh quilting for geometric texture synthesis,” in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 690–697.
- [66] R. De Toledo, B. Wang, and B. Levy, “Geometry textures and applications,” in *Computer Graphics Forum*, vol. 27, no. 8. Wiley Online Library, 2008, pp. 2053–2065.
- [67] E. Landreneau and S. Schaefer, “Scales and scale-like structures,” *Comput. Graph. Forum*, vol. 29, no. 5, pp. 1653–1660, 2010.
- [68] N. A. Carr and J. C. Hart, “Painting detail,” in *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 845–852.
- [69] S. Jeschke, D. Cline, and P. Wonka, “A GPU Laplacian solver for diffusion curves and Poisson image editing,” *ACM Trans. Graph.*, vol. 28, no. 5, pp. 116:1–116:8, 2009.
- [70] S. Boyé, P. Barla, and G. Guennebaud, “A vectorial solver for free-form vector gradients,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 173, 2012.
- [71] Y. Lipman, R. M. Rustamov, and T. A. Funkhouser, “Biharmonic distance,” *ACM Transactions on Graphics (TOG)*, vol. 29, no. 3, p. 27, 2010.
- [72] E. Nava-Yazdani and K. Polthier, “De casteljau’s algorithm on manifolds,” *Computer Aided Geometric Design*, vol. 30, no. 7, pp. 722–732, 2013.
- [73] S.-Q. Xin, X. Ying, and Y. He, “Constant-time all-pairs geodesic distance query on triangle meshes,” in *Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games*. ACM, 2012, pp. 31–38.
- [74] S.-Q. Xin and G.-J. Wang, “Improving chen and han’s algorithm on the discrete geodesic problem,” *ACM Trans. Graph.*, vol. 28, no. 4, pp. 104:1–104:8, Sep. 2009.
- [75] Y.-J. Liu, Z. Chen, and K. Tang, “Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1502–1517, 2011.
- [76] N. Greene, M. Kass, and G. Miller, “Hierarchical z-buffer visibility,” in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’93. New York, NY, USA: ACM, 1993, pp. 231–238.
- [77] S. Kumar, D. Manocha, W. Garrett, and M. Lin, “Hierarchical back-face computation,” *Computers & Graphics*, vol. 23, no. 5, pp. 681 – 692, 1999.
- [78] P. Dollár and C. L. Zitnick, “Structured forests for fast edge detection,” in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ser. ICCV ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1841–1848.
- [79] Z. Li and J. Zhang, “Pixel-level guided face editing with fully convolution networks,” in *2017 IEEE International Conference on Multimedia and Expo, ICME 2017, Hong Kong, China, July 10-14, 2017*, 2017, pp. 307–312.

REFERENCES

- [80] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [81] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang, “Exemplar-based face parsing,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- [82] C. Villani, *Topics in optimal transportation*. American Mathematical Soc., 2003, no. 58.
- [83] G. Monge, “Mémoire sur la théorie des déblais et des remblais,” *Histoire de l’Académie Royale des Sciences de Paris*, p. 666–704, 1781.
- [84] J. Rabin, G. Peyré, J. Delon, and M. Bernot, “Wasserstein barycenter and its application to texture mixing,” in *International Conference on Scale Space and Variational Methods in Computer Vision*. Springer, 2011, pp. 435–446.
- [85] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister, “Blind video temporal consistency,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 196, 2015.
- [86] Q. Fu, Y. He, F. Hou, Q. Sun, A. Zeng, Z. Huang, J. Zhang, and Y.-J. Liu, “Poisson vector graphics (pvg)-guided face color transfer in videos,” *Pending minor revision, IEEE computer graphics and applications*, 2020.
- [87] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
- [88] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, “Wing loss for robust facial landmark localisation with convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2235–2245.
- [89] A. Shrivastava, A. Gupta, and R. Girshick, “Training region-based object detectors with online hard example mining,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.
- [90] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
- [91] R.-L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, “Face detection in color images,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 696–706, 2002.
- [92] M. Leordeanu, R. Sukthankar, and C. Sminchisescu, “Efficient closed-form solution to generalized boundary detection,” in *European Conference on Computer Vision*. Springer, 2012, pp. 516–529.
- [93] Y. He, J. Shi, C. Wang, H. Huang, J. Liu, G. Li, R. Liu, and J. Wang, “Semi-supervised skin detection by network with mutual guidance,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2111–2120.
- [94] S. Sengupta, A. Kanazawa, C. D. Castillo, and D. W. Jacobs, “Sfsnet: Learning shape, reflectance and illuminance of faces in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6296–6305.
- [95] X. S. Poma, E. Riba, and A. Sappa, “Dense extreme inception network: Towards a robust cnn model for edge detection,” in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1923–1932.