面向图像视频编辑的图层分解方法研究

(申请清华大学工学博士学位论文)

培 养 单 位： 计算机科学与技术系

学　　科： 计算机科学与技术

研　究　生： 杜 正 君

指 导 教 师： 徐　　昆　副教授

二〇二三年五月

Research on Layer Decomposition Methods for Image and Video Editing

Dissertation submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Doctor of Philosophy

in

Computer Science and Technology

by

Du Zhengjun

Dissertation Supervisor: Associate Professor Xu Kun

May, 2023

学位论文公开评阅人和答辩委员会名单

公开评阅人名单

张磊	教授	北京理工大学
张慧	副研究员	清华大学

答辩委员会名单

主席	陈宝权	教授	北京大学
委员	史元春	教授	清华大学
	张磊	教授	北京理工大学
	刘永进	教授	清华大学
	张松海	副教授	清华大学
	徐昆	副教授	清华大学
秘书	刘舫	助理研究员	清华大学

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》及上级教育主管部门具体要求，向国家图书馆报送相应的学位论文。

本人保证遵守上述规定。

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

图像视频涵盖现代生活的方方面面，相关编辑技术已经在平面设计、电视转播、游戏娱乐、航空航天、军事国防等诸多传统领域发挥了重要作用。近年来，随着数码相机、智能手机的快速普及和移动互联网的飞速发展，人们比以往任何时候都更容易采集、创作、分享和获取图像视频数据。与此同时，随着短视频平台的强势崛起，每天数以亿计的图像视频数据在互联网上快速传播。大量的应用场景和海量的数据都对图像视频的高效处理提出了新的要求和挑战。

为了实现高效的图像编辑，有经验的设计人员通常使用图层来组织和管理图像。利用这些图层，设计人员可以方便地添加文字、绘制图案、调整颜色以及制作各种有趣的特效。受此启发，本文重点研究图层分解技术。根据特定的应用场景，将输入图像分解为一组图层，并进一步通过这些图层实现高效的图像编辑。在图像视频处理领域，图层分解技术涵盖图像矢量化、图像重着色、图像抠图、本征分解等在内的多个应用场景。本文重点关注面向图像矢量化和图像重着色的图层分解技术，提出了一系列全新的算法对现有技术进行改进和拓展，具体包括：

1. 提出了一种面向图像矢量化的半透明图层分解算法。该算法将输入的光栅图像分解为一组空间上相互重叠的半透明矢量图层，同时这些图层可通过 alpha 混合高质量地重建出输入图像。与现有算法相比，在输入给定的情况下，本文算法实现了全自动的半透明图层分解。此外，本文算法分解的图层更好地反映了图像的形状和层次结构，从而也更加符合用户的编辑习惯。

2. 提出了一种面向图像内容感知重着色的加性图层分解算法。该算法将图像的低级颜色特征跟高级语义特征相结合，并在高维空间实施调色板提取和图层分解。相对于已有算法，本文算法分解的图层可以捕捉图像的语义信息，能够有效区分图像中颜色相似的不同物体。在此基础上，本文解决了这一领域长期存在的问题，实现了内容感知的颜色编辑。

3. 提出了一种面向视频重着色的时变图层分解算法。该算法首次将面向图像重着色的加性图层分解技术拓展到视频场景。针对给定的视频，算法输出一组随时间光滑渐变的图层和一个伴随的 4D 几何调色板。在视频重着色任务中，用户只需简单地修改 4D 几何调色板的颜色即可实现高效、自然的视频颜色编辑。

关键词：图像视频编辑；图层分解；图像矢量化；图像重着色；视频重着色

Abstract

Images and videos are ubiquitous in modern life. Related editing techniques have widely used in graphic design, television broadcasting, gaming and entertainment, aerospace, and military defense, etc. In recent years, with the rapid spread of digital cameras, smartphones, and the rapid development of mobile Internet, it is easier than ever for people to collect, create, share, and access images and videos. Meanwhile, with the substantial rise of short video platforms, hundreds of millions of photos and videos are rapidly spreading on the Internet every day. Many application scenarios and massive data put forward new requirements and challenges for efficient image and video processing.

To achieve efficient image editing, experienced designers often use layers to organize images. With the help of these layers, designers can easily add text, draw patterns, adjust colors, and create interesting effects. Inspired by this, this thesis focuses on layer decomposition techniques. According to specific application scenarios, the input image is decomposed into a set of layers, and these layers are further used to achieve efficient image editing. In image and video processing, the layer decomposition techniques cover several application scenarios, including image vectorization, image recoloring, image matting, and intrinsic decomposition. In this thesis, we focus on layer decomposition techniques for image vectorization and image recoloring. To summarize, the research in this thesis includes the following three aspects.

1. A semi-transparent layer decomposition algorithm for image vectorization is proposed. The algorithm decomposes the input raster image into a set of spatially overlapping semi-transparent layers. At the same time, these layers can reconstruct the input image with high quality after Alpha blending. Compared with existing methods, our approach achieves fully automatic layer decomposition with a given input. In addition, the layers generated by our method better reflect the shape of the image, so it is more in line with the user's editing habits.

2. An additive layer decomposition algorithm for image content aware recoloring is proposed. The algorithm combines low-level color features with high-level semantic features, and performs palette extraction and layer decomposition in high-dimensional space. Compared with state-of-the-art methods, the layers decomposed by our method can capture the semantic information of the image. They can effectively distinguish dif-

ferent objects with similar colors in the image. Based on this, we solve the long-standing problems in this field and achieve semantic-level, content-aware color editing.

3. A time-varying layer decomposition algorithm for video recoloring is proposed. This algorithm extends the layer decomposition algorithm for image recoloring to video scenes for the first time. The algorithm inputs a video and outputs a 4D geometric palette and a set of layers with smooth gradients over time. Applied to the video recoloring task, the user can modify the palette colors to achieve efficient and natural video color editing.

Keywords: Image and video editing; Layer decomposition; Image vectorization; Image recoloring; Video recoloring

目 录

插图清单

附表清单

符号和缩略语说明

RGB	RGB 颜色空间
RGBT	RGB 颜色与时间构成的四维空间
RGBXY	RGB 颜色与二维坐标构成的五维空间
Lab	Lab 颜色空间
LabXY	Lab 颜色与二维坐标构成的五维空间
Alpha	图像的半透明通道
MCTS	蒙特卡洛树搜索
RST	区域支持树
RAG	区域邻接图
DAG	有向无环图
SLIC	简单线性迭代聚类算法
PCA	主成分分析
EVD	特征值分解
SVD	奇异值分解
RBF	径向基函数
SLAM	即时定位与地图构建
RANSAC	随机抽样一致性算法
CRF	条件随机场
K-means	K-均值聚类算法
Bezier 曲线	贝塞尔曲线
Bezier 曲面	贝塞尔曲面
MVC	均值坐标
CNN	卷积神经网络
GAN	生成对抗网络
U-Net	U 型网络

第 1 章　引言

1.1　选题背景及意义

图像视频编辑涉及人们生产生活的方方面面，如图 1.1 所示，相关技术已经在平面设计、在线教学、电视转播、电影制作、动画特效、视频监控、短视频编辑等诸多领域发挥了至关重要的作用。近年来，随着智能手机、数码相机等图像采集设备的迅速普及和移动互联网的飞速发展，人们可以更方便地生成、创作、分享和获取图像视频数据。据统计，每天在 Facebook、Twitter、微信、QQ 等社交网络，以及 YouTube、Bilibili、抖音、快手等视频分享平台传播的图像和视频数以亿计！很多传统和新兴的应用场景都对图像视频编辑的有效性、高效性和便捷性提出了新的要求和挑战。

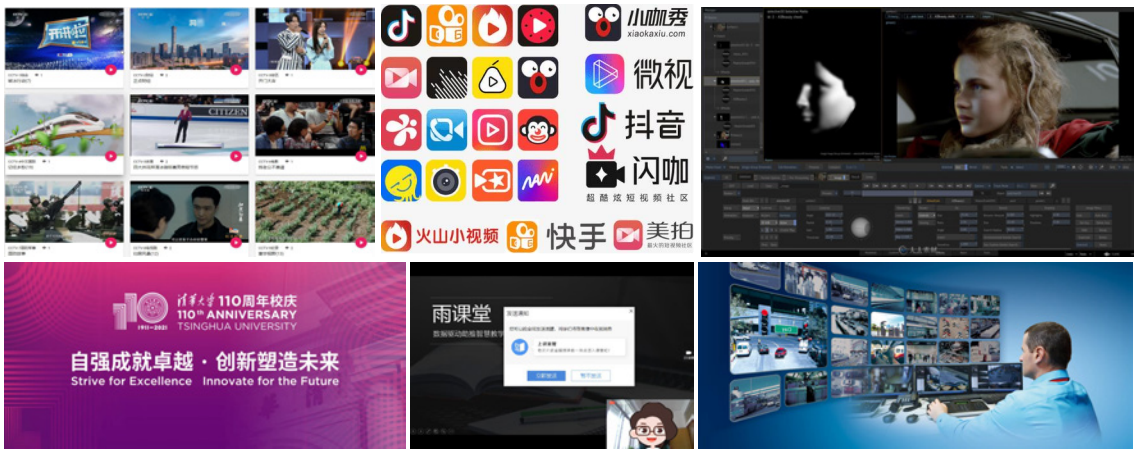


图 1.1　图像视频编辑的各种应用场景①

为了实现高效、便捷的图像编辑，平面设计师通常使用图层（Layer）来组织和管理图像。图层也是很多知名图像处理软件如 Adobe Photoshop，Adobe Illustrator，GNU Image Manipulation Program（GIMP），CorelDRAW，Inkscape 等的基础与核心，承载了包括颜色编辑、文字添加、图案绘制、特效制作等在内的所有基本操作。例如在 Adobe Photoshop 软件中，平面设计师就可以利用图层加入图像、文字、图形等各种元素，并通过某种混合方式生成最终的图像。如图 1.2 所示，左侧为

① 图像（从左到右，从上到下）分别来自于:
https://2021.tsinghua.edu.cn/xqbz.htm;
https://www.sohu.com/a/372197000_716256;
https://www.163.com/dy/article/EC7TU1HO0516BJGJ.html;
https://iptv.tsinghua.edu.cn;
https://www.sohu.com/a/555184254_120896862;
https://www.keneuc.cn/ruodian/6571.html。

Adobe Photoshop 的图层立体展示，中间为图层面板，右侧为合成图像。

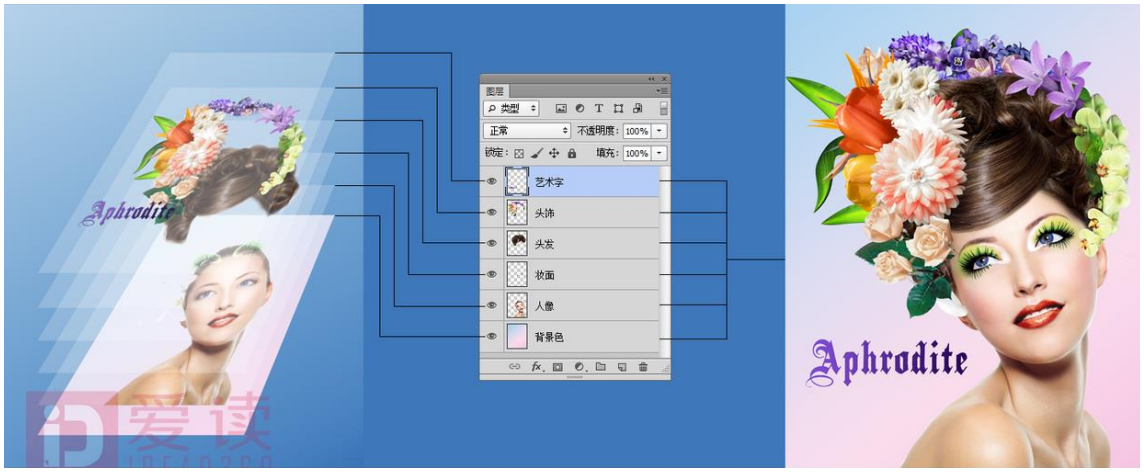


图 1.2　Adobe Photoshop 的图层示例①

图层的引入给图像编辑带来了巨大的便利。首先，图层之间相互独立，用户可以对某个图层进行单独地调整而不至于影响到其他无关的图层，方便用户对图像进行针对性的局部编辑。其次，用户可以通过调整图层的顺序、修改不透明度、增加或删除图层以实现一些精美的特效，降低了编辑的难度。最后，图层本身可以包含丰富的内容，比如图像的颜色、光照、材质、语义信息等，通过对这些属性的编辑可以实现更加逼真的效果，极大地提升了图像编辑的自由度和便捷性。

在图像处理软件中将图层合成为图像通常是比较容易的，但一旦将其导出为单张 JPG、PNG 或 BMP 等格式的位图，那么所有的图层信息都将会丢失，如同互联网上传播的大量自然图像一样，不再包含具体的图层数据。为了实现某些具体的编辑任务，针对单张图像的图层分解变得非常必要。因此，本文所关注的就是这样一个逆向工程问题：给定一张输入图像，根据具体的应用场景，分解出对应的图层，目的是通过分解到的图层实现高效的图像编辑任务。

接下来，本章首先对图层分解技术进行概述，然后介绍这一技术在相关应用中的研究现状、面临的挑战和存在的问题，最后引出本文的研究内容并进一步阐述主要的创新点和贡献。

1.2　图层分解技术概述

1.2.1　图层分解的概念

本小节首先简单地介绍图层的概念并给出图层合并的形式化定义，然后引出图层分解的概念。

① 图像来自于 http://www.iread360.com/article/info/3100。

图层 为了方便理解，本文涉及的图层可以简单地理解为一张图像。根据不同的应用场景，它可以是单通道的灰度图也可以是多通道的彩色图。

图层合成 给定一组图层 $L = \{L_1, L_2, \cdots, L_n\}$，通常可以通过某种特定的方式将其合成为单张图像 I：

$$I = f(L) \tag{1.1}$$

其中，$f(\cdot)$ 称为合成函数。在不同的应用场景，合成函数往往具有不同的表现形式。如在图像矢量化领域，合成函数通常表现为 Alpha 混合，而在本征图像分解领域，合成函数则表现为乘法混合（图层对应位置的像素值相乘）。

图层分解 图层分解就是逆向求解上述图层合成问题，即通过给定的输入图像 I，预测一组图层 $L = \{L_1, L_2, \cdots, L_n\}$，并且期望这些图层能够通过合成函数 f 完美地重建输入图像。

针对不同的应用场景，图层分解往往具有不同的表现形式。接下来，本文对几种常见的图层分解形式进行介绍。

1.2.2　图层分解的不同形式

根据不同的图层合并方式，本文将图层分解总结为四种形式：半透明图层分解、加性图层分解、时变图层分解和非线性图层分解。

1）**半透明图层分解** 半透明图层分解旨在将输入图像分解为一组空间上相互重叠的、有序的半透明图层 $L = \{L_1, L_2, \cdots, L_n\}$（$L_1$ 在底部，L_n 在顶部），并期望这些图层通过 Alpha 合成后跟输入图像尽可能接近。图层的 Alpha 混合定义如下：

$$I^k = \begin{cases} A_k C_k + (1 - A_k) I^{k-1}, & k > 0 \\ C_0, & k = 0 \end{cases} \tag{1.2}$$

半透明图层 L_k 是一张四通道的 RGBA 图像。其中，C_k 和 A_k 分别表示图层 L_k 的颜色和不透明度。I^k 表示前 k 个图层经 Alpha 合成后的三通道 RGB 图像。$I^0 = C_0$ 表示默认的画板（通常为白色且不透明），$I = I^n$ 表示所有图层经 Alpha 合成后的最终结果。半透明图层分解技术常用于图像矢量化和图像抠图等应用场景。

2）**加性图层分解** 加性图层通常将输入图像分解为一组顺序无关的图层，这些图层通过线性组合的方式重建输入图像：

$$I = \sum_{i=1}^{n} L_i C_i \tag{1.3}$$

这里的每个图层 L_i 是一个单通道的灰度图像，像素的取值范围为 $[0, 1]$。C_i 表示 L_i 对应的三通道 RGB 颜色。加性图层分解技术常用于图像重着色领域，在该领域，集合 $\{C_i\}$ 被称为调色板。顾名思义，用户可通过调色板控制图像的颜色变化。

3）**时变图层分解**　上述两种图层分解技术主要应用于图像编辑。针对视频场景，本文提出时变图层分解的概念。时变图层分解技术旨在将输入视频分解为一组随时间光滑渐变的图层，用以捕捉视频中颜色随时间的变化情况。这种技术可以自然的应用到视频重着色任务中，本文第 5 章将对这一技术进行详细阐述。需要注意地是，时变图层本质上属于加性图层，为了跟面向图像的加性图层分解技术区分开，这里单独给出时变图层分解的概念。

4）**非线性图层分解**　半透明图层分解通过 Alpha 混合生成最终的图像，加性图层分解通过线性加权生成最终的图像，二者可统一归纳为线性图层分解技术。除了这两种常见的图层分解方式之外，还有一大类复杂的、非线性的图层分解方法，本文统一将其称为非线性图层分解。这类图层分解方法主要用于光照调节、材质编辑、纹理替换等基于物理的高真实感图像编辑。

本文重点关注半透明图层分解技术和加性图层分解技术，并将图像矢量化和图像重着色作为主要的应用场景展开研究。接下来，本章首先介绍这两种图层分解技术在图像矢量化和图像重着色领域的研究现状和面临的挑战，然后导出本文的主要研究内容。

1.2.3　图层分解在图像矢量化中的应用

矢量图（Vector Graph）通过点、直线、曲线、多边形等基本几何图元描述图像的形状信息。相对于点阵表示的位图，矢量图在实际应用中具有存储空间小，易于编辑和无限缩放不失真等优点。图像矢量化（Image Vectorization）专门研究如何将光栅图或位图转换为矢量图，是图像处理的经典问题，相关技术在字体创作、Logo、广告、地图设计等领域得到了广泛应用。图像矢量化自提出以来已经走过了几十年的发展历程，一直是计算机图形学和图像视频处理领域较为活跃的研究方向。

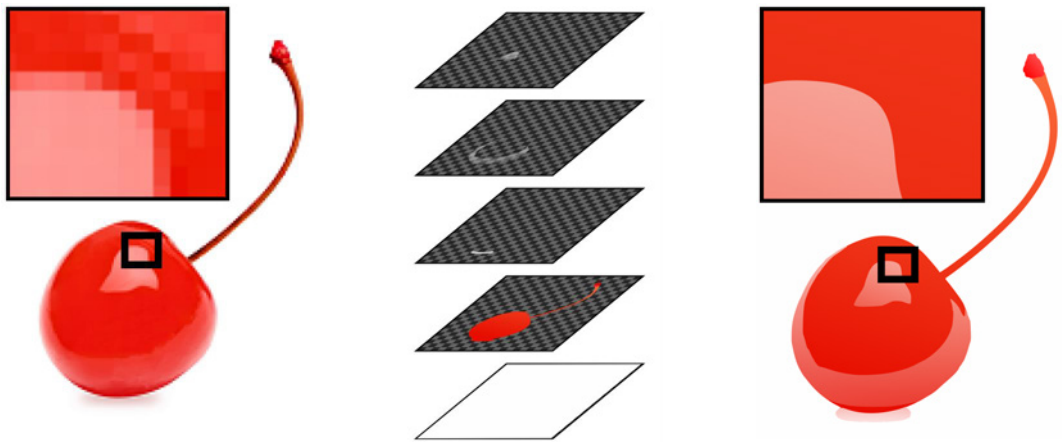
早期的图像矢量化算法如基于梯度网格[1-2]（Gradient Mesh）的算法，基于扩散曲线[3-6]（Diffusion Curve）的算法，基于 Patch[7]的算法等虽然实现了高质量的图像重建，但仍然存在一些缺陷。首先，这些算法对形状的描述和颜色的表达较为复杂，对于没有经验的用户来讲使用成本较高。其次，这类算法只能将图像矢量化为互不重叠的不透明区域。对于具有明显层次感或半透明堆叠效果的图像，这类算法分解的图层不能反映图像的层次结构，同时，产生的大量不透明区域将变得难以编辑和维护。

为了解决上述问题，近期的研究[8-9]提出基于半透明图层分解的图像矢量化方法。如图 1.3 所示，这类算法将输入的光栅图像分解成多个重叠的半透明图层，同时使用简单的线性渐变函数或径向渐变函数表达图层的颜色，最终的图像通过

Alpha 混合[10]（式（ 1.2））的方式合成。分解得到的半透明图层不仅可以精确地重建输入图像，同时使用简单的线性函数表示图像的颜色和不透明度，给用户编辑提供了巨大的便利。

面向图像矢量化的半透明图层分解技术是一个具有挑战性的问题。需要根据输入的光栅图像预测一组空间上顺序排列的图层。图层的数目，图层的堆叠顺序，图层的颜色和不透明度等都是待求解的未知量，而已知的只有输入的光栅图像。因此，这是一个严重欠约束的病态问题，理论上存在无穷解。另外，上述待求解的未知量既包含离散的参数（图层数目，堆叠顺序等）又包含连续的参数（表示颜色和不透明度的参数），难以统一优化求解。

现有算法[8-9]主要存在两个亟待改进的缺陷。首先，算法不够自动化，这类算法要么需要用户在输入图像中手动勾勒待分解的区域，要么需要用户指定不透明区域作为算法的初始值，一定程度上限制了算法的适用性。其次，分解的半透明图层的质量不高，不能直观地反映图像本身的形状结构，不利于后续编辑。



(a) 输入图像　　(b) 分解的半透明矢量图层　　(c) 图层合成结果

图 1.3　半透明图层分解在图像矢量化中的应用（图像来自于 Favreau 等[9]）

1.2.4　图层分解在图像重着色中的应用

颜色编辑是图像视频处理的核心任务，涵盖图像增强、编辑传播、风格迁移、图像彩色化、图像重着色等多个研究领域。图像重着色（Image Recoloring）是近年的热门研究方向，主要研究如何通过简单的交互高效地修改给定彩色图像的颜色，相关技术已经在艺术设计、图像美化、服装设计、广告、电影、短视频编辑等多个领域广泛应用。

经典的图像重着色方法通常分为两步即调色板提取和图层分解。本文涉及的调色板指的是一个包含少量代表性颜色的集合用以表示图像的主要颜色分布或模拟绘画颜料，目前主要通过聚类[11]或构建凸包[12]的方式从图像中提取。图层分

解是图像重着色技术的核心，旨在根据提取的调色板将输入图像分解为一组单通道的灰度图层（图层数量跟调色板颜色数目一致），并期望这些图层通过式（ 1.3）所示的方式高质量地重建出输入图像。图层分解的主要任务就是求解每个图层的逐像素的灰度值（介于 0~1 之间的实数），目前主要通过径向基函数（Radial Basis Function，RBF）插值，广义重心坐标[13]（Generalized Barycentric Coordinates）插值或能量优化的方式求得。图 1.4 给出了这类算法的一个示例，这里的灰度图就是分解得到的图层。实际的编辑过程中，用户只需简单地修改调色板的颜色就可以对图像颜色进行调整，编辑过程简单高效。例如这里将最下方的调色板颜色从红色修改为淡黄色，使得图像中的红色的苹果和草莓自然地变成了淡黄色。

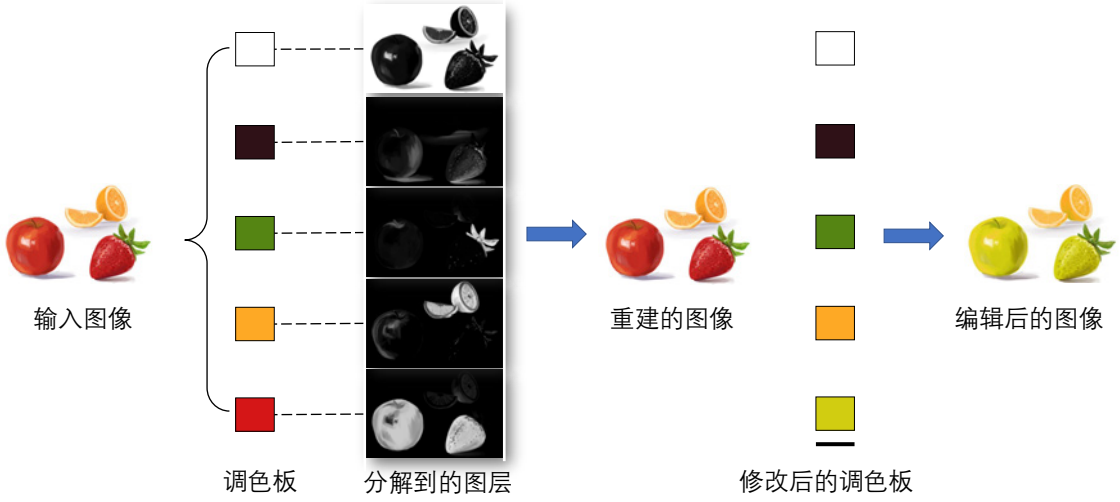


图 1.4　加性图层分解在图像重着色中的应用

作为图像重着色技术的重要基础，图层分解质量的好坏直接影响图像编辑的结果。本文认为图层分解的质量可以从以下四个方面进行考量。1）重建误差，分解的图层经合成后应当与输入图像尽可能接近。2）图层的光滑性，输入图像中颜色相似的像素在各个图层中应当具有相似的颜色值，否则可能导致重着色过程中出现颜色突变现象。3）图层的稀疏性，图层的稀疏性由图层中颜色值为 0 的像素数目决定，数目越多则越稀疏，较好的稀疏性才能保证良好的编辑局部性。否则，即使修改单一调色板颜色都可能导致大量的像素颜色发生改变。4）语义感知能力，图层应当具备一定的语义感知能力。如此一来，图像中的不同物体即使具有相似的颜色也能被有效地区分，方便用户实施针对性的物体级别的编辑。

现有技术已经在上述前三个方面取得了较好的进展。目前主要的缺陷在于分解的图层不包含语义信息。简而言之，若图像中存在多个颜色相似的不同物体，现有算法不能从语义上加以区分并分别着色。如图 1.4所示，当用户修改调色板中的红色时，苹果和草莓的颜色均发生了变化，无法只修改苹果或草莓的颜色。存在

这一问题的根源在于现有算法都在 RGB 颜色空间实施调色板提取和图层分解，本质上不可能捕捉图像的语义信息，因此无法做到语义级别的图像编辑。这是一个长期存在的问题，直到现在没有得到根本解决。

此外，尽管加性图层分解技术在图像重着色领域取得了很好的进展，但相关技术并没有拓展到视频场景。针对视频场景，加性图层分解面临的主要挑战在于图层难以捕捉随时间变化的颜色信息，因此，难以实现时变的视频颜色编辑。但无论如何，随着视频分享网站尤其短视频平台的快速发展，视频编辑具有广泛的应用前景，因此将加性图层分解技术拓展到视频场景并借此实现简单高效的视频编辑将具有重要的应用价值。

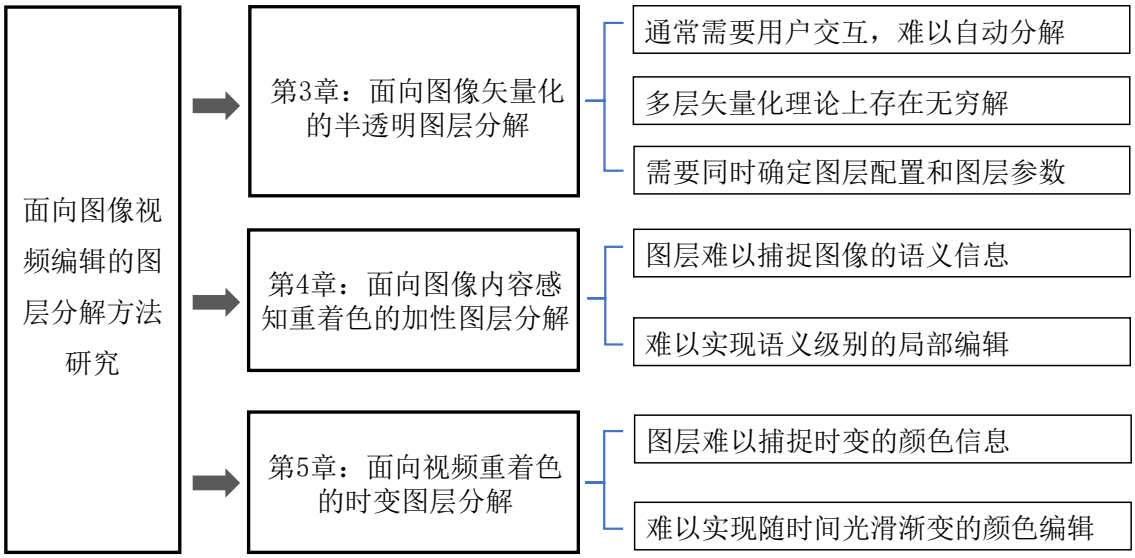


图 1.5　本文主要研究内容及面临的挑战

1.3　本文研究内容及创新点

图层分解技术在图像视频编辑中扮演了重要的角色，具有广泛的应用前景和重要的研究价值。本文重点关注半透明图层分解技术和加性图层分解技术，调研了这两种技术在图像矢量化和图像视频重着色领域的研究现状，面临的挑战以及存在的主要缺陷。为了解决现有技术存在的问题，本文提出了一系列方法对现有技术进行改进和扩展。图 1.5 给出了本文的主要研究内容和面临的挑战。具体地，本文研究内容和主要的创新点如下：

1. 提出了一种面向图像矢量化的半透明图层分解算法。主要的创新点在于：1）在给定输入的情况下，提出的算法全自动地将光栅图像分解为一组半透明矢量图层，整个过程完全无需用户交互，提升了算法的易用性；2）此外，本文还引入了一些基于感知理论的约束条件，在压缩搜索空间的同时排除了大量

不合理的图层分解结果；3）最后，相对于已有算法，本文算法分解的半透明图层具有更好的完整性（通常不包含孔洞），更能反映图像本身的形状结构，因此更加方便后续编辑。

2. 提出了一种面向图像内容感知重着色的加性图层分解算法。主要的创新点在于：1）将图像的低级特征跟高级语义特征相结合，并在高维空间实施图层分解，使得分解的图层能够有效地捕捉图像的语义信息；2）在此基础上，实现了语义级别的图像颜色编辑，解决了图像重着色领域长期存在的问题即无法对图像中颜色相似的不同物体有效区分并分别着色；3）提出的方法简单易用，为用户实施针对性的局部编辑提供了便利。
3. 提出了一种面向视频重着色的时变图层分解算法。主要的创新点在于：1）在 RGBT 四维空间对视频进行调色板提取和图层分解，分解的图层在时间轴上很好地反映了视频颜色的光滑渐变；2）在此基础上，成功地将图层分解技术应用到视频颜色编辑场景，实现了直观、自然的重着色效果，推动了加性图层分解技术的发展。

本文三个研究内容的主要联系在于：1）它们都将输入图像或视频分解为一组图层，且分解的图层都可以通过某种确定的混合方式高质量地重建输入图像或视频；2）交互方式简单，都通过对图层的简单操作实现高效的图像视频编辑。本文三个研究内容的主要区别在于：1）第 3 章提出的半透明图层是顺序相关的而第 4 章提出的加性图层和第 5 章提出的时变图层都是顺序无关的；2）半透明图层是矢量图层而加性图层和时变图层都是位图图层。总体来讲，本文首先研究顺序相关的图层分解技术，然后研究顺序无关的图层分解技术，并将应用场景从图像编辑扩展到视频编辑。

1.4　本文组织结构

本文后续章节安排如下：第 2 章对图层分解技术，图像矢量化及图像重着色领域的相关研究工作进行回顾、总结和分析；第 3 章对本文提出的面向图像矢量化的半透明图层分解算法的相关背景、问题定义、技术细节等进行详细的描述；第 4 章对本文提出的面向图像内容感知重着色的加性图层分解算法的研究动机、基本原理、算法细节、实验等进行详细的描述；第 5 章对本文提出的面向视频重着色的时变图层分解算法的设计思路、技术细节、实验设计等进行详细的描述；第 6 章对本文内容进行全面总结，并对未来的研究方向进行展望。

第 2 章　相关工作

本章首先对几种常见图层分解技术进行综述，然后对图像矢量化和图像重着色两个研究领域的其它相关工作进行调研、回顾和总结。

2.1　图层分解相关工作

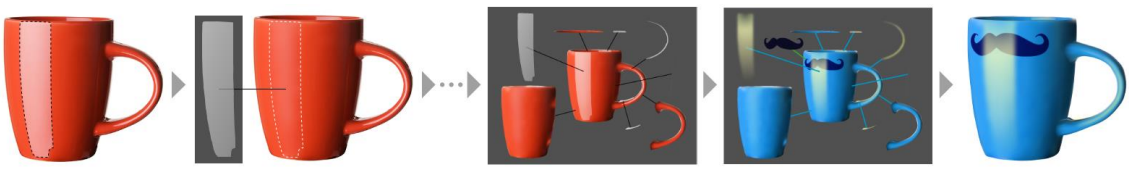
半透明图层分解

半透明图层分解旨在将输入图像分解为一组自底向上相互重叠的半透明图层：$L=\{L_1, L_2, ...L_n\}$。其中，L_1 位于底部，L_n 位于顶部。且要求这些图层通过 Alpha 混合（式（ 1.2））后跟输入图像尽可能接近。根据输入图像 I 逆向预测多个半透明图层是一个挑战性的问题。需要同时估测图层的数目以及各个图层的颜色跟不透明度，而已知条件只有输入图像，因此这是一个严重欠约束的病态问题。

Richardt 等[8]首次提出一种基于用户交互的算法将输入图像分解为多个半透明矢量图层。该算法将半透明图层分解问题形式化为一个前景图层分离问题：

$$I = AF + (1 - A)B \tag{2.1}$$

其中，F 和 A 分别表示前景图层的颜色和不透明度，B 表示背景部分的颜色。如果输入图像由多个图层构成，那么算法可以迭代地通过不断分离前景图层的方法得到构成该区域的所有半透明图层。另外，为了方便编辑，作者将每个图层矢量化，即通过线性渐变或径向渐变的函数拟合图层的颜色和不透明度。图 2.1 展示了该算法的一个示例。首先由用户在输入图像中选定待分解的区域（图 2.1(a)），然后将前景图层分离（图 2.1(b)），图 2.1(c) 展示了所有分解出的图层，最后两列（图 2.1(c,d)）展示了编辑后的图层及合成后的结果。



(a) 输入图像　(b) 分解的单个区域　(c) 所有图层　(d) 编辑后的图层　(e)合成效果

图 2.1　半透明图层分解算法示例 1（图像来自于 Richardt 等[8]）

上述算法的主要问题在于其依赖大量的用户交互，需要用户指定所有待分解图层的边界轮廓，并指定算法的迭代次数以确定每个区域将被分解为多少个图层。因此整个算法将非常耗时，无法做到自动化的图层分解。

为了解决上述问题，Favreau 等[9]提出一种只需少量用户交互的半透明图层分解方案。算法的目标是从输入图像中估计出多个半透明的矢量图层。其中每个图层包括三个属性：1）图层的覆盖范围；2）表示颜色的线性渐变参数；3）表示不透明度的线性渐变参数。为了简化问题，这里假设每个图层的颜色和不透明度共享同一渐变方向。半透明图层分解本质上是一个欠约束的病态问题，因此对于同一个输入图像可能存在多个满足条件的图层分解方案。为了衡量不同分解方案的质量，作者给出了如下的能量函数：

$$U(\mathbf{x}) = (1-\lambda)U_{\text{fidelity}}(\mathbf{x}) + \lambda U_{\text{simplicity}}(\mathbf{x}) \tag{2.2}$$

其中，$\mathbf{x}$ 表示一种图层配置，$U_{\text{fidelity}}(\mathbf{x})$ 表示重建误差即合成图像跟输入图像的差异，$U_{\text{simplicity}}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{n} w_i$ 用于度量所有图层的简化性（若 L_i 为半透明图层，$w_i = 1.0$；若 L_i 为不透明图层，$w_i = 1.2$）。也就是说，重建误差越小，图层不透明度越低的图层配置被认为是更优的。寻找上述能量函数的最优解是一个复杂的优化问题，需要同时确定：1）包括图层数目、层叠顺序、图层覆盖范围等在内的离散参数；2）表示图层颜色和不透明度的线性渐变的连续参数。作者将上述问题形式化为一个树搜索的问题，将每一种可能的图层分解方案跟一棵树进行关联，并通过蒙特卡洛树搜索[14]（Monte Carlo Tree Search，MCTS）的方式减小搜索空间。为了降低求解难度，算法要求用户提供输入图像的区域分割结果作为额外的输入，并要求用户指定不透明区域对算法进行初始化。图 2.2 展示了该算法的一个示例，其中，图 2.2(a) 右上角的分割图像中的黑色圆点就是用户提供的额外输入，图 2.2 (b) 展示了所有分解的图层，图 2.2 (c) 展示了两种编辑结果。

(a) 输入图像及分割区域　　(b) 所有分解的半透明和不透明图层　　(c) 编辑效果

图 2.2　半透明图层分解算法示例 2（图像来自于 Favreau 等[9]）

半透明图层分解算法在图像矢量化领域取得了不错的进展。分解的半透明图层通过简单的线性渐变函数进行拟合，整体上易于维护和编辑。算法最主要的缺陷在于其或多或少依赖用户交互，使得算法的适用性受到一定影响。本文将在第三章提出一种全自动的图像矢量化算法来解决这一问题，提出的算法跟 Favreau 等[9]的算法类似，但在给定输入的情况下，完全无需用户交互。

抠图（Image Matting）[15]是图像处理领域另一个经典的研究课题，主要研究如何将图像中的前景图层和背景图层进行有效分离，在图像视频合成、电影制作等领域广泛应用。抠图问题本质上是求解式（ 2.1）所示的逆向问题，即通过输入图像 $\boldsymbol{I}$ 预测前景图层 $\boldsymbol{F}$ 和背景图层 $\boldsymbol{B}$。与上述多图层矢量化问题不同的是，抠图问题假设图像只包含一个前景图层和一个背景图层，同时图层不需要矢量表示。

(a) 输入图像　(b) 三区图　(c) 前景不透明度　(d) 前景图层　(e) 合成效果

图 2.3　图像抠图及其在图像合成中的应用（图像来自于 Wang 等[16]）

为了有效地求解该问题，传统方法往往借助三区图（Trimap）作为辅助输入。如图 2.3 所示，三区图将整个输入图像分为三个部分即确定的前景区域（$A^p = 1$）、确定的背景区域（$A^p = 0$）和前景背景混合的未知区域（$0 < A^p < 1$）。如此一来，只需要对未知区域进行求解，一定程度上降低了问题的求解规模。总体上，传统方法分为两类：一类是基于采样的方法[17-24]，这类方法通过采样确定前景和背景区域的像素来估计未知区域的不透明度；另一类是基于传播的方法[25-28]，这类方法假设图像的局部区域满足平滑性，从而构建关于已知区域和未知区域的能量函数，并通过优化求解前景图层的颜色和不透明度。传统方法通常依赖用户交互来提供额外的先验知识，增加了用户的编辑负担。

近年来，随着机器学习的快速发展和训练数据的不断丰富，很多基于深度学习的抠图算法如基于卷积神经网络（Convolutional Neural Networks，CNN）的深度抠图技术[29-34]，基于生成对抗网络（Generative Adversarial Networks，GAN）的抠图技术[35-36]，基于 Transformer 的抠图技术[37-38]等陆续被提出，较传统方法在易用性和精确度上都有了很大提升，但在泛化能力和鲁棒性上仍存在一些问题。

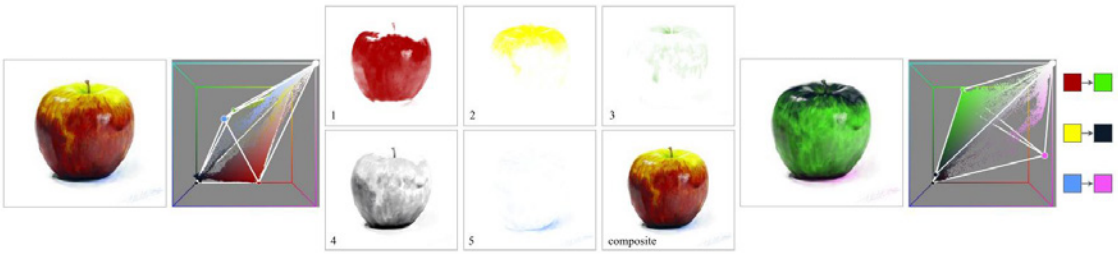
抠图问题自提出以来，已经走过三十多年的发展历程，得到了研究者的广泛关注。针对抠图问题，网站 Alpha Matting Evaluation Website①收录了近 70 种算法，并对这些算法的性能进行了详细的评测，更多研究进展和评测信息请参见该网站。

Tan 等[12]首次提出了基于 RGB 凸包的半透明图层分解算法，并将其应用到图像重着色任务中，实现了直观、自然的图像颜色编辑。跟 Richardt 等[8]等和 Favreau 等[9]等提出的半透明图层分解算法不同的是，Tan 等[12]分解的图层是位图而不是矢量图。相同的是，这里的图层也是有序的且都通过 Alpha 混合生成最终的图像。

① Alpha Matting Evaluation Website 网址：http://www.alphamatting.com/。

算法首先将图像投影到 RGB 空间，将整个图像视为 RGB 空间的一个三维点集。然后求解该点集的凸包并通过线性优化的方式对初始凸包进行迭代地简化，当简化后的凸包顶点数目等于用户预设的数值时算法停止。接下来，进一步根据获取的简化凸包确定图层的数量，图层的颜色和不透明度。具体而言，图层的数量跟凸包顶点的数目一致，且每个图层的颜色跟一个凸包顶点的坐标一一对应，为了消除歧义性，图层的顺序由用户手动指定。最后，根据简化凸包、给定的图层顺序以及图层的颜色通过能量优化或广义重心坐标插值的方式确定每个图层逐像素的不透明度。

根据算法特点不难发现，上述方法得到的任意图层中的所有像素颜色一致但透明度各异。为此，可以通过简单地修改各图层的颜色来调整合成图像的颜色。在重着色任务中，通常将图层的颜色集合称为调色板，用户通过修改调色板的颜色来控制图像的颜色变化。图 2.4 展示了该算法的一个示例，图2.4(a) 为输入图像和获取的 RGB 空间下的简化凸包（调色板）；图2.4(b) 展示了分解的 5 个半透明图层及合成结果，图层左下角的数字表示了图层的层叠顺序（1 表示最底部，5 表示最顶部）；图2.4(c) 展示了调色板的编辑过程和重着色结果。



(a) 输入图像及简化凸包　　(b) 分解的图层及合成效果　　(c) 编辑后图像及调色板

图 2.4　Tan 等[12]提出的图层分解算法示例（图像来自于 Tan 等[12]）

加性图层分解

跟半透明图层不同的是，加性图层通常是顺序无关的且合成方式更加简单，只需将各个图层线性求和即可。

注意到 Tan 等[12]提出的方法分解的半透明图层是有顺序的，这种图层适用于恢复某些使用颜料或水粉画创作的艺术绘画的颜料和对应的图层。但是很多自然图像并非通过 Alpha 混合而成，因此对自然图像进行图层分解的时候不必关注图层的顺序，可以使用更简单的加性混合方式（式（ 1.3））来合成图像。

为此，Tan 等[39]紧接着提出一种基于凸包的顺序无关的图层分解方法。该算法首先在 RGB 空间计算图像的凸包并简化，跟 Tan 等[12]的方法不同的是，这里通过重建误差阈值给出凸包简化的停止条件，因此凸包简化过程可以自动进行。类

似地，令图层数量跟简化凸包的顶点数目一致，且每个图层的颜色跟凸包顶点相关联。在图层分解阶段，为了考虑图层不透明度的空间连续性，作者首先将图像投影到 RGBXY 空间，求解图像的 5D 凸包并对像素进行插值，然后通过图像的 RGB 凸包对 5D 凸包顶点（只取 RGB 颜色数值）插值。通过两阶段插值，将图像像素表示为 RGB 凸包顶点的线性组合。最后，从插值结果中导出各个半透明图层的不透明度。类似地，将各图层的颜色（凸包顶点）集合作为调色板提供给用户，通过修改调色板颜色调整图像的颜色。

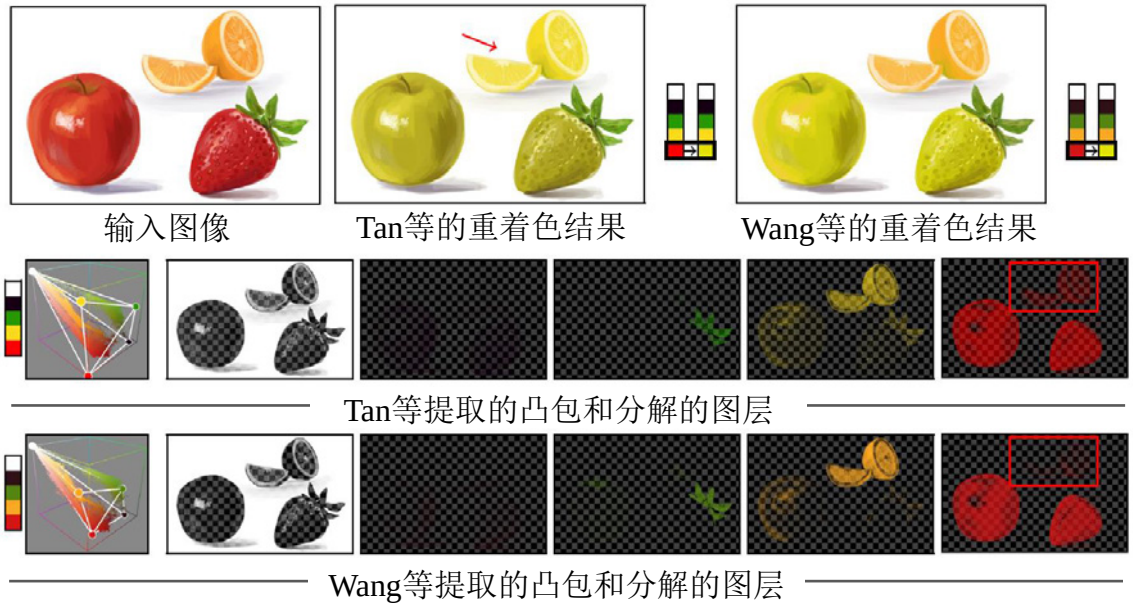


图 2.5　Wang 等[40]跟 Tan 等[39]的重着色算法对比（图像来自于 Wang 等[40]）

基于凸包的图层分解算法存在的主要问题在于简化后的凸包体积较大，不能在 RGB 空间紧致地包裹图像像素，使得凸包顶点对应的颜色（调色板）严重偏离图像的真实颜色，失去了代表性，导致分解的图层稀疏性较差。在颜色编辑过程中，修改单个调色板的颜色可能导致大范围的颜色改变。为此，Wang 等[40]对这一算法进行了改进，对简化凸包进行了优化，使得优化后的调色板具有更好的代表性，从而使得分解的图层具有更好的稀疏性，实现了更好的局部编辑。图 2.5 展示了 Wang 等[40]提出的算法跟 Tan 等[39]算法的对比。其中，第一行从左到右依次为输入图像，Tan 等的重着色结果，Wang 等的重着色结果；第二行展示了 Tan 等提取的凸包和分解的图层，第三行展示了 Wang 等提取的凸包和分解的图层。可以发现 Wang 计算的凸包的紧致性更好，在颜色编辑过程中局部性更好，修改调色板中的红色时只影响到输入图像中红色的苹果和草莓，很好地保留了橘子的颜色，而 Tan 的算法使得橘子的颜色发生了不自然的变化。

Chang 等[11]提出基于聚类的图层分解算法，并将其应用到重着色任务中，实

现了亮度感知的颜色编辑。该算法整体上分为两步即调色板提取和图层分解。在调色板提取阶段，算法首先将图像投影到 RGB 空间（将图像看作 RGB 空间的三维点集），然后使用 K-means[41]算法对采样点进行聚类，最后将聚类中心作为图像调色板，表示图像的代表性颜色分布。在图层分解阶段，算法对每个调色板颜色定义相似度函数，并通过径向基函数进行拟合。然后，计算图像中每个像素到调色板颜色的相似度实现图层分解。类似地，在重着色过程中，用户通过修改调色板控制图像的颜色变化。

图 2.6展示了该算法提取的调色板、分解的图层以及几个图像重着色结果。第二行图像下方展示的是修改后的调色板（下划线标记了修改的颜色）。

图 2.6　Chang 等[11]提出的重着色算法示例（图像来自于 Chang 等[11]）

Zhang 等[42]对上述图层分解算法进行了改进。该方法首先通过改进的 K-means 算法提取图像调色板，然后通过能量优化的方式求解图像像素关于调色板颜色的插值权重，特别地，能量函数加入了稀疏性约束，使得分解的图层具有良好的稀疏性。在重着色过程中，对于细节特征保持方面较之前的方法具有更好的视觉效果。

非线性图层分解

本征图层分解（Intrinsic Layer Decomposition）[43-50] 是一种典型的非线性图层分解技术。如图 2.7 所示，该技术旨在将图像分解为一个反射图层（Reflectance Layer）和一个光照图层（Shading Layer）。其中，反射图层反映物体在不同光照条件下依然保持不变的信息，表示物体的材质、颜色等本质属性；光照图层则反映不同的环境光照信息。图像一经分解，用户便可以对材质和光照分别进行编辑。输入图像与反射图层和光照图层的关系一般通过如下公式进行建模：

$$I^p = R^p S^p \tag{2.3}$$

即输入图像中的任意像素 I^p 可以表示为反射图层和光照图层中对应像素 R^p 和 S^p 的乘积。类似于抠图问题，本征图层分解问题本质上也是一个欠约束的病态问题。

(a) 输入图像　　(b) 反射图层　　(c) 光照图层

图 2.7　本征图层分解示例（图像来自于 Chen 等[44]）

除了常见的 Alpha 混合模式之外，平面设计师还经常使用一些高级颜色混合模式制作一些有趣的图像特效。为了满足用户对于图像特效制作的各种需求，W3C 最新标准提供了多达 16 种图层混合模式①。需要注意的是，除了常见的 Alpha 颜色混合模式（normal），该标准提供的绝大多数混合模式如 multiply，darken，screen，lighten 等都是非线性的，也就是说，最终的图像不是通过各图层线性组合而成的。相对于常见的 Alpha 混合模式，非线性颜色混合模式在阴影效果、对比度和光照的特效制作方面具有天然的优势。

近年来，针对非线性混合模式的多图层分解技术得到了一些研究者的关注。Koyama 等[51]提出一种非线性图层分解算法，该算法可以根据用户指定的图层数目和每个图层对应的颜色混合模式（如 color-burn，multiply，hard-light 等）通过优化的方式自动地分解出对应的非线性图层。Horita 等[52]提出了一种基于神经网络的非线性图层分解算法，相对于 Koyama 等人的算法，在计算效率和重建精度两方面都取得了较大的提升。

作为主要的绘画类型，油画和水彩画以植物油和水彩作为主要颜料进行作画，在艺术表现上具有厚重、细腻、层次感强等诸多优点。近几年，一些研究致力于恢复油画和水彩画的颜料及对应的图层，目的是实现颜料空间高真实感的图像编辑。代表性的工作为 Aharoni-Mack 等[53]提出的针对油画、水彩画的图层分解和调色板提取技术，该技术使用物理感知的 Kubelka-Munk 模型[54]对颜料的材质进行建模，通过优化的方式求解主要绘画颜料的材质参数，并进一步提取颜料对应的图层。Tan 等[55]对这一技术进行了改进，针对待求的绘画颜料采样了更多的波长，使用提取的颜料和对应的图层重建输入图像得到了更小的重建误差，图 2.8 展示了该方法的一个示例。

① Compositing and Blending Level 1 网址：https://drafts.fxtf.org/compositing-1。

(a) 输入图像 (b) 重建图像　　(c) 提取的颜料及对应的图层

图 2.8　基于油画、水彩画的图层分解示例（图像来自于 Tan 等[55]）

2.2　图像矢量化相关工作

位图和矢量图是两种主要的图像形式。位图通过点阵表示，存储结构简单，在颜色表示上更加细腻，能够精确表达图像的色调和色彩变化。矢量图通过点、直线、曲线、多边形等基本图元表示，相对于位图其优点主要体现在：1）占用存储空间更小；2）分辨率无关，随意放大缩小图像不失真；3）支持针对图像形状和颜色的精确快速编辑。因此，矢量图在 LOGO、字体、地图、网页、工程图纸设计等多个领域广泛应用。图像矢量化是计算机图形学和图像处理领域长期关注的热点，在近半个世纪的发展中涌现了大量优秀的研究成果。

基于半透明图层分解的图像矢量化方法如 Richardt 等[8]提出的基于用户交互的算法和 Favreau 等[9]提出的基于蒙特卡洛树搜索的方法已经在半透明图层分解部分做了详细的介绍，这部分主要对这一领域的其他经典工作进行分类综述。

基于样条曲线的图像矢量化方法

在计算机图形学和计算机辅助设计领域，样条曲线曲面被广泛应用于自然物体的形状建模，常见的样条曲线如 B 样条[56-61]，Bezier 样条[62-64]是几何建模领域的重要数学工具。在图像矢量化领域，基于样条的技术也得到了广泛的关注。

Lecot 和 Levy [65] 提出了一种自动化的图像矢量化方法“Ardeco”，该算法首先通过能量优化的方式将图像分割为多个区域，然后使用三次样条曲线表示这些区域的边界，并通过常量、一次、二次或三次函数拟合区域内的颜色。对于简单的图像而言，该算法可以实现令人满意的矢量化结果，但对于颜色较为丰富或细节较多的人脸矢量化时，仍需要额外的用户交互来提升精度，此外，该算法的能量优化过程比较耗时，在速度上具有一定的劣势。

Xia 等[7]提出分片的图像矢量化方法，该方法首先检测图像中明显的曲线边界，然后对图像进行三角剖分，接下来利用 Bezier 曲线表示的曲边三角形替换所有的三角面片，最后使用薄板样条（Thin-plate Splines ）拟合每一个分片的颜色。

Chen 等[66]提出实时的图像矢量化算法，该算法使用参数化的分片表示物体

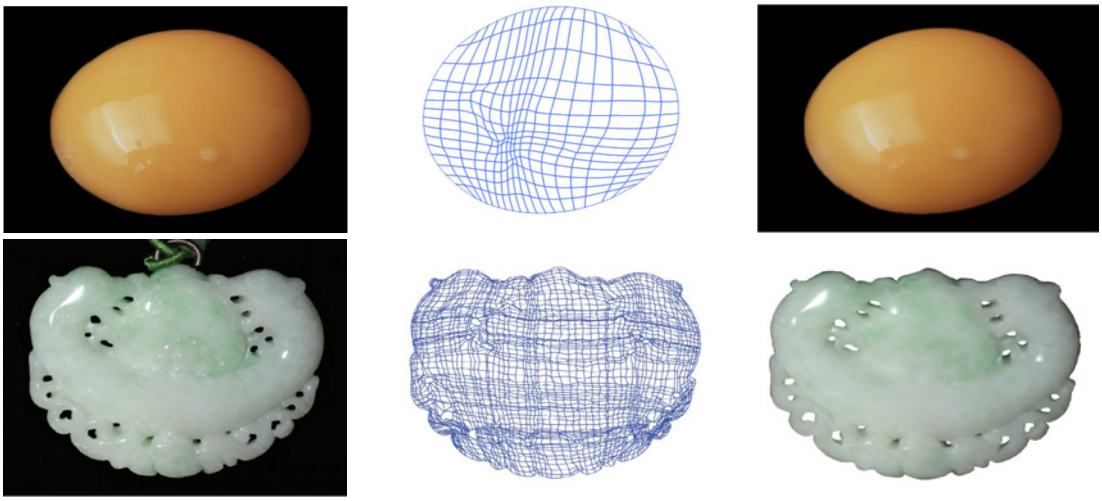
的轮廓，并使用薄板样条表示图像的颜色细节，同时使用两种混合的表示形式，不仅保证了颜色的精确拟合而且实现了图像形状的可编辑性。

Zhu 等[67]提出一种基于 TCB 样条的自动化的矢量化算法，该方法使用 TCB 样条迭代地优化图像的颜色和位置。该方法主要的优势在于其能够拟合任意形状的拓扑并且拟合的颜色具有很好的光滑性。

基于样条的图像矢量化算法在边界表示以及颜色的拟合上具有天然的优势，能够在视觉上最大程度地重建输入图像。但这类图像矢量化方法通常将图像分解为大量区域，且区域的颜色跟边界表示都相对复杂，对于非专业人员来讲编辑过程过于复杂，难以上手。

基于梯度网格的图像矢量化方法

梯度网格是图像处理软件 Adobe Illustrator 的核心工具，不仅容易编辑而且能很好地模拟光滑渐变的光照效果。注意到梯度网格的强大表达能力，Sun 等[1]首次将这种技术应用到图像矢量化领域，提出基于优化梯度网格的图像矢量化方法。梯度网格总体上是一个纵横交织的网状结构，每一个网格使用一个弗格森面片[68]（Ferguson Path）来表示。每个弗格森面片由四条边组成，而每条边则通过一条或多条三次 Bezier 曲线拟合而成。这种形式表达方式提供了较大的编辑自由度，用户可以通过修改网格顶点的位置、导数、颜色参数实现不同的视觉效果。该方法主要有两方面的缺陷：一是初始化阶段需要用户手动地对图像中的物体进行分割以建立初始的网格，加重了用户的使用负担；二是该方法不能有效处理物体边界包含孔洞等一些复杂的拓扑结构，限制了该方法的适用场景。



(a) 输入图像　(b) 梯度网格　(c) 矢量化结果

图 2.9　Sun 等[1]（第一行）和 Lai 等[2]（第二行）矢量化算法比较

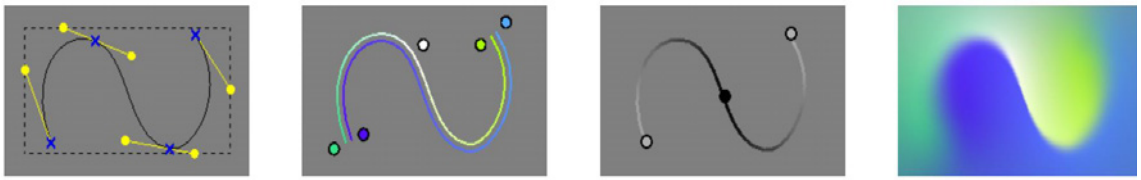
为了解决优化梯度网格无法处理孔洞拓扑的问题，Lai 等[2]提出了一种能够适

应任意拓扑的更通用的图像矢量化算法。该算法首先自动地将图像中的多个物体分割出来，然后将每个物体视为一个嵌套在高维空间的二流形曲面片（每个面片允许包含孔洞），接下来使用改进的梯度网格拟合这些面片的几何以及颜色信息。相对于之前的方法，该方法不仅允许复杂的拓扑结构，实现了全自动的矢量化，同时处理速度有了大幅度的提升。图 2.9 展示了上述两种算法的矢量化比较。

鉴于梯度网格在颜色和形状表达上的优势，后续的研究进一步扩展了这种技术。Xiao 等[69]提出了基于梯度网格的图像风格迁移方法，Wan 等[70]提出了基于梯度网格和简单涂鸦的图像颜色编辑算法等。总体来讲，这类方法更适用于颜色较为光滑，拓扑简单的图像，对于结构复杂且颜色变化丰富的图像，矢量化结果仍然不够理想。

基于扩散曲线的图像矢量化方法

Orzan 等[3]首次提出将扩散曲线作为新的图元应用到矢量图创作及图像自动矢量化任务中。如图 2.10 所示，几何上，扩散曲线由三次 Bezier 样条定义，整体上包括四个属性：1）几何控制点集合，用于控制扩散曲线的形状；2）曲线左侧的颜色控制锚点，用于初始化左侧区域的颜色；3）曲线右侧的颜色控制锚点，用于初始化右侧区域的颜色；4）模糊程度控制点，用于控制颜色整体的模糊程度。实际创作过程中，用户可以修改这些属性来修改扩散曲线的属性，最终通过求解一个泊松方程实现颜色的均匀扩散，生成最终的图像。该方法不仅适用于交互式的矢量图设计，同时也在自动化的图像矢量化任务中取得了不错的效果。



(a) Bezier样条控制点　(b) 两侧的颜色控制点　(c) 颜色模糊程度控制点　(d) 生成的图像

图 2.10　扩散曲线示例（图像来自于 Orzan 等[3]）

Xie 等[4]注意到 Orzan 等[3]的方法在图像矢量化过程中捕捉的边界曲线并不准确，对于包含丰富颜色细节的图像重建精度有待进一步提高。为此，提出层次化的拉普拉斯扩散曲线实现精确的图像矢量化。该方法首先将图像分别变换到拉普拉斯空间和二阶拉普拉斯空间，然后直接在这两个空间抽取扩散曲线，构建多尺度的扩散曲线，最后通过求解偏微分方程实现颜色的扩散。该方法在重建精度和鲁棒性上都比之前的方法有了较大改进和提升。

尽管扩散曲线矢量化系统允许用户通过控制点修改一些属性，但用户对扩散过程本身并不能直接控制，这限制了此类方法的表达能力且使得一些操作变得繁

琐。为此，Bezerra 等[71]对扩散过程施加了更多的控制如：1）设置屏障，阻止某些颜色的扩散；2）控制扩散的方向和各向异性；3）控制扩散的强度等。总体上，该方法提升了此类方法的编辑灵活性，简化了一些复杂的编辑操作。

后续的研究中，Jeschke 等[72]将 Gabor 噪声引入到扩散曲线中表达图像中的纹理细节，同时该方法自动地选择扩散曲线的最佳锚点，相对于之前的方法，极大地减少了颜色锚点的数量，因此进一步提升了扩散过程的求解效率。Jeschke 等[73]将扩散曲线扩展到三维模型曲面的真实感渲染中。

总体上，扩散曲线具有更好的可编辑性，可以方便地表示物体的边界和颜色信息，因此广泛应用于艺术设计领域。但这种方法对于细节丰富，颜色多变的真实图像重建质量仍然不高，最主要的原因在于此类方法通过求解热扩散方程模拟输入图像的颜色，而扩散方程的求解结果往往非常光滑，因此对于颜色变化不均匀的图像难以得到高质量的矢量化重建。

2.3　图像重着色相关工作

图像重着色主要针对给定彩色图像进行颜色编辑或修改，这类技术在艺术设计、电影特效、图像美化、短视频编辑等领域广泛应用。宽泛地讲，图像增强[74-78]、编辑传播[79-83]、风格迁移[84?-87]等传统研究领域也可归入图像重着色的范畴。但它们的研究目标和交互方式都存在显著的区别：图像增强主要致力于通过全局修改图像的亮度、对比度、去噪、去模糊等操作增强图像整体的视觉效果；编辑传播算法旨在将局部的颜色、亮度、对比度等自动地传播到空间上接近，几何、纹理、相似的其他像素，且该算法通常需要大量的用户交互；风格迁移通常要求用户额外提供一张风格图像，然后从中学习风格特征并将其迁移到输入图像中，这类算法完全不需要用户交互，一定程度上缺乏了编辑自由度；而图像重着色不仅保证了用户的编辑自由度和交互的便捷性，同时还关注编辑的实时性和重着色结果的有效性和自然性。

图像重着色领域的经典工作如基于几何凸包的方法[12,39-40]和基于聚类的算法[11]已经在半透明图层分解和加性图层分解小节做了较为详细的介绍，为此，这一小节主要介绍一些基于能量优化和深度学习的图像重着色算法。

基于能量优化的图像重着色方法

Aksoy 等[88]提出一种基于图像软分割的图层分解方法，并将其应用到图像重着色任务中，实现了较好的编辑局部性。具体来讲，该算法假设每一个图层的颜色服从一个高斯分布，多个图层通过加性混合的方式进行融合得到最终的图像。算

法整体上分为两步即初始图层估计和图层不透明度估计。在初始图层估计阶段，算法通过多次迭代逐步确定图层的数目和每一个图层的均值和协方差矩阵。具体来说，在每一次迭代中，都将那些目前还不能被已有图层精确重建的像素集中起来重新估计参数，直到大部分像素都已经被精确重建为止。在图层不透明度估计阶段，算法给出了关于图层的均值、协方差矩阵以及不透明度的损失函数，该函数同时考虑了图层的重建误差以及图层的稀疏性，最后通过优化该能量函数求解得到图层的不透明度。

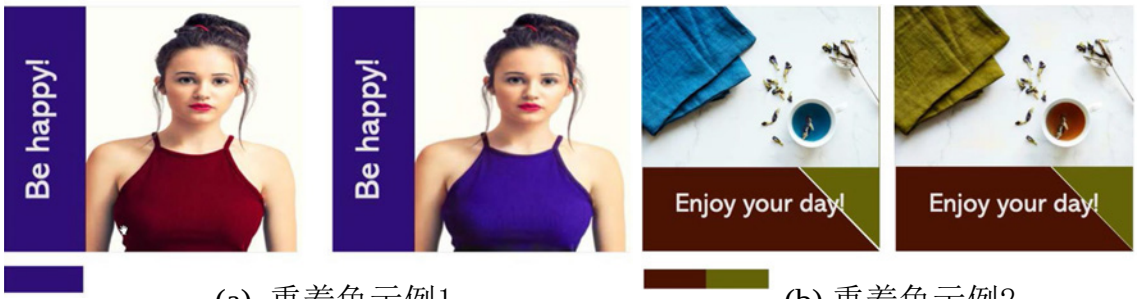
上述方法的问题主要体现在两方面：一是算法在初始图层估计和不透明度求解复杂度都较高，计算时间较长；二是用户只能通过修改图层的均值、协方差矩阵或导入到图像处理软件中进行编辑，因此不够直观。Lin 等[89]通过多元回归模型[90]从图像中抽取符合人类感知的调色板颜色，并通过优化的方式求解图层不透明度。Zhang 等[91]提出了一种联合优化的方法同时求解调色板和对应的插值权重。该优化算法不仅考虑了重建误差、重建图像的光滑性，插值权重的稀疏性等多个约束条件，还能自动决定调色板的颜色数量，使得整个调色板提取过程完全不需要用户交互。

基于深度学习的图像重着色方法

近年来，随着人工智能技术的快速发展，基于深度学习的图像重着色技术受到研究者的广泛关注。Cho 等[92]提出 PaletteNet，该网络自动地将给定调色板的颜色主题映射到目标图像上。PaletteNet 包含两个子网络：一是特征编码器（Feature Encoder）用于抽取输入图像的内容特征；二是重着色解码器（Recoloring Decoder）用于将内容特征和调色板解码到输出图像上。网络通过有监督的方式进行训练，数据集由设计网站上爬取的图像构成。相对于现有算法，PaletteNet 实现了内容感知的重着色，能够自动地将调色板颜色映射到图像中较为合理的区域。

Akimoto 等[93]提出一种基于 U 型网络（U-Net）的自动图层分解网络。该算法需要用户给定调色板，图层分解网络主要包括两个子网络：一是不透明度预测子网络，它根据给定的调色板和输入图像预测每一个调色板颜色对应的图层不透明度；二是图层不透明度优化子网络，第一步得到图层重建误差较大，这一步固定图层的不透明度对图层的颜色进行优化，使得优化后的图层能够精确重建输入图像。这种方法的最大问题在于图层并不是单色的，调色板不能完全表示图层的颜色信息，因此不能直接采用调色板进行编辑。

在平面设计中，用户经常需要在图像中添加新的元素，并且希望新加入的元素的颜色要符合图像的目标风格或颜色主题。为此，Zhao 等[94]提出一种内容感知的图像重着色网络，目的是帮助美术设计师实现自动的颜色调整。该网络主要包



(a) 重着色示例1　　(b) 重着色示例2

图 2.11　Zhao 等[94]提出的重着色算法示例（图像来自于 Zhao 等[94]）

括两个子网络：1）着色区域选择子网络，根据给定的元素，自动地选取着色区域；2）重着色子网络，对选定的着色区域进行重新着色，使得重着色后的图像和谐、美观且整体风格一致。图 2.11 展示了该算法的两组重着色示例，第 1 列和第 3 列是输入图像（图像下方是给定的提示颜色），第 2 列和第 4 列是根据提示颜色重着色后的图像。

2.4　本章小结

本章对图层分解相关技术进行了回顾，同时对图像矢量化和图像重着色两个领域的研究工作进行了详细的调研。针对图层分解，本章从半透明图层分解、加性图层分解和非线性图层分解三个方面对相关工作进行了综述。针对图像矢量化领域，对基于样条的、基于梯度网格的、基于扩散曲线的图像矢量化方法进行简要的回顾，并分析了存在的问题。针对图像重着色领域，分别对基于能量优化的方法和基于深度学习的方法进行了综述并分析了这些方法的优势和缺陷。相关工作调研为本文研究课题的选题提供了必要的支撑。

本文重点关注半透明图层分解技术和加性图层分解技术及其在图像视频编辑中的应用。针对半透明图层分解技术，现有工作或多或少依赖用户交互，无法实现自动的图层分解，同时分解的图层不能反映图像本身的形状和层次结构，不方便用户后续的编辑。本文第三章提出一种全自动的半透明图层分解技术，不仅在给定输入的情况下实现了全自动的图层分解，同时通过引入感知规则使得分解的图层更加直观，为后续的编辑提供了便利。针对加性图层分解技术，现有方法只在低维的颜色空间实施图层分解，无法捕捉图像的语义信息，因此难以实现语义级别的图像颜色编辑。本文第四章提出一种面向图像内容感知重着色的加性图层分解算法，有效地捕捉了图像的语义信息，实现了内容感知的图像颜色编辑。同时本文第五章将面向图像的加性图层分解技术拓展到视频场景，将输入视频分解为一组时变图层，实现了简单高效、直观自然的视频颜色编辑。

第 3 章　面向图像矢量化的半透明图层分解

位图和矢量图是两种主要的图像表示形式。位图通过点阵表示所有像素的颜色，矢量图通过数学定义的几何图元和函数表示图像的形状和颜色。相对于矢量图，位图能够更加精确地表现细微的颜色变化，尤其对于色彩和亮度变化较为丰富的自然景物，位图图像更为逼真。相对于位图，矢量图的优势主要体现在三方面：一是占用空间小，易于存储和传输；二是分辨率无关，无限缩放图像不失真；三是支持精确编辑，用户可以精确地修改图像中物体的形状和颜色。基于以上优点，矢量图在商标、地图、字体、徽章设计等领域广泛应用。

如第 2 章所述，很多经典的图像矢量化算法如基于梯度网格或基于样条曲线的算法通常将图像分解为多个不透明区域，并通过复杂的样条函数逼近每个区域的形状和颜色。尽管这类方法具有很高的重建精度，可以最大程度地恢复原始图像的颜色，但大量的不透明区域变得难以维护，同时每个区域的颜色通过样条函数定义，使得用户编辑较为繁琐。本章主要针对一类基于半透明图层分解的图像矢量化技术进行研究。这类方法将图像矢量化为空间上相互重叠的半透明图层，不仅很好地反映了图像的层次结构，同时通过简单的函数表示图层的颜色，极大地降低了用户的编辑负担。目前这类方法的缺点在于其仍然依赖大量的用户交互，且分解的图层不够完整，为了使算法更加自动化并提升图层的质量，本章提出一种全自动的面向图像矢量化的半透明图层分解算法。

3.1　本章概述

本章提出一种面向图像矢量化的半透明图层分解技术，将给定的位图分解为一组相互重叠的、线性渐变的半透明矢量图层。相比于 Richardt 等人[8]和 Favreau 等人[9]提出的半透明图层分解算法，本文算法的贡献主要体现在三方面：1）提出的算法可以自动运行而无需用户交互；2）引入了一些感知的规则，排除了一些不合理的图层分解，使得分解的半透明图层更能反映图像本身的形状结构；3）得到的图层具有更好的完整性，更加方便用户编辑。

本章剩余部分按如下方式组织：首先对本章算法涉及到的半透明线性渐变图层，图层的合成等基本概念进行介绍；其次介绍半透明图层分解问题的输入和输出并给出形式化定义；然后对本章提出的算法的技术细节进行全面细致的描述；接下来，给出实验结果和对比验证本章算法的有效性；最后对本章提出的算法进行总结并对可能的改进方向进行讨论。

3.2　相关背景

图层是很多图像处理软件如 Adobe Photoshop，Adobe Illustrator 等创建、组织和编辑图像的基础。本文方法致力于将给定的图像分解为一组空间上相互重叠的半透明图层。为了方便用户编辑，这里每个图层的颜色和不透明度通过线性渐变的函数定义，因此本章也将这些半透明图层称之为线性渐变图层。本节首先回顾线性渐变图层的基本概念，然后描述如何通过这些图层合成最终的图像。

3.2.1　线性渐变图层

形式上，一个线性渐变的半透明图层通过一个蒙板（Mask）L 和几个线性渐变的参数共同定义。具体而言，每个半透明的四通道 RGBA 图层中任意像素点 $\mathbf{p}=(x,y)$ 的颜色和不透明度表示为：

$$C_{\mathrm{RGBA}}(\mathbf{p})=\begin{cases}\mathbf{0} & ，若\ L(\mathbf{p})=0,\\ \mathbf{c_0}+(\mathbf{p}\cdot\mathbf{n})\cdot\mathbf{c_g} & ，若\ L(\mathbf{p})=1.\end{cases}\tag{3.1}$$

其中，$\mathbf{n}=(cos\theta,sin\theta)$ 表示渐变方向的单位向量，为了简化问题，我们规定颜色跟不透明的渐变方向一致。$\mathbf{c_0}$ 和 $\mathbf{c_g}$ 均为四维向量，$\mathbf{c_0}=(r_0,g_0,b_0,\alpha_0)$ 表示原点处的颜色和不透明度值，$\mathbf{c_g}=(d_r,d_g,d_b,d_\alpha)$ 表示颜色和不透明度在渐变方向上的变化率。蒙板 L 用于定义图层的覆盖范围，其中，$L(\mathbf{p})=1$ 表示图层的内部区域，$L(\mathbf{p})=0$ 表示图层的外部区域。除此之外，本文要求任意像素的颜色值和不透明度值位于单位超立方体 $[0,1]^4$ 内。图 3.1 展示了一个线性渐变的图层及其对应的各个参数。

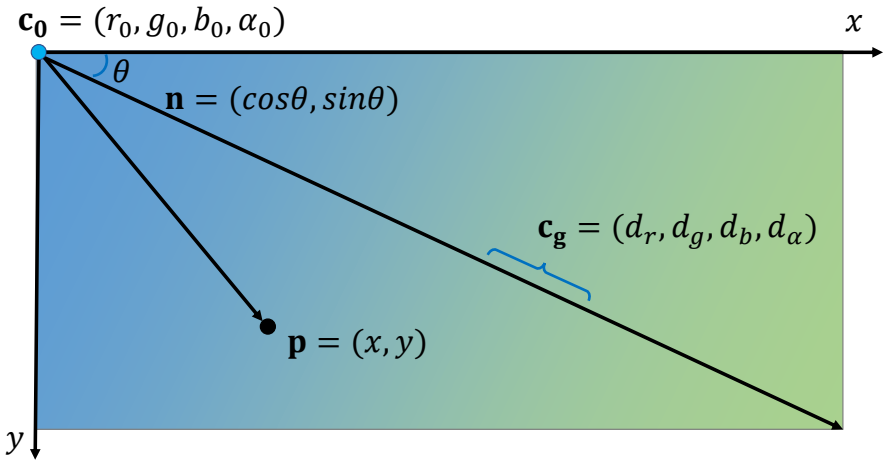


图 3.1　线性渐变图层各参数示意

总体上，除了蒙板 L 外，每个线性渐变的图层总共包含 9 个参数：θ，r_0，g_0，b_0，α_0，d_r，d_g，d_b，d_α。为了描述方便，如果不做特别说明，下文中的图层均指线性渐变的半透明图层。

3.2.2　图像的合成

给定一组按顺序排列的 n 个图层，可以通过 Alpha 混合[10]生成最终的图像。这里将式（ 1.2）详细展开为：

$$I_{\mathrm{RGB}}^{k}=\begin{cases} C_{\mathrm{A}}^{k}\cdot C_{\mathrm{RGB}}^{k}+(1-C_{\mathrm{A}}^{k})\cdot I_{\mathrm{RGB}}^{k-1}, & k>0 \\ I_{\mathrm{RGB}}^{0}, & k=0 \end{cases} \tag{3.2}$$

其中，I_{RGB}^{0} 表示默认画板的颜色（通常为不透明的白色图层），$I=I_{\mathrm{RGB}}^{n}$ 表示所有 n 个图层的合成图像。C_{RGB}^{k} 和 C_{A}^{k} 分别表示第 k 个图层的颜色和不透明度。需要注意的是，上述混合公式需要针对每个像素点单独计算，为了简化表示，这里省略了像素点的位置。

3.3　问题的形式化定义

简而言之，我们的目标是逆向求解式（ 3.2）所示的图层合成问题。即给定光栅图像 I 求解一组图层，同时这些图层可以通过式（ 3.2）高质量地重建出输入图像。这是一个富有挑战性的问题，具体来说，有两类参数需要求解：

1）离散参数，包括图层的数目、每个图层对应的蒙板以及这些图层的重叠顺序，本文将其统称为图层配置；

2）连续参数，如 3.2.1 小节所述，表示每个图层的颜色和不透明度的 9 个线性渐变参数即 θ，r_0，g_0，b_0，α_0，d_r，d_g，d_b，d_α，本文将其统称为图层参数。

根据给定的输入图像逆向求解图层配置和图层参数是一个高度非线性、欠约束的病态问题。本文的思路是先求解出图层配置，然后再估计每个图层的参数。一般来讲，图层配置的计算非常困难，但图层配置一旦确定，后续的图层参数估计变得相对容易。为了降低图层配置的求解难度，除了输入图像之外，本文仿照 Favreau 等[9]的做法，要求用户额外提供一张输入图像的区域分割结果作为输入。

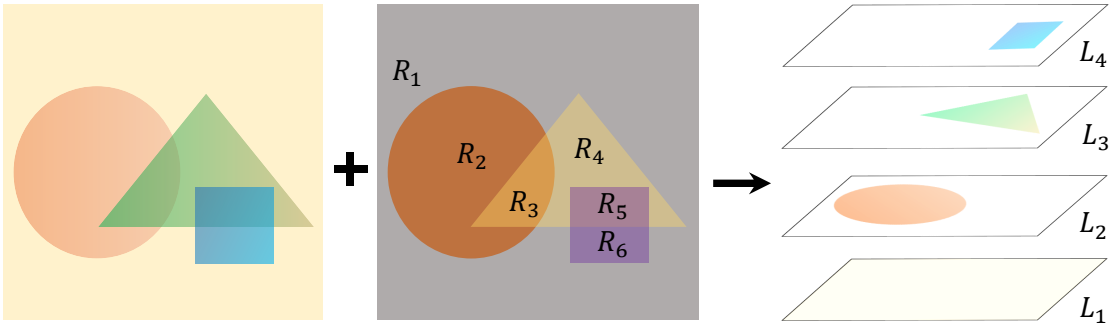
3.3.1　输入和输出

如图 3.2所示，本文提出的面向图像矢量化的半透明图层分解算法的输入和输出分别为：

- 输入：一幅光栅图像 I 和一幅该光栅图像的分割结果 $R=\{R_1,R_2,\cdots,R_m\}$。
- 输出：一组有序的半透明图层 $L=[L_1,L_2,\cdots,L_n]$。

3.3.2　两个基本假设

图层分解的一个重要任务就是要确定每个图层的覆盖范围即图层蒙板。假设给定了图层的数目为 n，那么任意像素 $\mathbf{p}=(x,y)$ 都可能被某个图层 L_i 覆盖或不覆



(a) 输入图像　　(b) 输入分割图　　(c) 输出的图层

图 3.2　本章算法的输入和输出示意

盖，因此，针对每个像素，图层的覆盖情况共有 2^n 种。又因为所有像素相互独立，对于一张宽度为 w，高度为 h 的图像，所有可能的图层蒙板组数为 2^{nwh} 种。由此可见，逐像素地确定图层的覆盖范围完全不可行。为了降低求解难度，我们要求用户提供额外的图像分割结果，并给出两个基本假设：

假设 1：属于同一个分割区域的所有像素被图层覆盖的情况完全一致。

假设 2：图层蒙板由分割图中相邻的区域构成，区域是构成蒙板的基本元素。

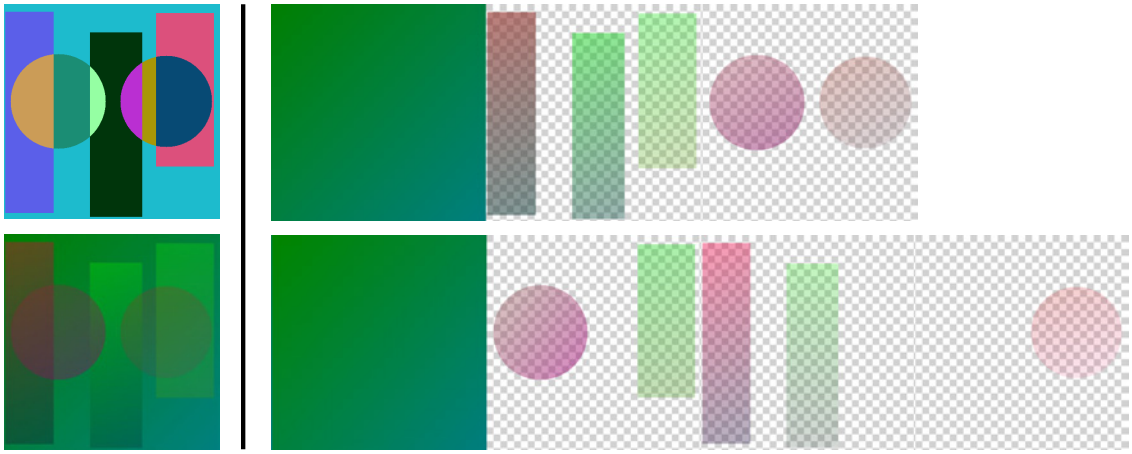
这里再次以图 3.2 为例对上述两个假设进行简要说明。假设 1 强调的是区域内的所有像素均被相同的一组图层覆盖，如 R_4 包含的所有像素均被图层 L_1 和 L_3 覆盖，R_5 包含的所有像素均被图层 L_1，L_3 和 L_4 覆盖。假设 2 强调的是图层蒙板完全由相邻区域组成，如最底部的图层 L_1 覆盖了图像的所有区域，因此 $L_1 = \{R_1, R_2, R_3, R_4, R_5, R_6\}$。图层 L_2 所示的圆形包括 R_2 和 R_3 两个区域，因此 $L_2 = \{R_2, R_3\}$。图层 L_3 所示的三角形包括 R_3，R_4 和 R_5 三个区域，因此 $L_3 = \{R_3, R_4, R_5\}$。

上述两个假设极大地降低了计算开销，使得图层蒙板的确定过程中不必逐像素而是针对每个区域内所有的像素进行统一考虑。假设给定图层数目为 n，图像分割后的区域数目为 m，那么所有图层蒙板的组数为 2^{nm}。尽管时间复杂度有所降低，但如果分割区域较多，其复杂度依然很高。为此本文将通过一些感知策略排除一些不合理的图层配置，压缩可行解的搜索空间，确保算法能在较短的时间内得到所有可能的图层分解方案。

3.3.3　能量函数

如上文所述，半透明图层分解本质上是一个欠约束的病态问题，针对同样的输入，往往可以得到多种图层分解方案。如图 3.3 所示，针对左侧（图 3.3(a)）的输入图像和分割图，右侧（图 3.3(b)）展示了两组可能的图层分解方案，这里图层按

从左到右的顺序自底向上排列，尽管图层的层叠顺序差异很大但都能高质量地重建输入图像。事实上，针对图 3.3(a) 所示的输入，可能存在数十种图层分解方案。



(a) 输入　　(b) 分解的图层序列

图 3.3　图层分解问题的多解性示例

为了得到最优的图层分解，本文定义了一个能量函数对图层的质量进行度量。能量函数主要有三方面的考虑：一是重建误差要尽可能低，二是要保证图层的颜色和不透明值位于合理的范围，三是对于图层堆叠的偏好。为此，能量函数定义为重建误差项，值域误差项和的层叠偏好项的权重和：

$$E = w_r E_{\text{recon}} + w_g E_{\text{gamut}} + w_s E_{\text{stack}} \tag{3.3}$$

其中，E_{recon} 表示重建误差项，E_{gamut} 表示值域误差项，E_{stack} 表示图层的层叠偏好项，w_r，w_g，w_s 表示上述三项的权重，用于平衡各项对整个能量函数的贡献。根据经验，本文将这些参数设定为：$w_r = 20$，$w_g = 10$，$w_s = 0.02$。

重建误差是图像矢量化的重要评价指标，经过 Alpha 混合重建的图像应当跟输入图像尽可能接近。为此，本文将其定义为图层合成后的图像跟输入图像的均方误差（Mean Squared Error，MSE）的平均值：

$$E_{\text{recon}} = \frac{1}{N} \sum_{\mathbf{p}} \left\| I^n_{\text{RGB}}(\mathbf{p}) - I_{\text{RGB}}(\mathbf{p}) \right\|^2 \tag{3.4}$$

其中，$\mathbf{p}$ 表示图像中的任意像素，N 表示像素的总数，$I^n_{\text{RGB}}(\mathbf{p})$（参见式（3.2））和 $I_{\text{RGB}}(\mathbf{p})$ 分别表示 $\mathbf{p}$ 在合成图像和输入图像中的颜色值。

值域项主要有两方面的考虑：一是确保图层的颜色位于合理的范围即 $[0,1]^3$，二是为了良好的编辑性，本文期望分解得到尽可能多的半透明图层，为此对不透

明度较高（不透明度 > 0.8）的图层进行惩罚。总体来讲，图层值域误差定义如下：

$$E_{\text{gamut}} = \frac{1}{M}\sum_{i=1}^{n}\sum_{\mathbf{p}\in L_i}\left\|C^i_{\text{RGB}}(\mathbf{p}) - \overline{C^i_{\text{RGB}}(\mathbf{p})}\right\|^2 + \left\|C^i_{\text{A}}(\mathbf{p}) - \overline{C^i_{\text{A}}(\mathbf{p})}\right\|^2 \tag{3.5}$$

我们遍历每个图层的所有像素，检查每个像素的颜色和不透明度的取值，对超出给定取值范围的像素进行统计。其中，M 表示遍历的像素总数，$C^i_{\text{RGB}}(\mathbf{p})$ 和 $C^i_{\text{A}}(\mathbf{p})$ 分别表示图层 L_i 中像素 $\mathbf{p}$ 的颜色和不透明度。$\overline{C^i_{\text{RGB}}(\mathbf{p})} = \text{clip}(C^i_{\text{RGB}}(\mathbf{p}), 0, 1)$ 表示将 $C^i_{\text{RGB}}(\mathbf{p})$ 在区间 $[0, 1]$ 截断，$\overline{C^i_{\text{A}}(\mathbf{p})} = \text{clip}(C^i_{\text{A}}(\mathbf{p}), 0, 0.8)$ 表示将 $C^i_{\text{A}}(\mathbf{p})$ 在区间 $[0, 0.8]$ 截断。上式第一个平方项对颜色超出正常范围的图层进行惩罚，第二项对不透明度为负或不透明度超过 0.8 的图层进行惩罚。

层叠偏好项主要用于鼓励较小的图层位于较大的图层之上，这样方便用户在编辑过程中选取图层，定义如下：

$$E_{\text{stack}} = \sum\nolimits_{1\leqslant i<j\leqslant n, L_i\cap L_j\neq\varnothing} 1\left(\left|L_i\right| < \left|L_j\right|\right) \tag{3.6}$$

其中，$\left|L_i\right|$ 表示图层 L_i 的面积即其包含的像素数目，$1(\cdot)$ 表示指示函数，如果括号内的条件为真则函数值为 1，否则为 0。上式对所有空间上相互重叠的图层对 $\{L_{i<j}, L_j\}$ 进行统计，如果下方图层的面积小于上方图层的面积则能量值增加 1。

3.4　算法概览

为了将给定的光栅图像分解为一组不透明的图层，首先需要决定图层配置包括图层的数目、每个图层的蒙板以及图层的顺序，然后再通过优化的方式估计每个图层的参数。针对图层配置求解这个挑战性的问题，本文有两个重要的观察：

1）首先，图层配置可以自然地从本文定义的一种新型的数据结构即区域支持树（Region Supporting Tree，RST）中导出；

2）其次，区域支持树又可以从区域邻接图（Region Adjacency Graph，RAG）中通过枚举生成树得到。此外，本文还可以通过引入一些基于感知的规则对区域邻接图进行简化，从而加速生成树的枚举过程。

基于上述观察，本文设计了如图 3.4 所示的求解框架。提出的算法总体上分为如下五步：

第一步：构建区域邻接图。首先，根据输入的区域分割图像构建区域邻接图。具体来说，区域邻接图是一个有向图，每个结点代表分割图中的一个区域，每条有向边代表一对相邻区域的支持关系（Region Supporting Relationship）（参见 3.5 小节）。区域邻接图本质上定义了所有区域支持树（图层配置）的搜索空间。然后根据一些感知规则对初始的区域邻接图进行简化，剪除一些不合理的区域支持关系，

有效缩减搜索空间。区域邻接图的构建将在 3.6 小节详细介绍。

第二步：枚举区域支持树。其次，通过深度优先遍历的方式在区域邻接图中枚举所有生成树，每一个生成树对应一个可能的图层配置，从而得到所有可能的图层配置。为了加速搜索过程并进一步剪除不合理的图层配置，本文对生成树的深度进行限定并引入关于 X 型交叉的半透明假设，有效地提升了算法的运行效率。区域支持树的枚举将在 3.7 小节详细介绍。

第三步：合并图层。然后，针对每一个有效的区域支持树，本文首先导出一个初始的图层配置，然后根据 X 型交叉的半透明假设对某些图层执行必要的合并操作，最后导出该区域支持树对应的图层配置。图层合并将在 3.8 小节详细介绍。

第四步：求解图层参数。接着，针对每一个确定的图层配置，本文通过能量优化的方式求解所有图层的线性渐变参数。图层参数求解将在 3.9 小节详细介绍。

第五步：筛选最优结果。最后，对所有的图层分解方案，计算它们的能量损失，并选出能量最小的图层分解方案返回给用户。

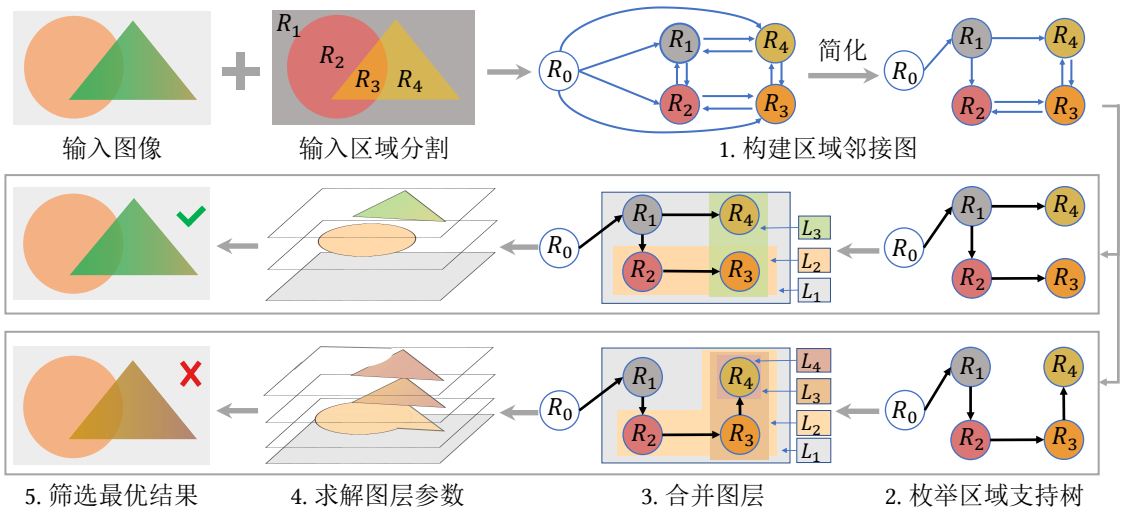


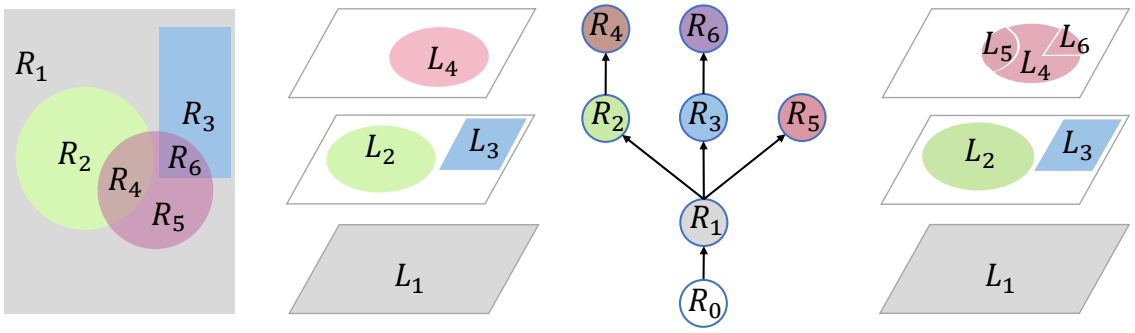
图 3.4　算法的求解流程图

3.5　基本概念

接下来，在正式介绍算法之前，为了方便理解，本节首先介绍几个贯穿本章的基本概念：图层高度，图层支持，顶级图层，区域支持，区域支持树。本节将通过图 3.5 来介绍这几个概念。图 3.5 (a) 所示的输入图像共包含 6 个区域，图 3.5 (b) 所示图层配置中各个图层的覆盖范围分别为：

$$L_1 = \{R_1, R_2, ..., R_6\}, L_2 = \{R_2, R_4\}, L_3 = \{R_3, R_6\}, L_4 = \{R_4, R_5, R_6\}. \quad (3.7)$$

图层高度　令最底部的图层的高度为 1（画板的高度为 0），图层 L_i 的高度定义为它所覆盖的图层的最大高度加 1。如图 3.5 (b) 所示，这里的 4 个图层按高度



(a) 输入　(b) 图层配置1　(c) 区域支持树　(d) 图层配置2

图 3.5　图层配置及支持关系示意

分为了三级，其中，L_1 的高度为 1，L_2 和 L_3 的高度为 2，L_4 的高度为 3。

图层支持　给定两个图层 $L_{i<j}$ 和 L_j（L_i 位于下方，L_j 位于上方），如果它们满足：1）空间上相互重叠；2）存在 1 个或多个重叠的区域没有被 L_i，L_j 之间的任意图层所覆盖。那么就称图层 L_i 支持图层 L_j，简单地表示为 $L_i \rightarrow L_j$，并将满足上述条件 2）的区域称为衔接区域。例如在图 3.5(b) 中，L_1 和 L_4 在空间上相互重叠，重叠的 3 个区域分别为 R_4，R_5 和 R_6。其中，R_5 除了被图层 L_1 和 L_4 覆盖之外，没有再被任何图层所覆盖，因此 L_1 支持 L_4 且 R_5 是 L_1 和 L_4 的衔接区域。R_4 和 R_6 并不是 L_1 和 L_4 的衔接区域，因为 R_4 和 R_6 除了被 L_1 和 L_4 覆盖以外，还分别被中间图层 L_2 和 L_3 覆盖。类似地，L_2 支持 L_4 且 R_4 是它们的衔接区域，L_3 支持 L_4 且 R_6 是它们的衔接区域。

顶级图层　一个区域可能同时被多个图层自底向上地覆盖，本文将覆盖某个区域最上方的高度最大的图层称为该区域的顶级图层。如图 3.5(a,b) 所示，R_4 同时被 3 个图层 L_1，L_2 和 L_4 覆盖，因为 L_4 位于最上方，因此 R_4 的顶级图层为 L_4。类似地，R_2 的顶级图层为 L_2，R_6 的顶级图层为 L_4。

区域支持　给定两个区域 R_i 和 R_j，假设他们的顶级图层分别为 L_i 和 L_j，如果它们满足：1）R_i 和 R_j 相邻；2）L_i 支持 L_j；3）R_j 是 L_i 和 L_j 的衔接区域。那么就称区域 R_i 支持区域 R_j，简单地表示为 $R_i \rightarrow R_j$。如图 3.5 (a,b) 所示，R_1 支持 R_5，R_2 支持 R_4，R_3 支持 R_6。但 R_1 并不支持 R_4 和 R_6，尽管 R_1 跟 R_4 和 R_6 都相邻且 R_1 的顶级图层 L_1 支持 R_4 和 R_6 的顶级图层 L_4，但 R_4 和 R_6 并不是 L_1 和 L_4 的衔接区域。

给定输入图像的区域分割结果以及对应的图层配置，可以导出关于区域支持的两个重要性质：

性质 1：一个区域尽管可以同时支持多个其他区域，但它最多只能被一个区域支持。假设存在某个区域 R_k 同时被两个区域 R_i 和 R_j 支持，假定他们的顶级图层分别为 L_k, L_i 和 L_j。那么根据定义，R_k 同时是 $\{L_i, L_k\}$ 和 $\{L_j, L_k\}$ 的衔接区域，

而这违背了衔接区域的定义，因此不存在一个区域同时被多个区域支持的情况。

性质 2：定义 $R_i \to R_j \to \cdots \to R_s$ 为从任意区域 R_i 开始到 R_j 结束的一条支持路径，那么这条路径不含环路。根据区域支持的概念，沿着支持路径方向，区域的顶级图层必将在空间上逐渐上升（图层高度递增），如果存在环路，那么必然存在某个上方的图层反过来支持某个下方的图层，而这显然是不可能的，因此该支持路径不可能包含环路。

基于区域支持的概念以及导出的两条重要性质，接下来我们介绍区域支持树这一重要概念。

区域支持树　我们发现区域支持关系可以形式化为一个有向无环图（Directed Acyclic Graph，DAG）。在这个 DAG 中，每个结点跟分割图中的某个区域一一对应，每条有向边表示了两个相邻区域的支持关系。任意结点的入度小于等于 1，如果该结点对应的区域不被任何区域支持，那么它的入度为 0，反之为 1。为了简单起见，本文新增一个虚拟的结点 R_0 表示画板，同时让 R_0 向所有入度为 0 的区域引出一条有向边。如此一来，该有向无环图形式上变成了一棵树且 R_0 充当这棵树的树根。如图 3.5所示，根据图 3.5(a) 所示的区域分割和图 3.5(b) 所示的图层配置，我们可以得到所有的区域支持关系：$R_1 \to R_2$，$R_1 \to R_3$，$R_1 \to R_5$，$R_2 \to R_4$，$R_3 \to R_6$。同时新增结点 R_0 并让其指向入度为 0 的结点 R_1，最终得到图 3.5(c) 所示的区域支持树。

需要注意的是，尽管一个图层配置只对应一棵区域支持树，但一棵区域支持树却可以同时对应多个图层配置。如图 3.5(d) 所示的图层配置也对应图 3.5(c) 所示的区域支持树。图 3.5 (b) 和图 3.5(d) 所示的图层配置的区别在于：前者的顶部是一个完整的圆形图层，后者的顶部是三个碎片化的图层。为了得到完整的图层，利用一些规则对图层进行合并是必不可少的步骤，本章 3.8小节将介绍具体的方法对图层实施必要的合并操作。

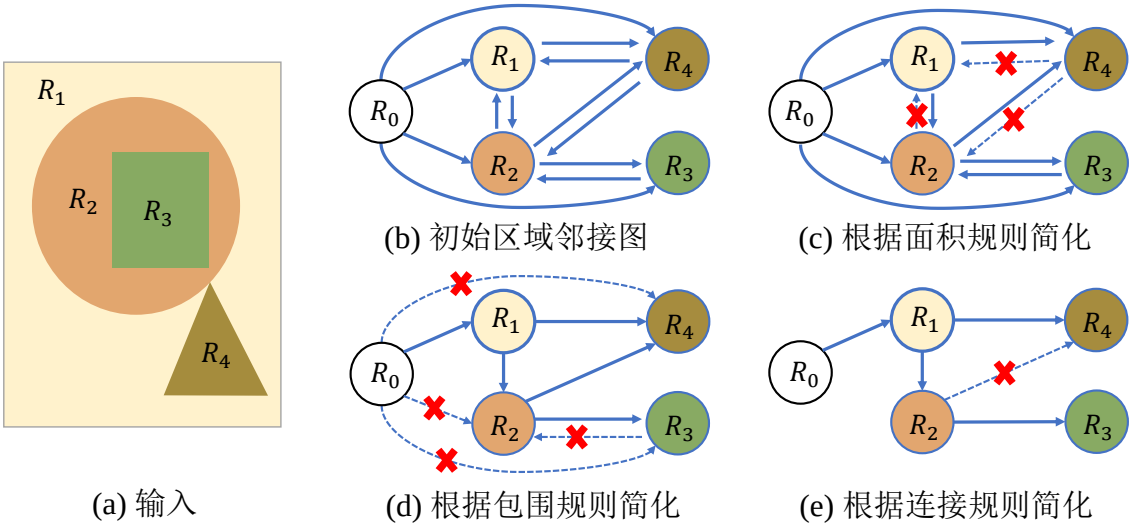
介绍完区域支持、区域支持树等基本概念之后，接下来我们对本章算法分步进行描述。

3.6　构建区域邻接图

算法的第一步是根据给定的分割图像构建区域邻接图。具体来说，区域邻接图中的每个结点跟输入分割图中的一个区域一一对应，每条有向边跟两个相邻区域的支持关系一一对应。初始情况下，区域的支持关系是未知的，为此，本文首先在所有相邻区域对之间添加双向边，允许它们相互支持。除此之外，本文还添加一个虚拟结点 R_0 表示画板，并从 R_0 出发向所有其他结点引出有向边。给定图 3.6

(a) 所示的分割图像，构建的初始区域邻接图如图 3.6 (b) 所示。

区域邻接图本质上定义了所有区域支持树的搜索空间，每一个区域支持树跟区域邻接图的一个生成树（Spanning Tree）一一对应。因此可以通过遍历区域邻接图的所有生成树获取区域支持树并进一步确定图层配置。然而，初始区域邻接图是一个包含大量有向边的稠密图，搜索生成树的时间复杂度非常高。为此，本文引入一些基于人类感知的规则对初始区域邻接图进行简化。



(a) 输入　(b) 初始区域邻接图　(c) 根据面积规则简化

(d) 根据包围规则简化　(e) 根据连接规则简化

图 3.6　区域邻接图的构建与简化

本文提出了三个规则对初始区域邻接图进行简化，用以删除一些感知上不合理的连边，从而缩小生成树的搜索空间。

- **面积规则** 给定两个相邻的区域 R_i 和 R_j，如果 R_i 的面积远小于 R_j 的面积（$|R_j|/|R_i| > 2$），那么 R_i 将不被允许支持 R_j。区域支持关系实际隐含了可能的图层覆盖顺序，如果允许面积较小的区域支持面积较大的区域，那么可能导致最终较大的图层覆盖在较小的图层之上，这样不符合编辑习惯，同时也违反了图像背景感知理论（Figure-ground Perception）[95]。如图 3.6 (c) 所示，这里删除了违反这一规则的区域支持关系：$R_2 \to R_1$，$R_4 \to R_1$，$R_4 \to R_2$。
- **包围规则** 如果一个区域 R_i 完全包围另一个区域 R_j，那么本文期望 R_j 所在的顶级图层紧邻地位于 R_i 的顶级图层的上方。因此 R_j 将不被允许支持 R_i（即使 $|R_j| > |R_i|$），也不允许 R_j 被 R_0 支持。此外，如果一个邻接的区域集合被某一个单独的区域包围，那么这个集合中的任意区域将不允许被 R_0 支持，否则它将违反图像背景感知理论[95]。如图 3.6(d) 所示，这里删除了违反这一规则的区域支持关系：$R_0 \to R_2$，$R_0 \to R_3$，$R_0 \to R_4$，$R_2 \to R_3$。
- **连接规则** 某些区域可能同时跟多个区域相邻，但不同邻接的区域对之间的连接强度却不尽相同。本文将两个区域 R_i 和 R_j 的连接强度 $S_{i,j}$ 定义为它们

共同边界的长度，如果 $S_{i,j}/S^i_{max} < 0.4$（S^i_{max} 表示 R_i 跟其他区域的最大连接强度），本文认为区域 R_i 跟 R_j 的关联性很弱，因此不再允许 R_i 支持 R_j，因为这违反了形状感知理论（Skeletal Model of Shape Part Perception）[95]。如图 3.6(a) 所示，R_2 跟 R_4 的共享边界很短、连接强度很弱，为此这里将图 3.6(e) 中 R_2 到 R_4 的有向边删除。

应用上述三条规则后，删除了区域邻接图中大量不合理的连接，很大程度缩减了生成树的搜索空间。接下来，介绍如何从区域邻接图中搜索有效的生成树（区域支持树）。

3.7　枚举区域支持树

本文通过深度优先遍历的方式在简化的区域邻接图中搜索所有的生成树。本文设计的区域支持树包含一个特殊的虚拟画板结点 R_0 作为树根，因此在区域邻接图中从 R_0 开始搜索所有生成树。为了进一步加速生成树的搜索并进一步排除一些不合理的区域支持树，本文设计了两个剪枝策略：

1）限制生成树的深度。为了使得生成树的数量可控并避免不必要的复杂的图层重叠，本文对生成树的深度进行了限制。根据经验，生成树的最大深度被限定为 8，因此图层的最大高度也被限定为 8。

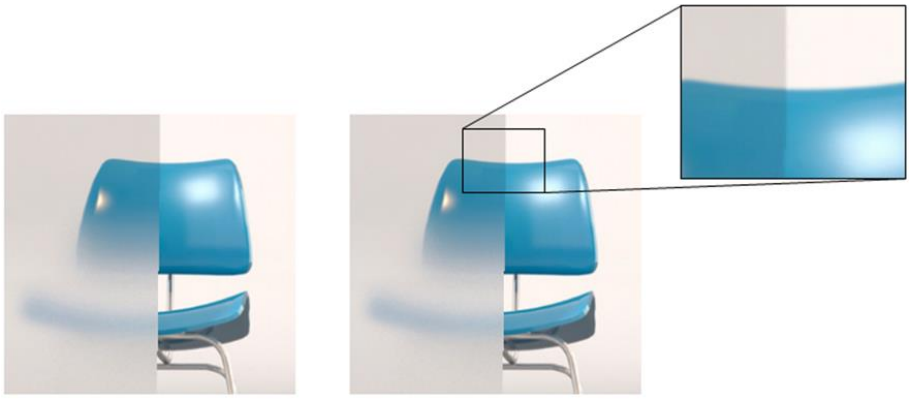


图 3.7　X 型交叉示意①

2）应用关于 X 型交叉的半透明约束。X 型交叉通常指形如 2×2 布局的 4 个相邻区域，且它们的共享边界呈“X”形状。该假设认为这种 X 型的边界是由一个半透明的图层覆盖在其它两个图层上造成的视觉效果。如图 3.7 所示，半透明的灰色图层覆盖在椅背和背景图层上，造成了明显的 X 型交叉效果。X 型交叉假设常用于半透明感知建模[96-98]。本文充分利用关于 X 型交叉的半透明假设排除不合理的图层配置，并在后续的图层合并过程中再次应用该假设对某些图层进行合并。

① 图像来自于 https://www.saund.org/transparency-demo/discussion.html。

具体而言，对于给定的某个 X 型交叉 $\{R_a, R_b, R_c, R_d\}$，假设这些区域的顶级图层分别为 $\{L_a, L_b, L_c, L_d\}$。如图 3.8 所示，根据区域间的支持关系，本人将 X 型交叉涉及的图层的覆盖方式分为 4 种:

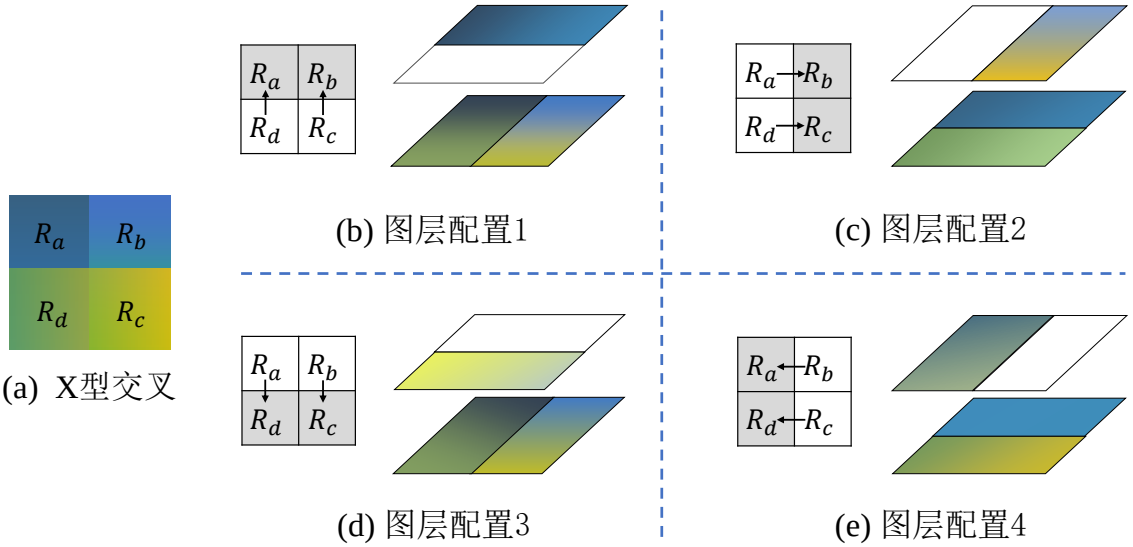


图 3.8　X 型交叉以及 4 种可能的图层覆盖方式

- 如果 $R_d \to R_a$ 且 $R_c \to R_b$（图 3.8(b)），则 R_a 和 R_b 共享同一顶级图层（$L_{a,b} = L_a = L_b$），且该图层位于 R_c 和 R_d 对应的顶级图层 L_c 和 L_d 的上方；
- 如果 $R_a \to R_b$ 且 $R_d \to R_c$（图 3.8(c)），则 R_b 和 R_c 共享同一顶级图层（$L_{b,c} = L_b = L_c$），且该图层位于 R_a 和 R_d 对应的顶级图层 L_a 和 L_d 的上方；
- 如果 $R_b \to R_a$ 且 $R_c \to R_d$（图 3.8(d)），则 R_a 和 R_d 共享同一顶级图层（$L_{a,d} = L_a = L_d$），且该图层位于 R_b 和 R_c 对应的顶级图层 L_b 和 L_c 的上方；
- 如果 $R_a \to R_d$ 且 $R_b \to R_c$（图 3.8(e)），则 R_c 和 R_d 共享同一顶级图层（$L_{c,d} = L_c = L_d$），且该图层位于 R_a 和 R_b 对应的顶级图层 L_a 和 L_b 的上方。

因此，对于任意的 X 型交叉相关的 4 个区域，它们的连接方式只能是上述 4 种方式其中之一。简单来说，对于某个 X 型交叉关联的 4 个区域，如果将其按图 3.8 所示的顺时针或逆时针顺序排列成 2×2 的样式，我们发现两个平行箭头的指向（区域支持方向）是一致的。因此，在搜索生成树的过程中，如果发现 X 型交叉相关的 4 个区域违背了这一假设如 $R_a \to R_b$ 且 $R_c \to R_d$，就可以及时终止当前分支搜索。当然，针对某些特殊的例子如国际象棋棋盘格，X 型交叉可能并不满足上述半透明假设，为此，本文也允许用户根据实际情况启用或禁用 X 交叉假设。

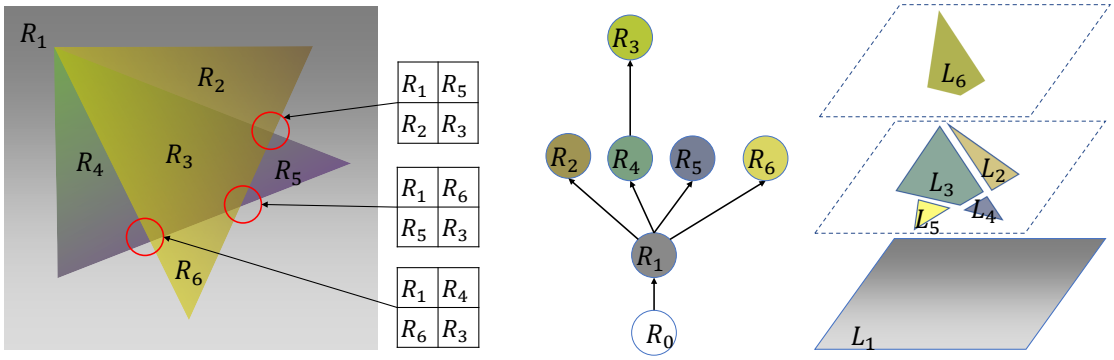
本文使用 Gabow 等[99]提出的算法在区域邻接图中使用深度优先遍历的方式搜索所有生成树，理论上的算法时间复杂度为 $\mathcal{O}(V + E + EN)$，其中 V, E, N 分别表示区域邻接图的顶点数目，边数和生成树的数量。因为本文在构建区域邻接图的过程中使用一些感知规则删除了一些不必要的边，并在生成树的搜索过程中

通过 X 型交叉假设进行了剪枝，有效地减少了生成树的数量，从而大大降低了算法的时间复杂度，因此算法的运行时间整体可控。

3.8　合并图层

针对每一个区域支持树，本文首先导出一个初始图层配置，然后再根据 X 型交叉的假设对一些图层进行合并。为了从区域支持树导出图层配置，本节首先将每个区域视为一个单独的图层，并将每个图层的高度定义为区域支持树中对应结点的深度（区域支持树中根节点 R_0 的深度为 0）。根据区域支持的定义，如果 $R_a \to R_b$，那么 R_a 所在的图层必然包含区域 R_b。为此，针对区域支持树中的每一个非叶子节点 R_i，我们将其子孙节点表示的区域扩展到 R_i 所在的图层。如图 3.9所示，输入图像包含 6 个区域和 3 个 X 型交叉，图 3.9(c) 是搜索到的某个区域支持树，图 3.9(d) 则是从该区域支持树导出的初始图层配置，各个图层的覆盖范围分别为：

$$\begin{aligned} &L_1 = \{R_1, R_2, R_3, R_4, R_5, R_6\}, L_2 = \{R_2\}, L_3 = \{R_3, R_4\}, \\ &L_4 = \{R_5\}, L_5 = \{R_6\}, L_6 = \{R_3\}. \end{aligned} \tag{3.8}$$



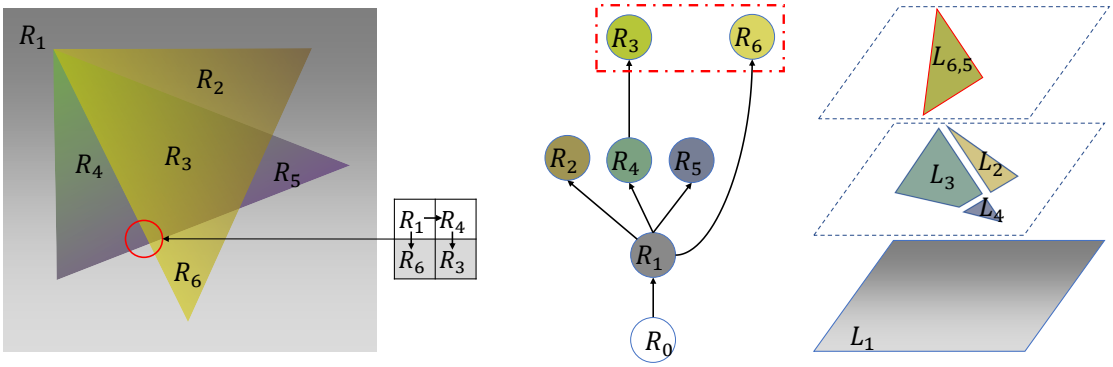
(a) 输入图像及区域分割　(b) X-junction　(c) 区域支持树　(d) 图层配置

图 3.9　初始区域支持树对应的图层配置

接下来，我们依次检查每一个 X 型交叉，并尽可能对某些图层执行合并操作。我们首先检查图 3.10(a) 中最下方的 X 型交叉 $\{R_1, R_4, R_3, R_6\}$，这 4 个区域在区域支持树中的支持关系为：$R_1 \to R_4$，$R_1 \to R_6$，$R_4 \to R_3$。根据 X 型交叉假设，这里应该将区域 R_6 合并到 R_3 所在的图层中，因为区域 R_6 和区域 R_3 原本的图层分别为 L_5 和 L_6，我们将二者合并后的图层命名为 $L_{6,5}$，并删除 R_6 原先的图层 L_5。合并之后我们得到图 3.10(d) 所示的图层配置，各图层的覆盖范围分别为：

$$\begin{aligned} &L_1 = \{R_1, R_2, R_3, R_4, R_5, R_6\}, L_2 = \{R_2\}, L_3 = \{R_3, R_4\}, \\ &L_4 = \{R_5\}, L_{6,5} = \{R_3, R_6\}. \end{aligned} \tag{3.9}$$

接下来，我们检查另一个 X 型交叉（图 3.11(a)），它关联的 4 个区域按顺时

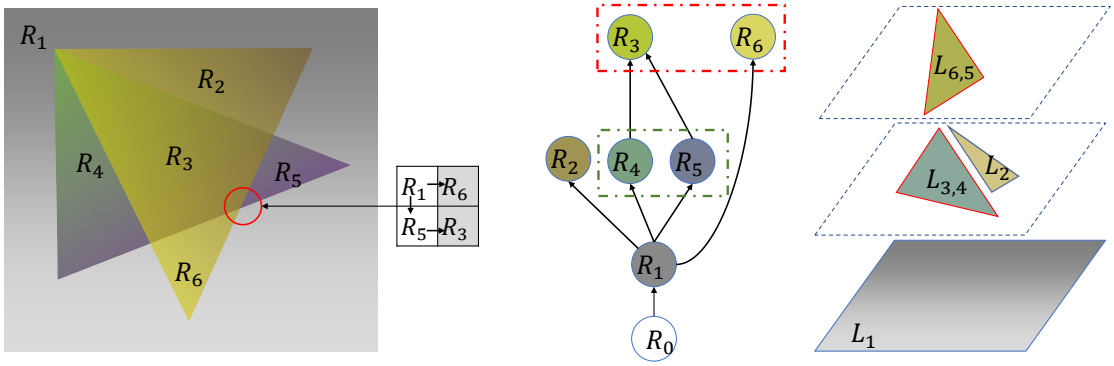


(a) 输入图像及区域分割　(b) X-junction　(c) 区域支持树　(d) 图层配置

图 3.10　合并图层 L_5 和 L_6

针方向排列为：R_1，R_6，R_3，R_5。需要注意的是，这个 X 型交叉涉及的两个区域 R_3 和 R_6 对应的图层已经在上一步合并，且当前的区域支持关系为：$R_1 \to R_5$，$R_1 \to R_6$。根据 X 型交叉假设，只有同时满足 $R_1 \to R_6$，$R_5 \to R_3$，R_3 和 R_6 所在的图层才能被合并，所以这里直接让 $R_5 \to R_3$ 即可。于是现在 R_4 和 R_5 支持同一个区域 R_3。根据区域支持的概念，一个区域 R_i 不能同时被多个区域支持，如果存在这种情况，那么这些支持 R_i 的区域必将共享同一个顶级图层。为此，这里进一步将 R_5 合并到 R_4 所在的图层 L_3，将合并后的图层命名为 $L_{3,4}$，并删除 R_5 原先的图层 L_4。合并后的图层配置如图 3.11(d) 所示，各图层的覆盖范围分别为：

$$L_1 = \{R_1, R_2, R_3, R_4, R_5, R_6\}, L_2 = \{R_2\},$$
$$L_{3,4} = \{R_3, R_4, R_5\}, L_{6,5} = \{R_3, R_6\}. \tag{3.10}$$



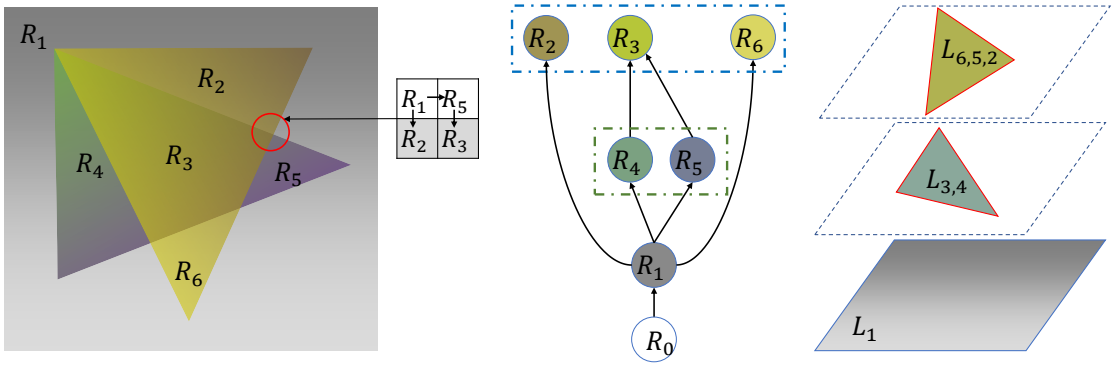
(a) 输入图像及区域分割　(b) X-junction　(c) 区域支持树　(d) 图层配置

图 3.11　合并图层 L_3 和 L_4

现在，我们检查最后一个 X 型交叉 (图 3.12(a))，它关联的 4 个区域按顺时针方向排列为：R_1，R_5，R_3，R_2。这 4 个区域在区域支持树中的支撑关系为：$R_1 \to R_2$，$R_1 \to R_5$，$R_5 \to R_3$。显然，这里需要将区域 R_2 合并到 R_3 所在的顶级图层 $L_{6,5}$，将合并后的图层命名为 $L_{6,5,2}$，并删除 R_2 原先的图层 L_2。最终我们得到了如图 3.12(d)

所示的图层配置，它包含三个图层，各图层的覆盖范围分别为：

$$L_1 = \{R_1, R_2, R_3, R_4, R_5, R_6\},$$
$$L_{3,4} = \{R_3, R_4, R_5\}, L_{6,5,2} = \{R_2, R_3, R_6\}. \tag{3.11}$$



(a) 输入图像及区域分割　(b) X-junction　(c) 区域支持树　(d) 图层配置

图 3.12　合并图层 L_2 和 $L_{6,5}$

基于 X 型交叉假设的图层合并算法的时间复杂度为 $\mathcal{O}(KV + Klog(V))$，其中，$V$ 表示区域支持树的结点数目，K 表示输入分割图中 X 型交叉的数目。

到此为止，每一棵区域支持树对应的图层配置（包括图层数目，每个图层的蒙板以及图层的层叠顺序）已经确定下来。下一节将通过优化方法来估计每个图层对应的线性渐变参数。

3.9　求解图层参数

经过上述几步，本文已经获取了输入光栅图像所有可能的图层配置，本节主要介绍如何求解每一个图层配置中各个图层关于颜色和不透明度的线性渐变参数。

如 3.2.1 小节所述，每个图层共包含 9 个参数：表示线性渐变方向的角度 θ，原点处的颜色和不透明度 $\mathbf{c_0} = (r_0, g_0, b_0, \alpha_0)$，颜色和不透明度在渐变方向上的梯度 $\mathbf{c_g} = (d_r, d_g, d_b, d_\alpha)$。本文通过最小化式（3.3）所示的能量函数求解每个图层的线性渐变参数。具体而言，本文使用非线性优化库 NLopt[100] 并调用 L-BFGS 算法[101-102] 求解该优化问题。L-BFGS 算法常用于求解非线性优化问题，不仅内存开销较小，同时收敛速度很快。该算法需要用户预先计算能量函数的梯度，而式（3.3）所示的能量函数高度非线性，手动计算梯度有一定的困难，为此本文使用一个自动求导的库 autodiff[103] 计算能量函数的梯度。本文使用随机的方式对图层的参数进行初始化，当迭代次数大于 1000 次或能量误差基本保持不变时优化过程结束。

总体上，优化时间跟图层配置的数目以及每个图层配置中的图层数目相关。图

层配置数目越大，每个图层配置中的图层数目越多，算法运行时间越长。一般情况下，当输入区域分割图包含中等数量的区域时，本文优化算法通常能在 1 分钟以内返回结果。本文将在实验部分性能统计小节展示算法各阶段的运行时间。

3.10　筛选最优结果

一般情况下，针对给定的光栅图像，通常能得到数十个甚至上百个图层分解方案。为了得到最优的图层分解结果，本文对所有的图层分解结果按式（ 3.3）所示的能量损失进行排序，最后将能量值最小的图层分解结果保留下来。需要注意的是，目前本文只是对图层的颜色和不透明度进行了参数化的表示，还没有对图层的轮廓边界（图层蒙板）进行矢量表示。为此本文进一步使用 Portrace 库[104]对图层蒙板进行矢量化，最后将完整的矢量图层返回给用户。需要注意的是，这里只是返回了最优结果，用户也可以根据实际需求输出前 K 个最优的图层分解结果进行筛选。

3.11　实验

这一部分通过实验对本章算法的有效性进行验证，主要包括图层分解结果、方法对比、消融实验、性能估计、用户实验五部分。本章实验环境为：

- 操作系统：Windows 10；
- 处理器：Intel i9-9900K 3.6 GHz CPU、16 核；
- 内存：16 GB RAM；
- 编程环境：Microsoft Visual Studio 2019。

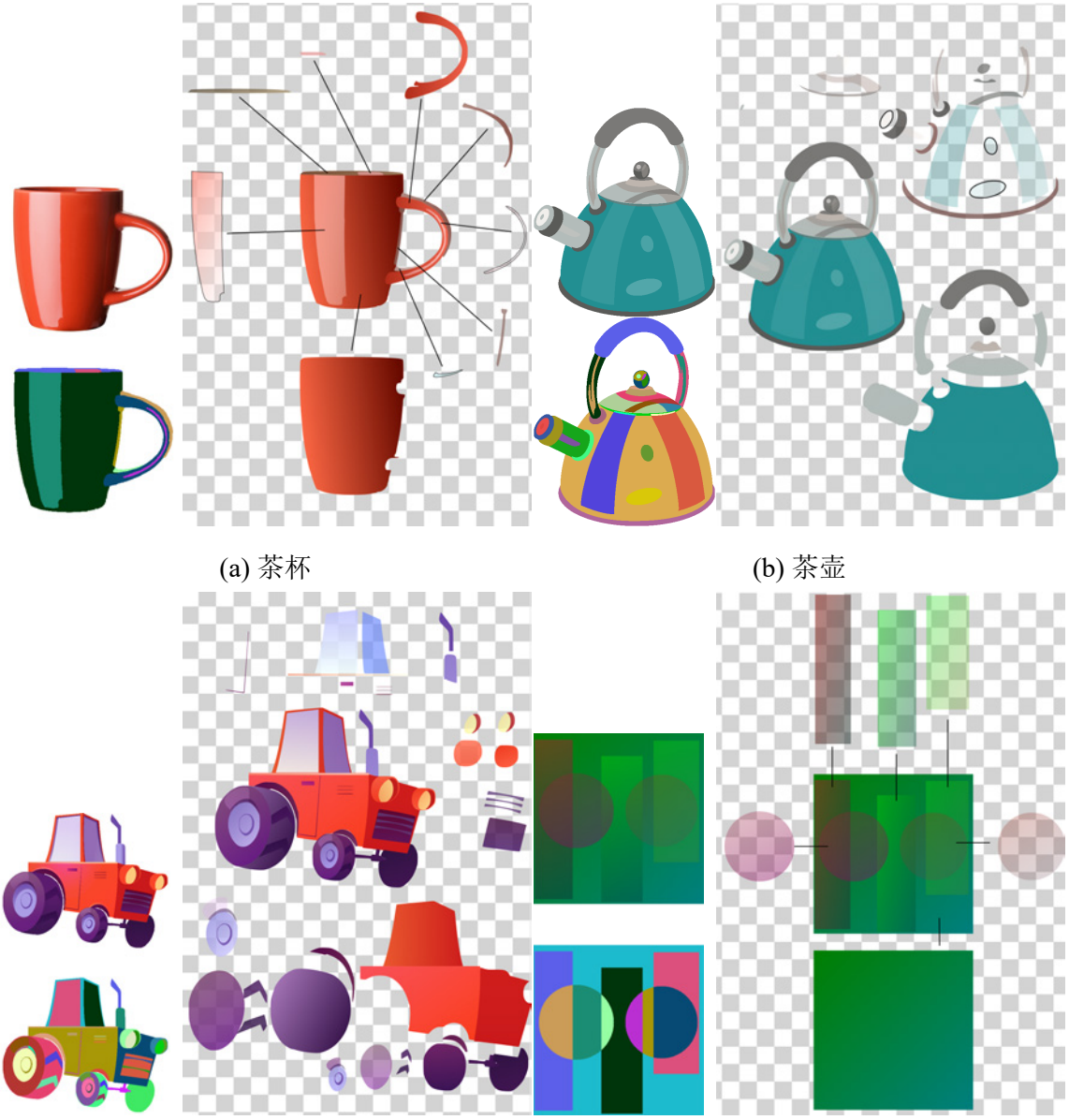
本章算法通过 C++ 语言实现，用到的第三方库有 NLopt[100]，autodiff[103]，Portrace[104]，OpenCV[105]。本文使用多线程技术并行计算每个区域支持树对应的图层参数，其余部分均采用单线程计算。

3.11.1　图层分解结果

图 3.13 和图 3.14 展示了一些图层分解的例子。针对每一个例子，左侧是输入（第一行是输入图像，第二行是输入图像的区域分割结果），右侧是重建图像及图层分解结果，这里使用棋盘格作为背景来突显图层的透明度。总体上，本文算法实现了高质量的图层分解。首先，分解的图层具有很高的重建质量，这些图层合成的图像在视觉上跟输入图像非常接近；其次，图层的颜色和不透明度均位于色域范围之内；最后，图层的覆盖顺序符合预期，面积较小的图层位于较大图层之上。

在图 3.13(a) 中，杯子被分解为多个图层，除了杯身和杯把等少量位于底部的

图层不透明外，大部分图层是半透明的，如杯身和杯口的高光部分都被成功地分离出来。图 3.13(b) 和图 3.13(c) 对应的输入分割图像分别包含 34 个区域和 45 个区域，并且包含一些 X 型交叉（图 3.13(b) 中茶壶盖上的阴影部分，图 3.13(c) 中车轮上的高光部分），提出的算法很好地恢复了 X 型交叉部分的各个图层以及它们的层叠顺序。图 3.13(d) 是一张合成图像，算法共分解出 6 个图层（位于底部的绿色背景图层，位于中间的三个矩形图层以及位于顶部的两个圆形图层），这些图层合成的图像跟输入图像相比具有较高的保真度，此外，图层的堆叠顺序也符合我们的预期，较大的图层位于底部，较小的图层位于顶部。



(a) 茶杯　(b) 茶壶

(c) 拖拉机　(d) 几何形状 1

图 3.13　图层分解结果 1

图 3.14(a) 所示的路障图像既包含高光效果又包含多个 X 型交叉，本文算法将路障的主体分解为一个底部的深黄圆锥图层，两个中间的银白图层和一个顶部的高光图层，不仅成功地分离出了完整的高光部分，同时很好地重建了图中的 4 个 X 型交叉，分解的图层以及它们的层叠顺序符合直觉。图 3.14(b) 所示的图像包含较多的阴影效果，算法将阴影部分分解为完整的图层，同时对其它部分进行了细粒度的图层分解。图 3.14(c) 所示的台灯和图 3.14(d) 所示的高跟鞋均包含明显的高光效果，本文算法成功地分离出了所有的半透明高光图层。

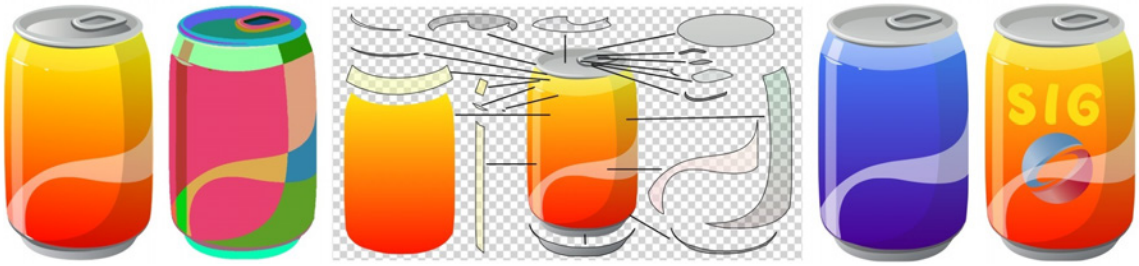
(a) 路障　　(b) 锤子

(c) 台灯　　(d) 高跟鞋

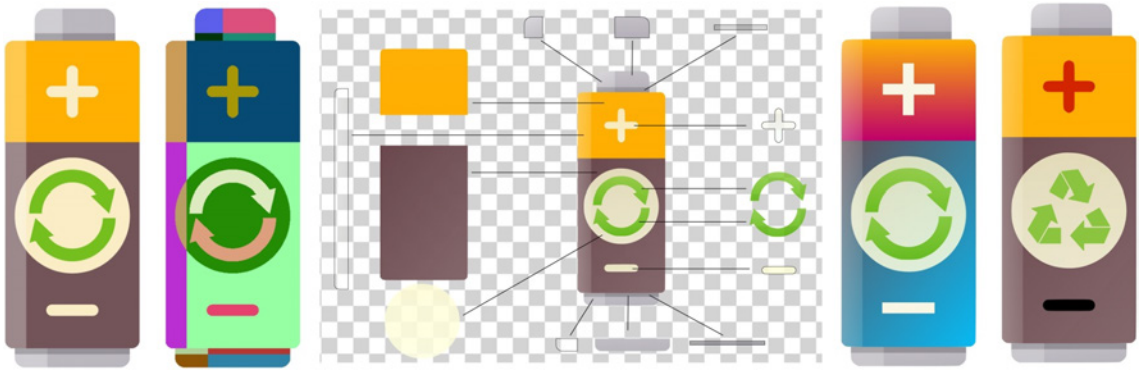
图 3.14　图层分解结果 2

矢量图支持高效、精确的颜色编辑。一旦将图像分解为多个半透明图层，那

么就可以通过修改图层的颜色、增加或删除图层来编辑图像。图 3.15 展示了两个矢量图的编辑效果。其中，第一列和第二列是输入的图像和对应的区域分割结果，第三列是图层分解结果，最后两列是编辑结果。针对图 3.15(a) 所示的易拉罐图像，本文算法分解的图层很好地反映了图像的层次结构，为用户编辑提供了便利。例如，用户通过简单地修改图 3.15(a) 第三列左下角的黄色图层就可以将易拉罐的整体色调从黄色变为蓝色，效果如图 3.15(a) 第四列所示。因为分解的图层透明度较高，因此也方便用户通过添加图层来合成有趣的效果，例如用户通过添加新的文字和图案合成了图 3.15(a) 最后一列的效果。类似地，对于图 3.15(b) 所示的电池图像，用户也通过修改图层的颜色或添加新的元素实现了高效的图像编辑。



(a) 易拉罐



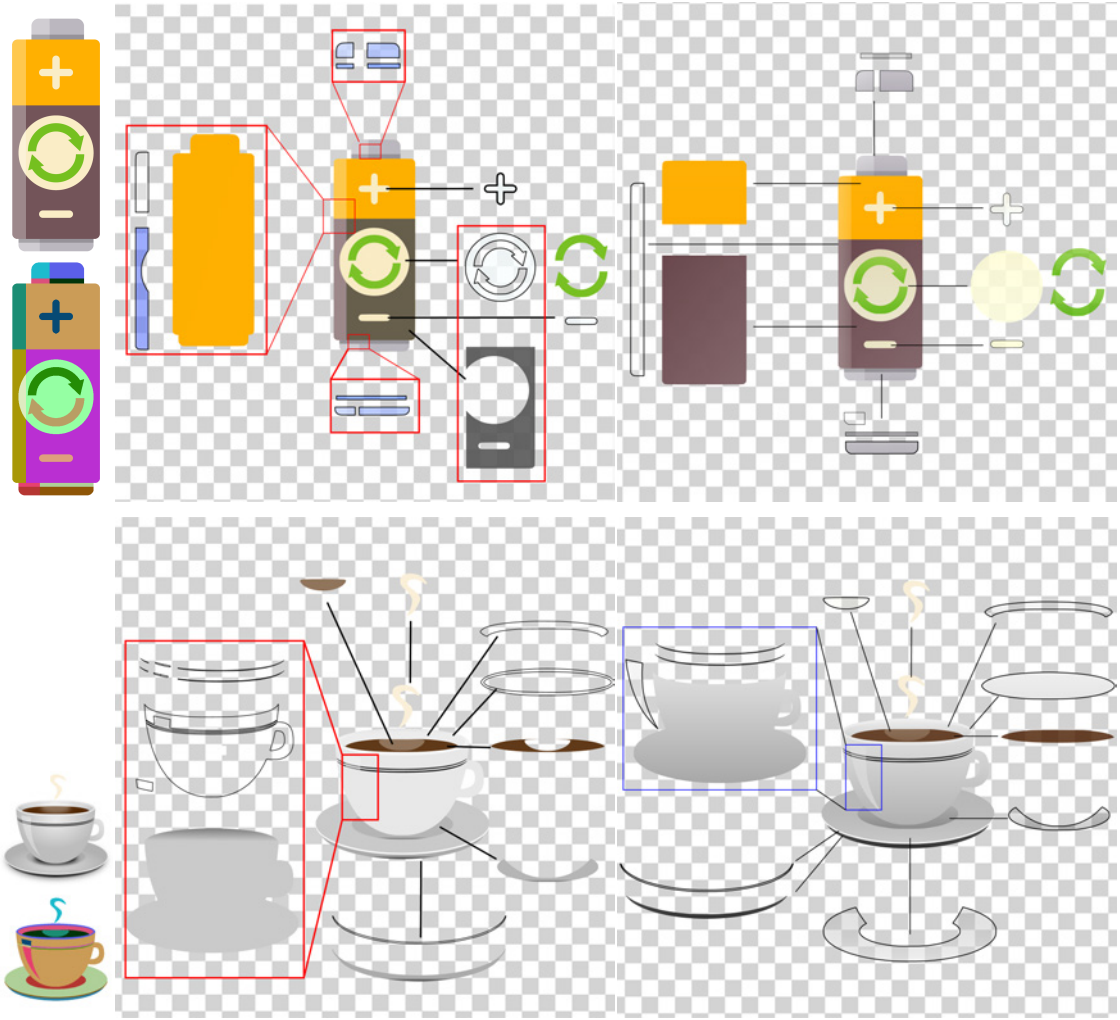
(b) 电池

图 3.15　矢量图编辑示例

3.11.2　方法对比

跟本章算法最相关的工作主要有两个：一是 Richard 等[8]于 2014 年提出的基于采样的半透明图层分解算法，二是由 Favreau 等[9]于 2017 年提出的基于蒙特卡洛树搜索的半透明图层分解算法。Richard 等[8]将半透明图层分解简化为单一前景图层分解问题，算法要求用户手动指定待分解的前景区域，然后通过采样确定前景区域边缘处的背景颜色，最后通过能量优化的方式求解前景图层的颜色和不透明度的线性渐变或径向渐变函数的参数。待前景图层分解后，用户可以继续对背景区域进行分解。该算法的主要问题在于其完全依赖用户交互，图像分解过程非

常耗时。Favreau 等[9] 对这一算法进行了改进，提出了半自动的半透明图层分解方案。作者将半透明图层分解问题形式化为一个联合优化问题，同时优化求解图层配置（图层数目，图层蒙板以及图层覆盖顺序）和图层参数（表示颜色和不透明度的线性渐变参数）。然后进一步将优化过程转化为树搜索的问题，并通过蒙特卡洛树搜索的方法压缩搜索空间。为了对问题进行简化，作者要求用户提供图像的分割区域作为辅助输入，并要求用户指定一些不透明区域为算法提供初始值。由于 Richard 等[8]提出的算法没有公开源代码，且该算法需要大量用户交互来选取待分解区域，因此本章算法将不与这一算法进行实验比较。而 Favreau 等[9]提供了算法的可执行程序，且输入跟本章算法一致，因此本章只与这一算法进行比较。

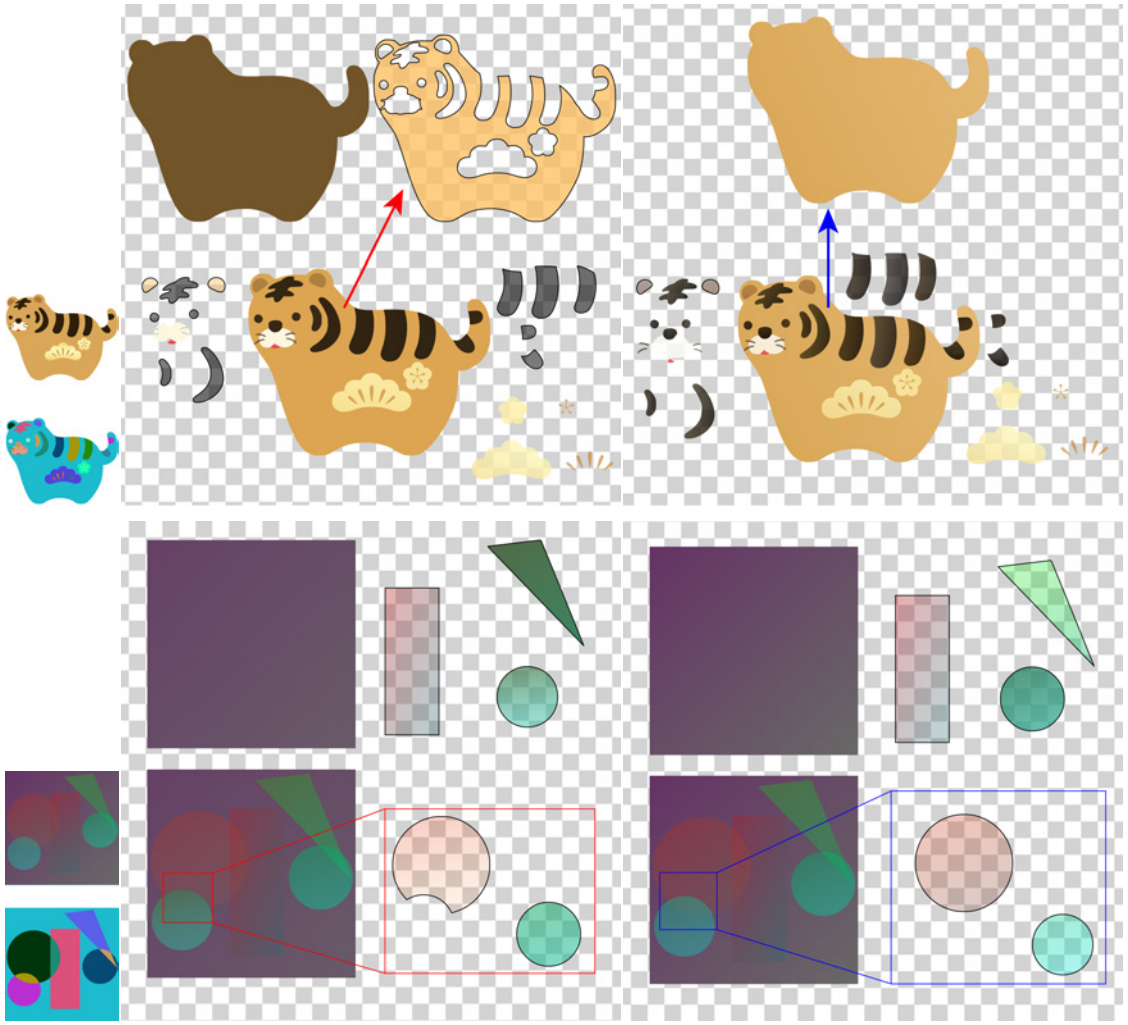


(a) 输入　　(b) Photo2ClipArt 矢量化结果　　(c) 本章方法矢量化结果

图 3.16　图层分解方法对比 1

我们在图 3.16和图 3.17中提供了四组例子将本章算法和 Favreau 等[9]提出的 Photo2ClipArt 算法的图层分解结果进行对比。对于每一组对比，第一列是输入，第二列是 Photo2ClipArt 的图层分解结果，第三列是本章算法的图层分解结果。

Photo2ClipArt 算法分解的图层通常包含不自然的孔洞，并且难以准确地分解出 X 型交叉涉及的图层的层次结构。如图 3.16 第一行所示的“电池”例子，右侧的红框包括的两个图层均包含孔洞，且图像中的几个 X 型交叉都被分解为多个独立的图层，没能还原出符合直觉的半透明图层及其空间结构。相反，本文算法分解的所有图层都不包含孔洞，且对于 X 型交叉部分的图层分解更为合理。如电池左侧黄色区域跟棕色区域接缝处的 X 型交叉，本文算法将其分解为下方的黄色和棕色图层，以及上方的高光图层，显得更为合理。图 3.16 第二行所示的“咖啡杯”例子中，两条黑色的曲线跟高光交互形成了多个 X 型交叉。Photo2ClipArt 算法难以还原出两条完整的黑色曲线，且不能获取高光部分的图层。相反，本章算法将此处的多个 X 型交叉分解为底部的杯身，中间的两条曲线以及顶部的高光，这几个图层的合成效果，更好地反映了咖啡杯局部的层次结构。



(a) 输入　(b) Photo2ClipArt 矢量化结果　　(c) 本章方法矢量化结果

图 3.17　图层分解方法对比 2

在图 3.17 第一行所示的“老虎”例子中，Photo2ClipArt 将老虎的黄色躯体分

解为棕色的背景图层和上方包含大量孔洞的图层，显得冗余且不自然，而本章算法分解出了完整的背景图层。在图 3.17 第二行所示的合成图像中，Photo2ClipArt 算法没能还原出左侧两个圆形的覆盖效果，且只分解出残缺的大圆，而本文算法成功地分解出两个完整的圆形图层（小圆置于大圆之上）。

3.11.3　消融实验

为了缩减区域支持树的搜索空间，本文使用了面积规则、包围规则和连接规则对初始的区域邻接图进行简化，排除了不合理的图层配置，并有效地提升了算法的搜索效率。为了对这些规则的有效性进行验证，这里分别统计了使用所有规则和仅禁用某一规则情形下，简化的区域邻接图的边数和枚举的所有区域支持树的数量。如表 3.1 所示，表中“en”表示初始区域邻接图的边数，“se”表示应用各规则简化后的区域邻接图的边数，“tn”表示从简化后的区域邻接图中枚举的区域支持树的数量。

表 3.1　禁用不同感知规则后的区域邻接图的边数及包含的区域支持树的数量对比

名称	en	启用所有规则		禁用面积规则		禁用包围规则		禁用连接规则	
		se	tn	se	tn	se	tn	se	tn
茶杯	35	14	10	17	18	14	10	19	194
茶壶	142	53	960	75	80,689	54	960	71	60,234
拖拉机	171	58	112	88	1,152	63	672	82	10,814
几何形状 1	38	16	8	27	16	17	8	17	8
路障	47	33	10	43	147	33	10	34	10
锤子	72	26	6	37	12	31	12	36	200
台灯	58	27	56	34	170	28	74	30	249
高跟鞋	50	21	46	33	120	21	46	27	182
易拉罐	109	39	64	69	120	41	64	50	2,304
电池	68	29	92	50	328	29	92	32	150
咖啡杯	75	33	96	53	96	35	144	37	432
老虎	114	40	16	52	64	40	16	49	512
几何形状 2	37	12	8	23	8	12	8	15	8

从统计结果来看，本文使用的这三个规则在区域邻接图的简化过程中发挥了重要作用。首先，面积规则对区域的支持关系进行了限定，不允许面积特别小的区域支持较大的区域，对于某些有较多邻居的区域来讲，去掉了很多不必要的连

接。如“茶壶”示例，在使用所有规则的情况下，简化的区域邻接图共包含 53 条边，并可以枚举出 960 棵区域支持树，但是当禁用面积规则后，区域邻接图的边迅速增长到 75 条边，同时区域支持树的数量暴涨到 80000 棵以上。其次，包围规则使得被包围的区域只能被包围它们的区域支持，采用该规则可以帮助排除了一些不符合直觉的图层覆盖关系。如在“拖拉机”例子中，使用所有规则时，区域支持树的数量为 112，禁用包围规则后，区域支持树的数量增长 5 倍。最后，连接规则去掉了关联较弱的区域之间的连接，在区域连接图的简化过程中几乎发挥了最重要的作用，从所有例子来看，该规则的简化效果最为显著。

表 3.2　算法时间性能统计

名称	rn	tn	ln	t_enum (s)	t_lm (s)	t_opt (s)	t_sum (s)
茶壶	34	960	30	45.6	0.2	849	894.8
茶杯	9	10	9	0.1	0.1	2.3	2.5
拖拉机	45	112	43	0.9	0.1	204.3	205.3
几何形状 1	10	8	6	0.1	0.1	3.9	4.1
路障	13	10	9	0.1	0.1	0.1	0.3
锤子	19	24	18	0.1	0.1	40.2	40.4
台灯	20	56	20	0.1	0.1	32.8	33
高跟鞋	12	46	10	0.1	0.1	11.8	12
易拉罐	25	64	20	0.5	0.1	65.8	66.4
电池	18	92	14	0.1	0.1	48.7	48.9
咖啡杯	19	96	13	0.1	0.1	48.7	48.9
老虎	32	16	28	0.1	0.1	23.6	23.8
几何形状 2	9	8	6	0.1	0.1	3.2	3.4

3.11.4　性能统计

表 3.2 展示了算法在各个阶段的时间开销。其中，“rn”表示输入分割图的区域数量，“tn”表示区域支持树的数量，“ln”表示最终的图层数量，“t_enum”表示枚举区域支持树消耗的时间，“t_lm”表示图层合并消耗的时间，“t_opt”表示求解图层参数消耗的时间，“t_sum”表示算法总共耗时，所有时间单位为秒。

总体上，算法的执行时间跟区域支持树的数量以及图层数量正相关。从提供的示例来看，对于输入图像分割区域少于 20 个的例子，本文算法基本都能在 1 分钟以内返回结果。对于较为复杂的例子如“茶壶”和“拖拉机”，因为包含较多的区域支持树，所以需要花费数分钟到十几分钟不等的时间进行处理。本文算法在

图层参数优化阶段耗时最多，每个图层有 9 个未知量待求解，对于一些图层较多的例子如“茶壶”包含 30 个图层，共有 270 个参数待优化，因此需要耗费较多时间。将来我们期望对这一过程进行优化和改进，以提升算法的运行效率。

3.11.5　用户实验

为了进一步评估本章提出的半透明图层分解算法的有效性，我们邀请了 32 名介于 19 岁至 32 岁之间的用户进行了一项用户实验。每个用户需要对 32 组图像的图层分解结果进行评估，每组结果包含一张输入图像和两个图层分解结果，一个由 Photo2ClipArt 生成，另一个由本章方法生成，并且以匿名和随机排列的方式展现给用户。如图 3.18 所示，针对每一个图层分解结果，我们提供了两种方式展示图层：一是将所有分解的图层展示给用户，二是允许用户交互地选取图层并观察图层的合成效果。对于每组结果，用户需要对以下三个问题进行投票：

- Q1（重建质量）：哪个图层分解结果更接近输入图像？
- Q2（形状一致性）：哪个图层分解结果更好地反映输入图像的形状结构？
- Q3（编辑方便）：哪个图层分解结果更方便用户编辑？

对于每一个问题，用户可以选择其中一个图层分解结果，也可以选择二者效果相当。总体上，用户实验得到了积极的反馈。对于这三个问题，本文方法分别在 32 个示例中的 27 个（84%）、30 个（94%）和 30 个（94%）中获得了更多的投票。用户实验结果充分表明，本文方法分解的图层在重建质量、形状一致性和编辑便捷性三方面整体优于 Photo2ClipArt 算法。

图 3.18　用户实验中图层分解结果的呈现方式

3.12　本章小结

本章提出一种面向图像矢量化的半透明图层分解算法，输入是一张光栅图像和其对应的区域分割图像，输出多个空间上相互重叠的半透明图层。相对于前人的方法，本文提出的方法完全无需用户交互，可以全自动地对给定输入图像进行图层分解。同时通过引入感知规则，使得分解的图层更加直观地反映了图像的层次结构，同时更方便用户编辑。

针对给定光栅图像的图层分解是一个挑战性的问题，不仅要预测图层配置（包括图层的数目、图层的蒙板以及图层的层叠顺序等在内的离散参数），还要预测图层参数（关于颜色和不透明度的线性渐变的连续参数）。通常，图层配置的计算更加困难，而图层参数相对简单。为此，本文提出的算法首先从输入图像推导出图层配置，然后通过优化的方式求解图层参数。关于图层配置，本文有两个重要观察：一是图层配置可以形式化地表示为一棵区域支持树，二是区域支持树又可以从区域邻接图中遍历生成树得到。基于这个观察，本文设计了整体的求解流程：1）根据给定的输入构建区域邻接图并简化；2）从区域邻接图枚举所有可能的区域支持树；3）从区域支持树导出初始图层配置并执行必要的合并操作；4）通过能量优化的方式求解图层参数；5）最后对所有图层分解结果进行排序并将能量损失最小的图层分解结果返回给用户。为了验证算法的有效性，我们将本文算法跟已有算法分解的图层进行了对比，同时通过邀请用户参与实验调查。实验结果表明本文算法分解的图层较已有算法更完整，更符合直觉。对包含 X 型交叉的图像，分解的图层更能反映图像本身的形状结构。

本文算法仍存在一些局限性有待进一步改进。首先，本文算法需要分割图像作为输入。这对于没有经验的用户来说是不容易的。近期基于深度学习的分割算法如 Kirillov 等[106]提出的 SAM 和 Wang 等[107]提出的 SegGPT 等通用分割模型取得了突破性的进展，将来我们希望在这些算法基础之上提出无需分割图像作为输入的图层分解算法，进一步降低用户的使用成本。其次，本文方法只支持线性渐变的图层。未来我们考虑引入其他类型的渐变函数，如径向或二次渐变函数。最后，多图层矢量化本质上是一个有歧义的问题。对于给定的输入，通常有许多可能的分解。本文方法无法枚举所有可能的分解。它只考虑了符合感知规则的分解方案。未来我们希望利用更多的感知规则来改善这一点。

第 4 章　面向图像内容感知重着色的加性图层分解

颜色编辑是图像处理的核心任务，涉及图像增强、图像彩色化、风格迁移、编辑传播、图像融合等一系列研究领域。本章重点关注一类基于调色板的图像重着色技术，针对给定的彩色图像，用户通过简单的交互即可实现直观、高效的颜色编辑。相对于风格迁移和编辑传播等经典颜色编辑技术，基于调色板的图像重着色技术既保证了足够的控制自由度又尽可能地降低交互的难度，相当于同时结合了风格迁移和编辑传播二者的优点，并弥补了它们各自的缺陷。基于调色板的图像重着色技术自提出以来，受到了相关领域研究者的广泛关注，一些代表性的技术如基于聚类的算法[11]和基于凸包的算法[12,39-40]陆续被提出。相关技术已经在艺术设计、动漫制作、服装设计、图像美化等领域广泛应用。

流行的图像重着色技术涉及两个主要的步骤即调色板提取和图层分解。当前这两个步骤都在 RGB 空间进行，所有操作只涉及图像的低级颜色信息，因此无法实现内容感知的颜色编辑。如一张图像中同时存在颜色相似的多个不同类别的物体，当用户修改其中一个物体的时候，其它物体的颜色也会随之发生变化。为了解决这个问题，本章提出一种面向图像内容感知重着色的加性图层分解算法，并将其应用到重着色任务中，以实现内容感知的语义级别的颜色编辑。

4.1　本章概述

本章提出一种面向图像内容感知重着色的加性图层分解算法，将给定的输入图像分解为一组包含语义信息的图层，并借此实现内容感知的图像重着色。相对于已有的基于聚类的算法[11]和基于凸包的算法[12,39-40]，提出的算法的主要贡献在于：1）结合图像的低级颜色特征和高级语义特征并在高维空间提取图像调色板，更好地反映了不同物体的颜色分布；2）基于提取的调色板，根据像素与调色板颜色的相似度对图像进行图层分解，分解的图层反映了图像的语义信息；3）应用到图像重着色任务中，实现了内容感知的、直观高效的颜色编辑。

本章后续部分按如下方式组织：首先对本文涉及的背景知识如基于聚类的图层分解技术、超像素分割、双边滤波和引导滤波、主成分分析等进行简要介绍；其次对算法的输入输出和求解的问题进行描述；然后对本章算法的两个主要步骤即调色板提取和图层分解进行全面剖析；接下来通过不同方法的图层分解结果、图像重着色结果等进行对比以验证本章算法的有效性；最后对本章算法进行简单总结并对可能的改进方向进行展望。

4.2　相关背景

本章算法建立在 Chang 等[11]提出的基于聚类的加性图层分解算法基础之上。类似地，本章也通过聚类的方法提取图像的调色板，然后通过径向基函数插值的方法实现图层分解。不同点在于本文将调色板提取算法和图层分解算法从 RGB 三维空间提升到高维空间，目的是为了更好地捕捉图像语义层面的信息，实现内容感知的图像重着色。为了更好地理解提出的算法，本文首先在这一节对背景知识和相关技术进行简要介绍。

4.2.1　基于聚类的图层分解

Chang 等[11]首次提出基于调色板的加性图层分解算法，提取的调色板很好地反映了图像中主要颜色的分布，实现了简洁高效的图像重着色。该算法首先将输入图像 I 投影到 RGB 三维空间，将图像视为 RGB 空间中的点集。然后通过改进的 K-means 算法对其聚类，并将收敛后的 k 个聚类中心作为图像的调色板 C。为了实现简单高效的图像重着色，算法进一步将图像分解为多个图层。颜色编辑过程中，用户通过修改调色板的颜色实现对图像颜色的调整。接下来，简单回顾一下该方法如何根据提取的调色板对输入图像进行图层分解。

图层的定义

给定输入图像 I 和对应的调色板 $C=\{C_1, C_2, \cdots, C_k\}$，针对每个调色板颜色 C_i，定义一个相似度函数 $S_i(x)$，表示某个像素的颜色 x 与 C_i 的相似程度。相似度函数的定义将在下一小节详细介绍。

接下来，通过给定的相似度函数将图像分解为多个图层。具体而言，每个相似度函数 $S_i(x)$ 对应一个图层 L_i，该图层中任意点 $\mathbf{p}$ 的颜色值定义为：

$$L_i^{\mathbf{p}}=\frac{S_i\left(I_{\mathbf{p}}\right)}{\sum_{j=1}^{k} S_j\left(I_{\mathbf{p}}\right)} \tag{4.1}$$

其中，$I_{\mathbf{p}}$ 表示输入图像中位置 $\mathbf{p}$ 处的颜色。从上式可以看出，图层颜色的取值范围为 $L_i^{\mathbf{p}} \in [0,1]$，因此可以简单地将每个图层看作一张单通道灰度图像。重着色过程中，用户通过修改调色板的颜色（$C \rightarrow C'$）间接地控制图像的颜色变化：

$$I'_{\mathbf{p}}=I_{\mathbf{p}}+\sum_{i=1}^{k} L_i^{\mathbf{p}}\left(C'_i-C_i\right) \tag{4.2}$$

可以看出，Chang 等[11]提出的方法将重着色后的图像表示为输入图像加上调色板颜色变化量的线性组合。为了对图像进行图层分解，首先需要定义相似度函数。接下来，我们介绍相似度函数的定义以及求解方法。

相似度函数

给定输入图像的调色板 $C=\{C_1,C_2,\cdots,C_k\}$，首先针对每个调色板颜色 C_j 定义一个径向基函数：

$$\phi(x,C_j)=\exp\left(-\frac{\left(x-C_j\right)^2}{2\sigma^2}\right) \tag{4.3}$$

其中，σ 是调色板中所有颜色对的平均距离。接下来，针对调色板中的任意颜色 C_i，根据上述径向基函数定义其对应的相似度函数 $S_i(x)$，表示某个像素颜色 x 与 C_i 的相似程度。$S_i(x)$ 定义为上述所有调色板颜色对应的径向基函数的线性组合：

$$S_i(x)=\sum_{j=1}^{k}\lambda_{i,j}\phi\left(x,C_j\right) \tag{4.4}$$

其中，$\lambda_{i,j}$ 表示待求解的系数。针对 $S_i(x)$，当 $x=C_i$ 时，$S_i(x)=1$；当 $x=C_{j\neq i}$ 时，$S_i(x)=0$。也就是说，每个调色板颜色跟自身最为相似（相似度为最大值 1），不同调色板颜色最不相似（相似度为最小值 0）。为此，根据上述条件，针对每个相似度函数可以构建一个线性方程组，这里以 $S_1(x)$ 为例，可以通过求解如下的线性方程组求得对应的权重：

$$\begin{cases}\lambda_{1,1}\phi\left(C_1,C_1\right)+\lambda_{1,2}\phi\left(C_1,C_2\right)+\cdots\lambda_{i,k}\phi\left(C_1,C_k\right) &= 1\\ \lambda_{2,1}\phi\left(C_2,C_1\right)+\lambda_{2,2}\phi\left(C_2,C_2\right)+\cdots\lambda_{2,k}\phi\left(C_2,C_k\right) &= 0\\ \cdots & \\ \lambda_{k,1}\phi\left(C_k,C_1\right)+\lambda_{k,2}\phi\left(C_k,C_2\right)+\cdots\lambda_{k,k}\phi\left(C_k,C_k\right) &= 0\end{cases} \tag{4.5}$$

可以看到，每个相似度函数关联 k 个线性方程组，因此，所有 k 个相似度函数将关联 k^2 个线性方程组，为了简化求解，统一将其写为如下矩阵乘法的形式：

$$\begin{bmatrix}\lambda_{1,1} & \lambda_{1,2} & \cdots & \lambda_{1,k}\\ \lambda_{2,1} & \lambda_{2,2} & \cdots & \lambda_{2,k}\\ . & . & . & .\\ \lambda_{k,1} & \lambda_{k,2} & \cdots & \lambda_{k,k}\end{bmatrix}\begin{bmatrix}d_{1,1} & d_{2,1} & \cdots & d_{k,1}\\ d_{1,2} & d_{2,2} & \cdots & d_{k,2}\\ . & . & . & .\\ d_{1,k} & d_{2,k} & \cdots & d_{k,k}\end{bmatrix}=\begin{bmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & \cdots & 0\\ . & . & 1 & .\\ 0 & 0 & \cdots & 1\end{bmatrix} \tag{4.6}$$

其中，$d_{i,j}=d_{j,i}=\phi\left(C_i,C_j\right)$。这里将左侧第一个矩阵用 Λ 表示，第二个矩阵用 D 表示。因为右侧为单位矩阵，所以 $\Lambda=D^{-1}$。Λ 矩阵的第 i 行表示 $S_i(x)$ 对应的系数。至此，所有调色板颜色对应的相似度函数 $S_1(x),S_2(x),\cdots,S_k(x)$ 被确定下来。结合式（4.1）即可将输入图像 I 分解为多个图层 $L_1,L_2,\cdots,L_k$，并进一步将其应用到图像重着色任务中。

4.2.2　图像的超像素分割

超像素是指颜色、纹理、语义等特征相似的一定数量的相邻像素的集合。超像素分割将输入图像分割为一组超像素的集合，目的是通过少量的超像素表示整个图像的特征，降低图像处理的规模和复杂度。

常见的超像素分割算法主要包括两类即基于图的方法[108-112]和基于聚类[113-117]的方法。基于图的方法将每个像素看作单独的节点，然后不断地合并相似的相邻像素，直到当前最相似的两个子图的相似度超过预设的阈值，算法停止。最终，将每个连通子图关联的像素集合作为一个超像素。基于聚类的方法通常将像素当作 RGBXY 五维或更高维空间的点集，然后对点集进行聚类，最终将属于某一簇的所有像素当作一个超像素。

本章用到的超像素分割方法为简单线性迭代聚类算法（Simple Linear Iterative Clustering，SLIC），由 Achanta 等[118]于 2010 年提出。该算法大致分为以下 5 步：

1. 将像素投影到 LabXY 空间，将每个图像视为该空间中的点集。
2. 初始化，在空间均匀采样 k 个聚类中心，每个聚类中心对应的网格边长为 $E=\sqrt{N/k}$。其中，N 为图像的像素总数。
3. 并将每个聚类中心移动到其 3×3 邻域中梯度最小的位置。
4. 对于每个聚类中心，在以它为中心的面积为 $2E\times2E$ 的区域为每个像素重新分配类标签。具体而言，该区域内每个像素点 $\boldsymbol{I}_i=(l_i,a_i,b_i,x_i,y_i)$ 到该聚类中心 $\boldsymbol{C}_j=(l_j,a_j,b_j,x_j,y_j)$ 的距离表示为：

$$d_{i,j}=\sqrt{\left(\frac{d_c}{m}\right)^2+\left(\frac{d_s}{S}\right)^2} \tag{4.7}$$

其中，m 是一个给定的常数，$d_c=\sqrt{\left(l_j-l_i\right)^2+\left(a_j-a_i\right)^2+\left(b_j-b_i\right)^2}$ 表示像素点 $\boldsymbol{I}_i$ 与调色板颜色 $\boldsymbol{C}_j$ 在 Lab 空间的距离，$d_s=\sqrt{\left(x_j-x_i\right)^2+\left(y_j-y_i\right)^2}$ 表示像素点 $\boldsymbol{I}_i$ 与聚类中心像素 $\boldsymbol{C}_j$ 在 XY 空间的距离，像素点 $\boldsymbol{I}_i$ 可能同时位于多个聚类中心的邻域，它的标签由到它距离最近的聚类中心决定。

5. 迭代地运行步骤 3 和 4，直到聚类中心不再移动，算法停止。最终，将标签相同的像素集合视为独立的超像素。

SLIC 是当前最流行的超像素分割算法。这种算法生成的超像素整齐、紧凑，能够很好地表示图像的边界特征。相比于现有其他超像素分割方法，SLIC 算法在时间性能、边界质量、紧凑度方面都更优秀。

本章使用 SLIC 算法对图像进行超像素分割，然后在分割的基础上对超像素进行聚类以快速地提取图像的调色板。

4.2.3　双边滤波和导向滤波

双边滤波

双边滤波（Bilateral Filter）是一种广泛应用的图像平滑方法。这种滤波方法同时考虑邻域像素的颜色相似度和空间相似度，在平滑过程中很好地保留了图像的边缘特征。具体而言，双边滤波将输入图像 I 中任意像素的滤波结果 $I'_{\mathbf{p}}$ 表示为：

$$I'_{\mathbf{p}} = \frac{1}{w_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_{\mathbf{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}\left(\left|I_{\mathbf{p}} - I_{\mathbf{q}}\right|\right) I_{\mathbf{q}} \tag{4.8}$$

其中，$\mathbf{q}$ 是 $\mathbf{p}$ 的邻域中的任意像素点。$G_{\sigma_{\mathbf{s}}}(\|\mathbf{p}-\mathbf{q}\|)$ 是一个用于度量两个像素空间相似度的高斯核函数，$G_{\sigma_{\mathrm{r}}}\left(\left|I_{\mathbf{p}} - I_{\mathbf{q}}\right|\right)$ 是一个用于度量两个像素颜色相似度的高斯核函数。$w_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma_{\mathbf{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}\left(\left|I_{\mathbf{p}} - I_{\mathbf{q}}\right|\right)$，用于对结果进行归一化。

从上式可以看出，任意像素点的平滑程度（上述两个高斯核函数的乘积）由它跟邻域像素的空间相似性和颜色相似性共同决定。位于边缘上的像素点，虽然跟周围其他非边缘的像素点在空间上很近，但颜色差异较大，导致平滑程度很微弱，因此很好地保留了边缘特征。

导向滤波

双边滤波并不稳定，在边缘附近可能出现梯度反转的问题。为了解决这个问题，He 等[119]于 2010 年提出导向滤波（Guided Filter）。算法输入为一张待滤波的图像 $\mathbf{p}$ 和一张引导图像 I，输出为滤波后的图像 $\mathbf{q}$。为了使得滤波后的图像尽可能保持引导图像的边缘特征，该算法在局部窗口中将滤波后的图像建模为引导图像的线性组合：

$$\mathbf{q}_i = a_k I_i + b_k, \forall i \in \omega_k \tag{4.9}$$

其中，ω_k 表示 $\mathbf{q}_i$ 所在的局部窗口，a_k 和 b_k 表示该窗口的两个常系数。除了保持滤波后图像的边缘信息跟引导图像一致，还期望滤波前后图像在视觉上尽可能接近。为此，进一步定义了如下的能量函数度量输入图像在滤波前后的差异：

$$E\left(a_k, b_k\right) = \sum_{i\in\omega_k} \left(\left(a_k I_i + b_k - \mathbf{p}_i\right)^2 + \epsilon a_k^2\right) \tag{4.10}$$

然后可以通过求极值的方法解析地得到每个局部窗口对应的系数 a_k 和 b_k，最终将其代入式（ 4.9）实现图像的滤波操作。

导向滤波通过引导图像对输入图像进行平滑操作，相对于双边滤波，导向滤波对边界的保持效果更好，同时具有更低的时间复杂度。图像中抽取的语义特征通常包含大量的噪声，本文通过引导滤波对其进行平滑，抑制不必要的噪声。

4.2.4　主成分分析

高维的观测数据通常包含大量的冗余信息，不便于观察和分析，为了抓住重要特征，通常需要使用降维的方法对观测数据进行预处理。主成分分析方法[120-122]（Principal Component Analysis，PCA）是一种常见的数据降维方法，广泛应用于图像压缩、人脸识别等各种图像图形处理任务。主成分分析的主要目标是将高维数据映射到低维空间，主要思想是在原始高维空间找到一组相互正交的坐标轴，并将原始的高维数据映射到新的坐标系。PCA 算法的输入是一组 n 维空间的数据 $X = \{x_1, x_2, \cdots, x_n\}$ 以及低维空间的维度 k，输出是低维空间的数据 $Y = \{y_1, y_2, \cdots, y_k\}$。PCA 算法有两种实现，一是基于特征值分解（EigenValue Decomposition，EVD），二是基于奇异值分解（Singular Value Decomposition，SVD）。基于特征值分解的算法主要分为如下几步：

1. 计算原始数据各维度的平均值，并用每一维度的原始数据减去该维度对应的平均值，使得修改后的数据各维度的均值为 0；
2. 计算协方差矩阵 $C = \frac{1}{n} X X^T$；
3. 计算协方差矩阵 C 对应的特征值与特征向量；
4. 将矩阵 C 最大的 k 个特征值对应的特征向量组合成矩阵 M；
5. 将原始 n 维数据映射到 k 维空间：$Y = MX$。

主成分分析将一组可能存在相关性的冗余特征映射到若干线性无关的坐标轴上，不仅有效地抽取了数据中的有用信息，同时降低了数据分析的复杂度。本章通过 PCA 算法对图像抽取的高维语义特征进行降维，并将抽取到的主要语义特征与图像中的低级纹理特征结合，最终生成图像的调色板。

4.3　问题描述

我们的目标是将输入图像分解为一组语义相关的加性图层，并通过这些图层实现内容感知的图像重着色。欲实现图层分解，需要首先从输入图像中提取调色板，然后再根据像素与调色板颜色的相似度将输入图像分解为一组语义相关的图层。应用到颜色编辑任务中，用户通过修改调色板的颜色实现对图像颜色的编辑。相比于现有算法，提出的算法可以实现物体级别的颜色编辑，对于颜色相似的多个物体可以有效地区分并分别着色，实现了良好的编辑局部性。总结起来，算法的输入和输出分别为：

- 输入：一张图像 I。
- 输出：一个调色板 $C = \{C_1, C_2, \cdots, C_k\}$，一组图层 $L = \{L_1, L_2, \cdots, L_k\}$。

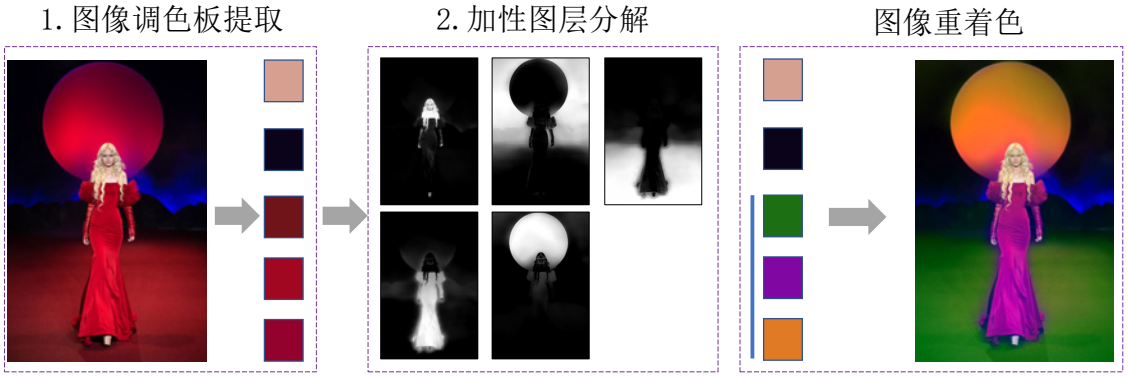
接下来，我们对提出算法的基本原理，算法设计等进行详细的描述，并通过对

不同算法分解的图层以及图像颜色编辑效果进行对比来验证本章算法的有效性。

4.4　算法总体流程

本文方法基于 Chang 等[11]提出的基于聚类的图层分解方法。与之不同的是，本章期望分解的图层尽可能包含必要的语义信息，并基于此实现内容感知的图像重着色。为了实现这个目标，本文将调色板的提取以及图层分解任务从 RGB 三维空间上升到同时包含颜色信息和语义信息在内的高维空间，使得抽取的调色板以及对应的图层能够反映图像的语义信息，从而实现语义级别的颜色编辑。

如图 4.1 所示，本章算法主要分为两个主要的步骤：一是调色板提取，二是加性图层分解。针对调色板提取，本章首先抽取输入图像的语义特征，然后将高级的语义特征与低级的颜色特征相结合，将图像看作高维空间的点集。接下来，在高维空间对该点集进行聚类，最终将收敛的聚类中心作为图像调色板。针对加性图层分解，首先在高维空间通过径向基函数拟合每个调色板颜色的相似度函数，然后根据相似度函数将图像分解为一组语义相关的图层。重着色过程中，用户通过修改调色板颜色，实现对图像的颜色控制。如图 4.1(d,e) 所示，用户通过修改调色板下方的三个颜色，对图像中同为红色的三个物体（裙子、地毯、圆形）分别实施了三种不同的颜色编辑。



(a) 输入图像　(b) 调色板　(c) 加性图层　(d) 修改的调色板　(e) 编辑结果

图 4.1　本章提出的语义图层分解算法流程

4.5　图像调色板提取

为了将图像分解为一组语义相关的图层，首先需要从输入图像中提取一个包含语义信息的调色板。本文通过预训练的神经网络抽取图像的高级语义特征，并将其与图像本身的颜色信息、空间位置信息等低级特征结合，将图像投影到高维

空间，最后通过聚类的方法提取图像的调色板。接下来，本节将从语义特征抽取和调色板提取两个步骤进行阐述。

4.5.1　语义特征抽取

本文使用 Aksoy 等[123]提出的深度卷积神经网络预测输入图像逐像素的语义特征。网络结构如图 4.2 所示，特征抽取部分基于 DeepLab-ResNet-101[124]实现，但为了最大化不同物体的特征的 L2 距离，网络采用度量学习[125]（Metric Learning）的方式进行训练。具体而言，输入图像首先经过 DeepLab-ResNet-101 网络进行特征提取；然后网络对多个级别的特征进行卷积、ReLU 激活、上采样等操作，并将上采样后的特征进行融合；最后通过多次下采样操作得到 128 个通道的逐像素特征图像。网络在 COCO-Stuff 数据集上进行训练，预测结果可以使得同类像素的特征尽可能相似，且不同类别像素的特征具有明显的差异。

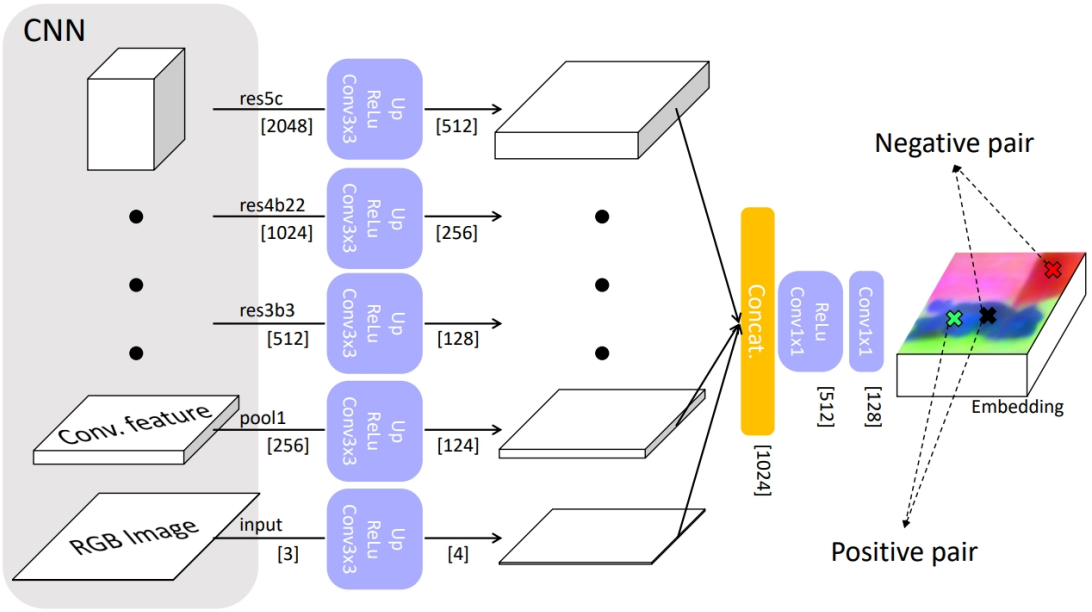
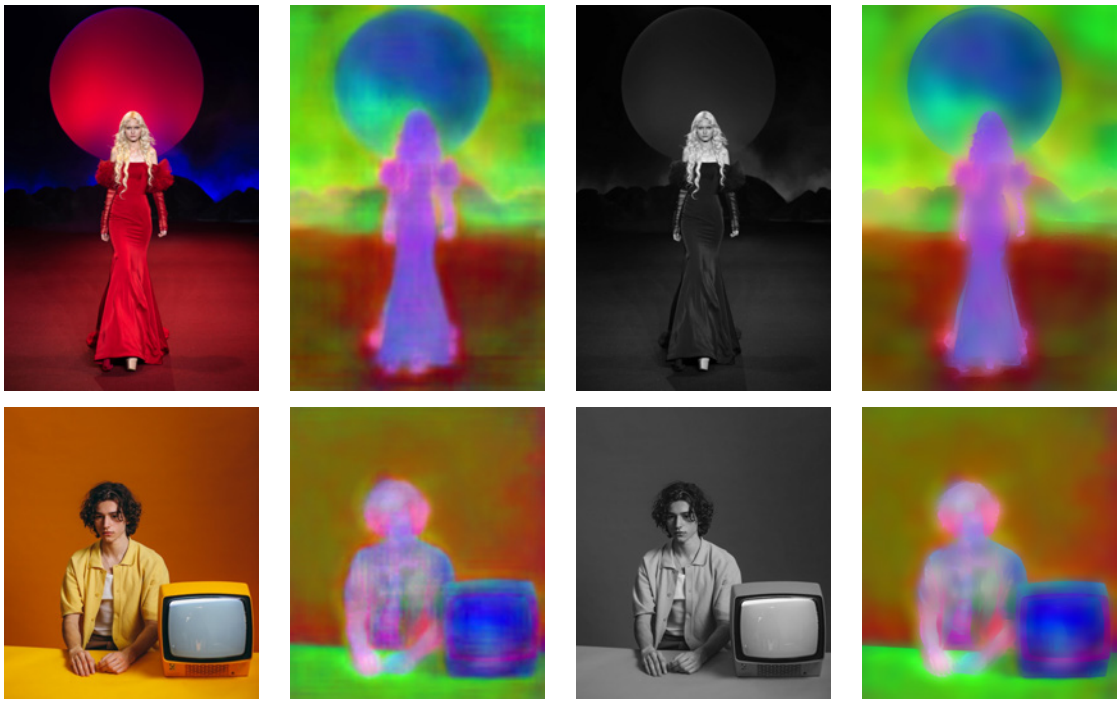


图 4.2　语义特征提取网络（图像来自于 Alsoy 等[123]）

本文使用上述预训练的模型对输入图像进行逐像素的语义特征抽取。抽取的特征图中每个像素包含 128 维的特征，为了降低计算开销并去除冗余数据，本文进一步使用 PCA[120]降维方法将 128 通道的语义特征图降到 3 通道，并将其看作一张三通道的 RGB 图像。我们注意到降维后的图像通常包含一些不必要的噪声，为此本文进一步通过导向滤波[119]的方法对抽取的特征图像进行平滑滤波。

图 4.3展示了两组示例。其中，图 4.3(a) 是输入图像，图 4.3(b) 是抽取的语义特征经 PCA 降维到三维后的可视化效果，图 4.3(c) 是输入图像的灰度图，它将作为导向滤波的引导图像，图 4.3(d) 是引导滤波后的结果。可以看到，深度卷积神经

网络抽取的语义特征符合直觉，经 PCA 降维后的特征大体反映了图像中不同物体的语义。初始的语义特征图像包含大量的噪声，引导滤波后的图像更加平滑，本文使用输入图像的灰度图像作为引导，很好地保留了特征图中不同物体的边界。



(a) 输入图像　(b) 初始语义特征　(c) 引导图像　(d) 引导滤波结果

图 4.3　图像语义特征及滤波结果可视化

4.5.2　调色板提取

为了从图像中抽取包含语义信息的调色板，本文首先将图像的高级语义特征和低级别的颜色信息和空间位置结合起来。为此，本文将图像中的每个像素表示为一个八维空间的点：$p_i = \left(r_i, g_i, b_i, x_i, y_i, f_i^1, f_i^2, f_i^3\right)$。其中，$\left(r_i, g_i, b_i\right)$ 表示像素的 RGB 颜色值，$\left(x_i, y_i\right)$ 表示像素在图像中的二维坐标，$\left(f_i^1, f_i^2, f_i^3\right)$ 表示像素的三个语义特征。为此，本文将输入图像当作八维空间的点集。接下来，我们通过聚类的方法计算输入图像的调色板。

本文采用改进的 K-means 算法对输入图像对应的八维点集进行聚类。为了降低计算复杂度，本文首先对输入图像进行超像素分割，然后将所有超像素的中心对应的八维空间的点作为采样点集合 $P = \{p_i\}$，K-means 算法只需对采样点聚类即可。K-means 算法对初值很敏感，为了得到较好的聚类结果，本文采用类似最远点采样的方式选取初始的 k 个聚类中心集合 C。

1. 对集合 P 中的任意像素点 p_i 赋予初始权重 $w_i = n_i$，n_i 表示 p_i 对应的超像素包含的像素数目；

2. 选取权重最大的像素点 p_i，将其加入到初始聚类中心集合 C；
3. 更新其它候选像素点 p_j 的权重：

$$w_j = \left(1 - \exp\left(-d_{i,j}^2 / \sigma_\alpha^2\right)\right) \cdot w_j \tag{4.11}$$

其中，σ_α 表示衰减参数。$d_{i,j}$ 表示 p_i 与 p_j 的距离，二者距离越大，权重衰减越小，后续被选中的可能性越大；

4. 返回 2，直到 k 个初始聚类中心全部选取到。

接下来，我们利用选定的初始聚类中心集合 C，使用 K-means 算法对采样像素点集 P 进行聚类。聚类过程中，任意像素点 p_i 到聚类中心 C_j 的距离表示为：

$$d\left(p_i, C_j\right) = \theta_c d\left(p_i^{RGB}, C_j^{RGB}\right) + \theta_p d\left(p_i^{XY}, C_j^{XY}\right) + \theta_s d\left(p_i^F, C_j^F\right) \tag{4.12}$$

上述距离函数包括三项。其中，$d\left(p_i^{RGB}, C_j^{RGB}\right)$ 表示 p_i 与 C_j 颜色分量的欧氏距离，$d\left(p_i^{XY}, C_j^{XY}\right)$ 表示二者坐标分量的欧氏距离，$d\left(p_i^F, C_j^F\right)$ 表示二者三个语义特征分量的欧氏距离。θ_c，θ_p 和 θ_s 分别表示这三项的相对贡献。

K-means 算法收敛后的聚类中心 C 即为要求的调色板。需要注意的是，这里的调色板不再是只包含颜色信息的狭义的调色板。当前调色板是一个八维向量集合，本文将调色板中的每一个元素称为一个条目。提取的调色板不仅包含颜色信息，还包含位置以及语义信息。也正因为此，我们才能进一步将输入图像分解为一组语义相关的图层，并在此基础上实现语义级别的图像编辑。

4.6　加性图层分解

接下来，本节介绍如何根据提取的调色板将输入图像分解为一组包含语义信息的加性图层。如 4.2.1小节所述，图层分解的核心在于定义调色板的相似度函数 $S_i(x)$。为此，本文首先在调色板 $C = \{C_i\}$ 中每个条目对应的八维顶点处定义径向基函数：

$$\phi(x, C_j) = \exp\left(-\frac{\left(x^{RGB} - C_j^{RGB}\right)^2}{2\sigma_c^2}\right) \exp\left(-\frac{\left(x^{XY} - C_j^{XY}\right)^2}{2\sigma_p^2}\right) \exp\left(-\frac{\left(x^F - C_j^F\right)^2}{2\sigma_s^2}\right) \tag{4.13}$$

上式包含三项，其中，第一项表示像素 x 与调色板 C_j 在颜色上的距离，第二项表示二者坐标距离，第三项表示二者在语义特征上的距离。σ_c，σ_p 和 σ_s 分别表示颜色、坐标和特征项对应的高斯核函数的标准差，一般通过计算所有调色板条目对的平均颜色、平均坐标和平均特征距离得到。以上公式表明像素点到调色板条目

的距离由他们的颜色距离，位置距离和特征距离三者共同决定。

接下来，本文仿照 Chang 等[11]提出的方法，将相似度函数定义为上述各个径向基函数的线性组合：$S_i(x)=\sum_{j=1}^{k}\lambda_{i,j}\phi\left(x,C_j\right)$（式（4.4））。类似地，针对 $S_i(x)$，当 $x=C_i$ 时，$S_i(x)=1$；当 $x=C_{j\neq i}$ 时，$S_i(x)=0$。构建式（4.5）所示的线性方程组并进一步求解每个相似度函数对应的多个权重 $\lambda_{i,j}$。最后，根据相似度函数计算图像中每个像素到调色板的相似度并进行归一化：$L_i^{\mathbf{p}}=S_i\left(I_{\mathbf{p}}\right)/\left(\sum_{j=1}^{k}S_j\left(I_{\mathbf{p}}\right)\right)$（式（4.1）），将图像分解为一组图层 $L=\{L_1,L_2,\cdots,L_k\}$。因为调色板包含语义信息，所以根据相似度函数分解的图层语义相关。在重着色任务中，我们只将调色板的 RGB 颜色展示给用户，用户通过修改调色板颜色调整图像的颜色，编辑后的图像表示为：$I'_{\mathbf{p}}=I_{\mathbf{p}}+\sum_{j=1}^{k}L_i^{\mathbf{p}}\left(C_i-C'_i\right)$（式（4.2））。当用户修改调色板的某个颜色时，只有那些跟它语义上接近（属于同一类物体），颜色相似，位置接近的像素的颜色才会发生变化，因此实现了很好的局部控制。

4.7　基于相似度量的位姿估计

上一节通过定义调色板的相似度函数将输入图像分解为一组语义相关的加性图层。此外，我们也将这种基于相似度量的方法应用到相机位姿估计任务中，实现了动态场景下的精确位姿估计。

相机位姿估计是即时定位与地图构建（Simultaneous Localization and Mapping，SLAM）技术的核心。现有的 SLAM 技术如 PTAM[126]，ORB-SLAM2[127]，DTAM[128]，InfiniTAM[129]，SVO[130]等大多基于静态场景，无法很好地适应真实的动态场景。一些基于动态场景的 SLAM 技术如 DVO[131]，BaMVO[131]等对于动态特征点的检测不够鲁棒，无法实现精确的位姿估计。还有一些基于动态三维重建的技术如 DynamicFusion[132]，MaskFusion[133]，StaticFusion[134]等需要解决非刚性的点云配准问题，计算开销较大。

我们提出一种轻量级的动态场景位姿估计算法。本文算法从粗到细分为两步：1）初始位姿估计，通过 Graph-Cut RANSAC[135] 计算初始的相机位姿并给出特征点初始分类（动态或静态）。2）动态路标点检测，通过相似度量的方法检测动态路标点和静态路标点，并通过静态路标点实现精确的位姿估计。针对动态路标点检测任务，我们构建了一个稠密的全连接的条件随机场[136-137]（Conditional Random Field，CRF），并通过最小化如下的 Gibbs 能量函数给出每个路标点的标签（静态：$x_i=0$，动态：$x_i=1$）：

$$E(X)=\sum_{i}\psi_u\left(x_i\right)+\sum_{i<j}\psi_p\left(x_i,x_j\right) \tag{4.14}$$

其中，$\psi_u\left(x_i\right)$ 是一元势函数，表示每个路标点的能量，由每个路标点的静态似然决定：

$$\psi_u\left(x_i\right)=\begin{cases}-\log(c)1\left(P_i^s>t\right) & \text{若 } x_i=0\\ -\log(1-c)1\left(P_i^s<t\right) & \text{若 } x_i=1\end{cases} \tag{4.15}$$

其中，c 是置信度，P_i^s 表示第 i 个路标点的静态似然，t 是给定的阈值，$1(\cdot)$ 表示指示函数，如果括号内的条件为真则函数值为 1，否则为 0。

$\psi_p\left(x_i,x_j\right)$ 是成对势函数，表示任意两个路标点对的能量。我们认为特征相似的路标点更有可能同属于静态或动态点。为此，我们设计成对势函数来惩罚特征相似却分配到不同标签的特征点对。设计的二元势函数包括两项用高斯函数定义的相似度函数，第一项用于度量观测统计相似度，第二项用于度量两个路标点的位置相似度。二元势函数的具体形式如下：

$$\psi_p\left(x_i,x_j\right)=\omega_1\exp\left(-\frac{\left|\alpha_i-\alpha_j\right|^2}{2\sigma_\alpha^2}-\frac{\left|\beta_i-\beta_j\right|^2}{2\sigma_\beta^2}\right)+\omega_2\exp\left(-\frac{\left|P_i-P_j\right|^2}{2\sigma_P^2}-\frac{\left|p_i-p_j\right|^2}{2\sigma_p^2}\right) \tag{4.16}$$

其中，α_i，β_i，P_i，p_i 分别表示路标点 i 的平均重投影误差，平均观测数量，3D 坐标，对应特征点的 2D 坐标。σ_α，σ_β，σ_P，σ_p 分别表示这几个量的标准差。

(a) 初始动态点检测结果　　(b) CRF 优化后的动态点检测结果

图 4.4　动态路标点的检测示例

通过在所有路标点上构建 CRF 并最小化 Gibbs 能量函数，场景中的动态路标点得以被精确地检测。如图 4.4 所示，第一列为初始位姿估计得到的检测结果，第二列为使用 CRF 之后的检测结果。其中，动态特征点标记为红色，静态特征点标记为绿色。可以发现，CRF 优化后动态点的检测更为准确。最后我们将动态路标点丢弃，并用静态路标点来估计相机的位姿。我们在 TUM[138]和 Bonn[139]动态数据集上对不同方法进行了对比，结果发现本文方法比现有大多数动态位姿估计方法更加精确。

4.8　实验

本节通过实验对面向图像内容感知重着色的加性图层分解技术的有效性进行验证。主要包括图层分解结果对比，图像重着色结果对比，参数评估三部分。本章实验环境为：

- 操作系统：Windows 10；
- 处理器：Intel i9-9900K 3.6 GHz CPU、16 核；
- 内存：16 GB RAM；
- 编程环境：Microsoft Visual Studio 2019，Pycharm 2020.2.4。

本文使用 Aksoy 等[123]提供的 Python 程序从输入图像抽取语义特征。通过 C++ 语言实现调色板提取和图像重着色功能。另外，我们还使用 Qt 编写了一个图像重着色的图形用户界面。

4.8.1　图层分解结果对比

本小节将本章提出的图层分解算法跟几种常见的图层分解算法进行对比，分别是 Chang 等[11]提出的基于聚类的方法，Tan 等[39]提出的基于凸包的方法，和 Wang 等[40]提出的基于凸包优化的方法。我们展示了每种方法提取的调色板以及对应的图层。图层中每个像素的取值介于 0 到 1 之间（0 是黑色，1 是白色），因此本节使用单通道的灰度图对图层进行可视化。每个图层跟调色板中的一个颜色相关联（调色板从上到下的顺序跟图层从左到右的排列顺序一一对应）。当修改调色板中的某个颜色时，这个颜色对应的图层中非零像素的颜色均会发生变化。我们期望图层具有良好的稀疏性并尽可能包含语义信息。这样一来，编辑调色板中的某个颜色时，只有最相关的像素的颜色发生变化，方便用户实施针对性的局部编辑。

如图 4.5 所示，左侧是输入图像，右侧是几种算法提取的调色板和分解得到的图层。前三行分别对应 Chang 等[11]，Tan 等[39]和 Wang 等[40]提出的算法的图层

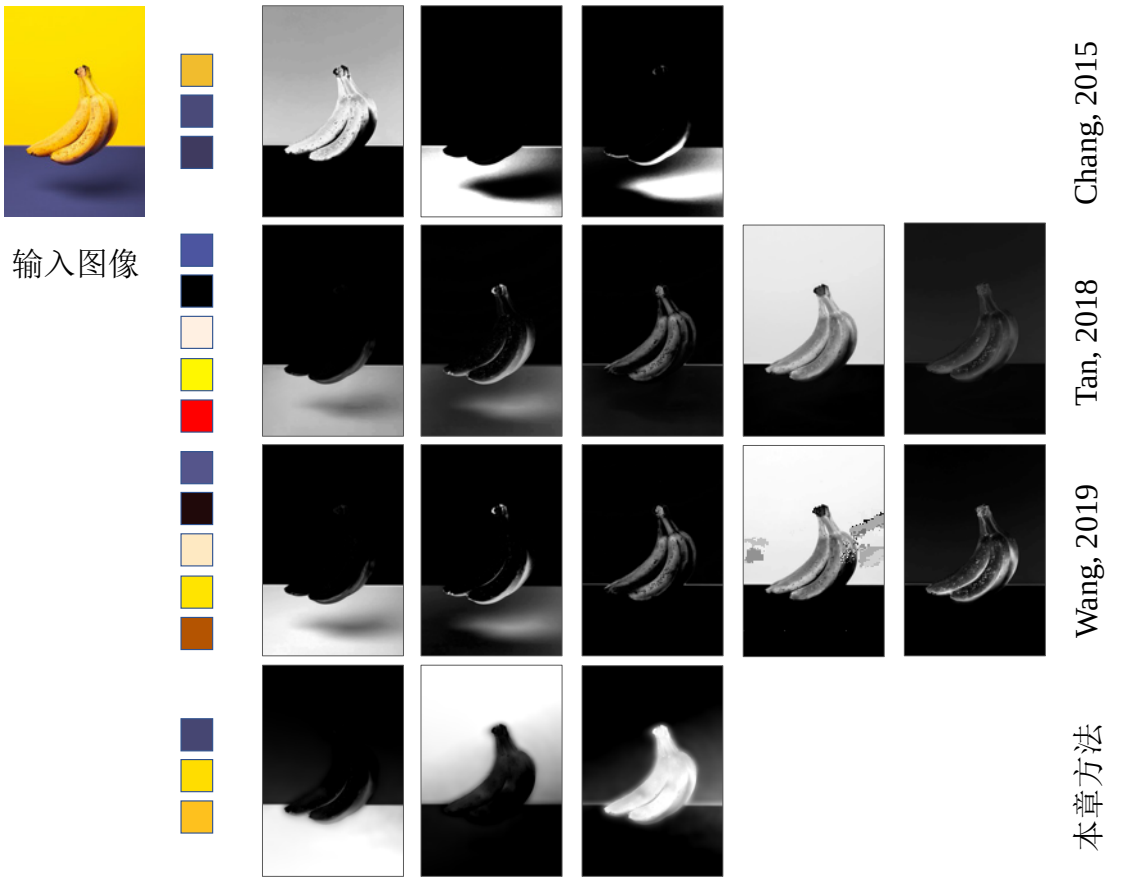


图 4.5　Chang 等[11]，Tan 等[39]，Wang 等[40]和本章方法分解的图层对比 1

分解结果，最后一行是本文算法的图层分解结果。在这个例子中，香蕉和背景颜色非常相似，几种方法都无法将香蕉和背景有效地分离，如图中第 1 行的第 1 个图层，第 2 行和第 3 行的第 4 个图层所示。针对输入图像，本文方法提取到包含三种颜色的调色板，并将输入图像分解为三个图层分别关联到图中的蓝色背景部分、黄色背景部分以及黄色香蕉部分，很好地表达了图像中不同物体的语义。本文方法提取的调色板包含两种黄色，这使得用户在重着色过程中可以分别对黄色的香蕉和黄色的背景分别进行颜色编辑且互不影响。而其余三种方法因为无法将香蕉和背景部分有效分离，因此在颜色编辑过程中必然无法单独对香蕉和背景部分进行操作。

图 4.6 展示了另一个图层分解的例子。在这个例子中，背景部分跟女士的衣服具有相似的颜色。Chang 等[11]分解的图层无法将女士的衣服跟墙面区分开（如第 1 行的第 3 个和第 4 个图层）。Tan 等[39]分解的图层同样如此，如第 4 行的第 1 个和第 2 个图层所示，女士的衣服跟地面和墙面混在一起。Wang 等[40]分解的图层同样不能有效地区分女士的衣服跟墙面部分，如第 5 行第 1 个图层所示。这些方

法使得重着色过程中，无法单独对墙面或女士的衣服进行颜色编辑。本文方法分解得到 4 个图层，包括干净的墙面部分，地面部分，中间人物部分以及家具部分。这些图层很好地表示了图像中不同物体的语义信息，方便用户针对不同的物体实施不同的颜色编辑。

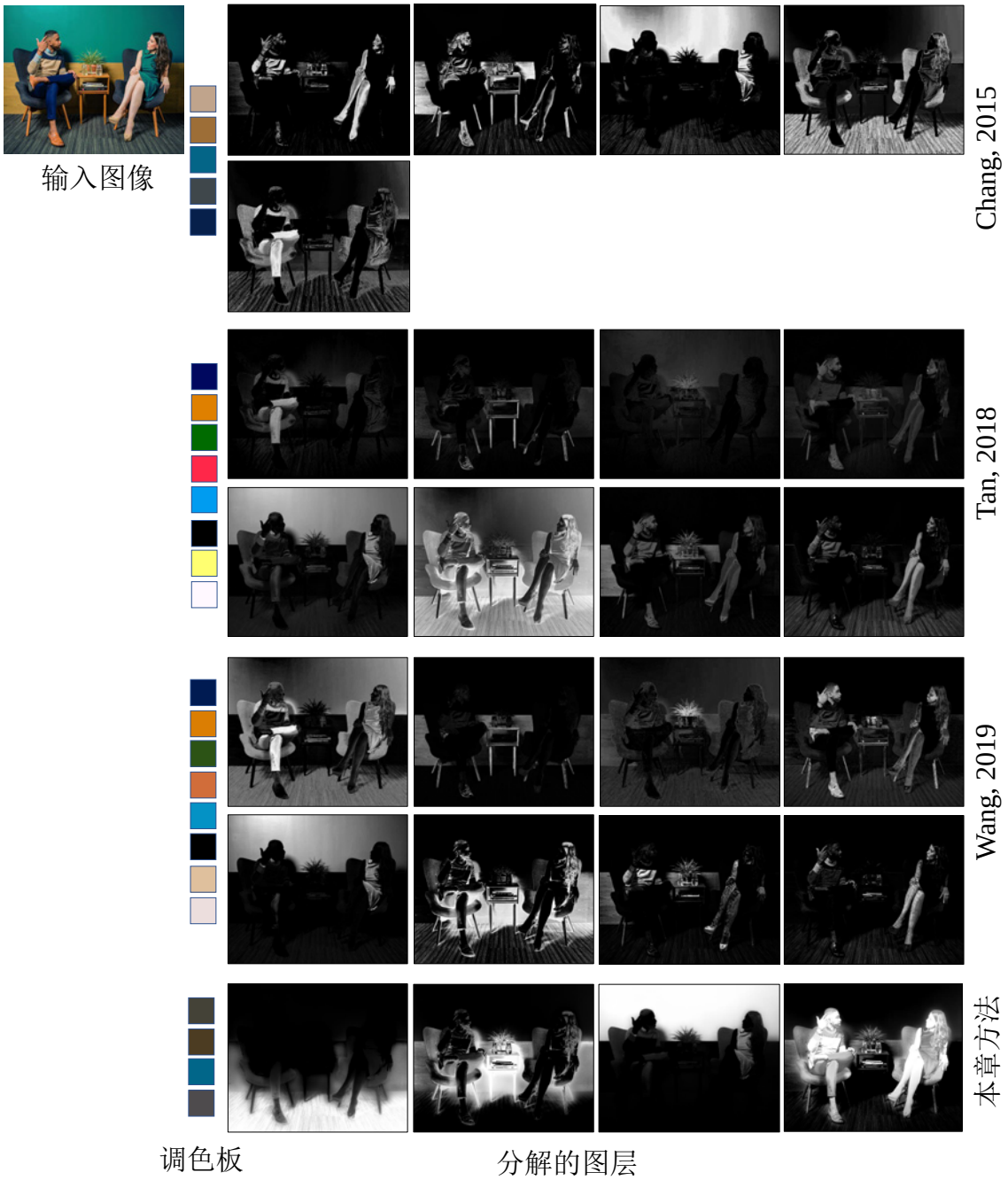


图 4.6　Chang 等[11]，Tan 等[39]，Wang 等[40]和本章方法分解的图层对比 2

4.8.2　图像重着色结果对比

这一小节重点对几种算法的重着色效果进行对比。本节提供了 5 组示例，针对每组示例，我们首先给定两个编辑任务，然后使用几种算法分别对图像进行编

辑，最后对这几种算法的重着色结果进行对照展示。

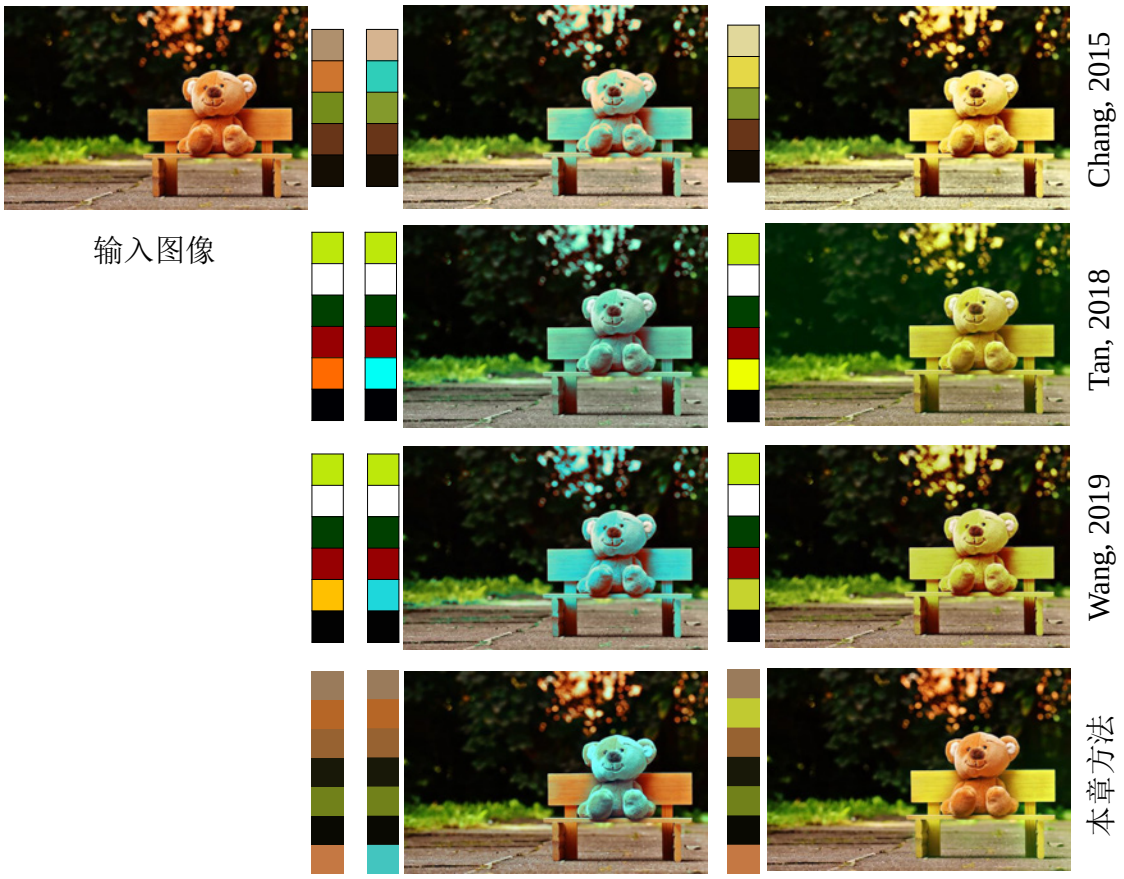


图 4.7　Chang 等[11]，Tan 等[39]，Wang 等[40]和本章方法的重着色结果对比 1

如图 4.7所示的例子中，玩具熊、椅子以及上方的光影具有相似的黄色，分别对这些物体进行颜色编辑具有很大的挑战性。给出的两个编辑意图分别为：1）将玩具熊的颜色修改为淡蓝色，2）将椅子的颜色修改为鲜黄色。图中第一列是输入图像，第二列是几种方法提取的调色板，随后是每种方法根据编辑意图修改后的调色板和重着色结果。图中从上到下分别是 Chang 等[11]，Tan 等[39]，Wang 等[40]和本文算法的重着色结果。Chang 的方法在修改玩具熊或椅子的同时，修改了图中其他部分的黄色物体。Tan 和 Wang 的方法提取的调色板只包含一种黄色，同样无法将玩具熊和椅子有效分离并分别着色。本文算法提取的调色板包含多种黄色，分别对应图中的玩具熊，椅子和光照部分。为此，用户可以针对性地修改玩具熊和椅子的颜色，而且不会对图中的其他无关部分造成影响。

图 4.8所示的例子中，小狗跟床单具有相似的棕色。给出的两个编辑意图分别为：1）将床单的颜色修改为紫色，2）将小狗的颜色修改为深棕色。可以发现，其它三种算法无法完成这两个编辑意图，不能将小狗和床单有效区分并分别着色，修

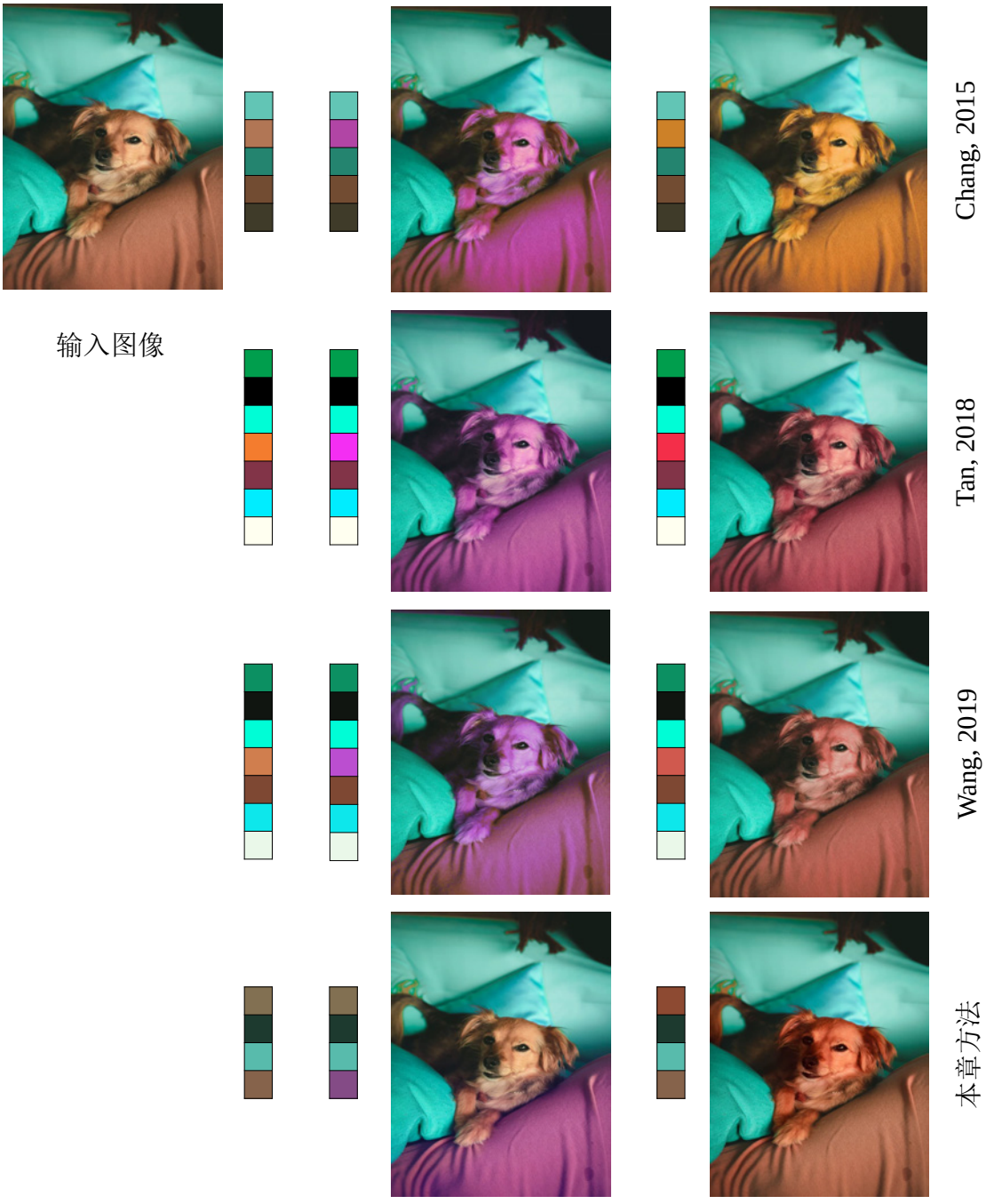


图 4.8　Chang 等[11]，Tan 等[39]，Wang 等[40]和本章方法的重着色结果对比 2

改其中一个物体必然导致另一个物体的颜色一同发生变化。本章算法提出的调色板包含两种棕色且分别对应床单和小狗，因此当用户将调色板中最底部的棕色修改为紫色时，对应的床单变成了紫色；当用户将调色板中最顶部的棕色修改为深棕色时，图中小狗的颜色变成了深棕色，而其它无关区域均不受影响。

图 4.9　Chang 等[11]，Tan 等[39]，Wang 等[40]和本章方法的重着色结果对比 3

针对图 4.9所示的例子，给定的两个编辑意图分别为：1）将桌面的颜色修改为白色，2）将背景颜色修改为红色。针对编辑意图一，其它三种算法在修改桌面的时候都不同程度修改了男士的衣服颜色以及电视的颜色，而本文方法在调整桌面的颜色时尽可能地保留了男士衣服以及电视机本身的颜色。针对编辑意图二，其它三种算法在修改背景部分的颜色时，同样对电视机，男人的脸部，甚至整体的色

调都产生了影响。而本文算法很好地完成了编辑意图二，在修改背景颜色的时候没有影响到其它的无关部分。

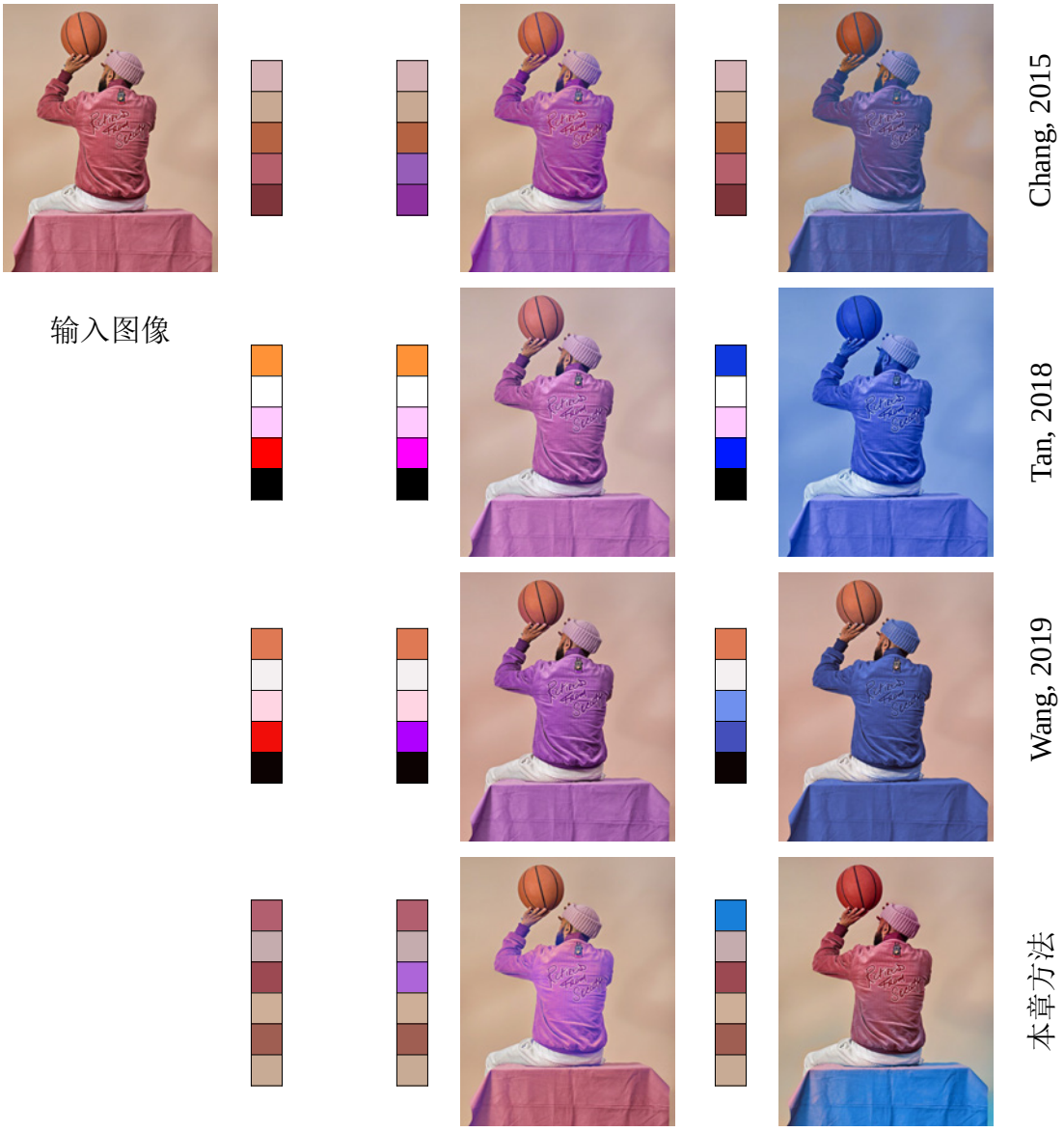


图 4.10　Chang 等[11]，Tan 等[39]，Wang 等[40]和本章方法的重着色结果对比 4

针对图 4.10所示的例子，给定的两个编辑意图分别为：1）将男士的衣服颜色修改为紫色，2）将桌布颜色修改为蓝色。其他三种算法在修改衣服（或桌布）的颜色时，都导致桌布（或衣服）的颜色发生变化，要么同时变为紫色，要么同时变为蓝色，无法做到单独编辑。如第四行所示，本文算法很好地完成了编辑任务，成功地将衣服的颜色修改紫色，将桌布的颜色修改为蓝色。

针对图 4.11所示的例子，给定的两个编辑意图分别为：1）将天空变得更蓝，2）将海面的颜色变得泛黄。输入图像中海天一色，传统的方法很难将二者有效地区

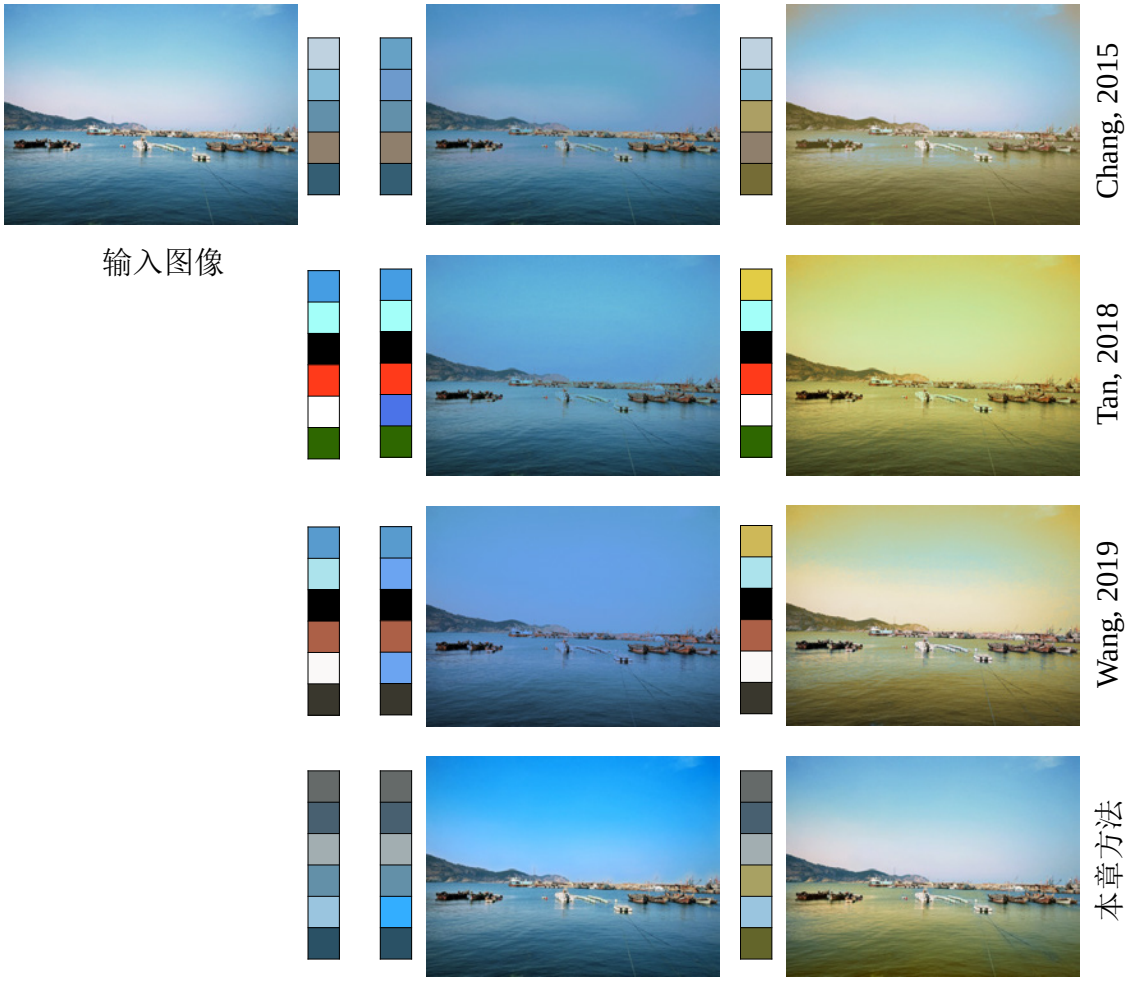


图 4.11　Chang 等[11]，Tan 等[39]，Wang 等[40]和本章方法的重着色结果对比 5

分。如前三行所示，传统方法修改天空时，大海的颜色相应地改变；修改海面的颜色时，天空的颜色也变得泛黄。而本文算法成功地将天空变得更蓝，将海面变得泛黄，且编辑效果直观、自然。

通过上述 5 组对比可以发现，本文提出的方法在重着色方面具有天然的优势。这得益于我们将图像的语义特征与低级的颜色信息、空间位置等结合起来，使得提取的调色板包含了足够的语义信息。进一步，通过相似度函数度量像素跟调色板颜色的相似性，在重着色过程中，只有跟调色板条目颜色相似、位置接近、语义相近的像素才会发生颜色变化。本文方法可以有效地将颜色相似语义不同的物体进行区分，并分别进行颜色编辑。而这对于传统的基于聚类的和基于凸包的图像重着色方法来讲几乎是不可能完成的任务。

4.8.3　参数评估

调色板提取阶段，本文通过改进的 K-means 算法对高维空间的采样像素进行聚类，最终将收敛的聚类中心作为图像的调色板。在 K-means 算法中，任意像素点到聚类中心的距离由二者的颜色距离、坐标距离和语义特征距离三者加权得到。本小节我们对该距离度量函数（式（ 4.12））中的三个权重系数 θ_c（颜色距离的权重），θ_p（坐标距离的权重）和 θ_s（语义特征距离的权重）进行评估。我们展示了不同参数设置下的调色板以及对应的图层分解结果，用以对参数进行评估。

图 4.12给出了一个参数评估的示例。第 1 行展示了输入图像以及其在默认参数设置（$\theta_c = 1.0$，$\theta_p = 0.2$，$\theta_s = 5.0$）下提取的调色板和分解的图层。第 2 行和第 3 行展示的是调整参数 θ_c 的图层分解结果，第 4 行展示的是调整参数 θ_p 的图层分解结果，第 5 行和第 6 行展示的是调整参数 θ_s 的图层分解结果。

θ_c 用于控制颜色差异对于整体距离的影响。当 θ_c 取值较小时，分解的图层无法将同一个物体中颜色差异较大的不同部分进行区分。如第 2 行所示，设置 $\theta_c = 0.3$ 时，女士的头发跟身体部分颜色差异较大但却被合并到同一个图层（第 2 行第 2 个图层），可能导致无法单独对头发进行颜色编辑。当 θ_c 取值较大时，颜色分量将占主导地位，使得颜色相似但语义不同的部分可能被合并到同一个图层中。如第 3 行所示，设置 $\theta_c = 10$ 时，使得女士被分散到多个图层（第 3 行前 3 个图层），且每个图层并没有明确的语义，这将导致完全无法做到语义级别的图像编辑。

θ_p 用于控制位置差异对于整体距离的影响。当 θ_p 取值较大时，属于同一语义的物体可能被撕裂成多个部分并分解到不同的图层。如设置 $\theta_p = 10$ 时，输入图像上方的背景将会分裂为左右两个图层（第 4 行第 4 个和第 5 个图层）。为了得到完整的、大面积的语义图层，一般将 θ_p 设置为较小的数值。

θ_s 用于控制语义特征差异对于整体距离的影响。当 θ_p 取值较小时，图层不能

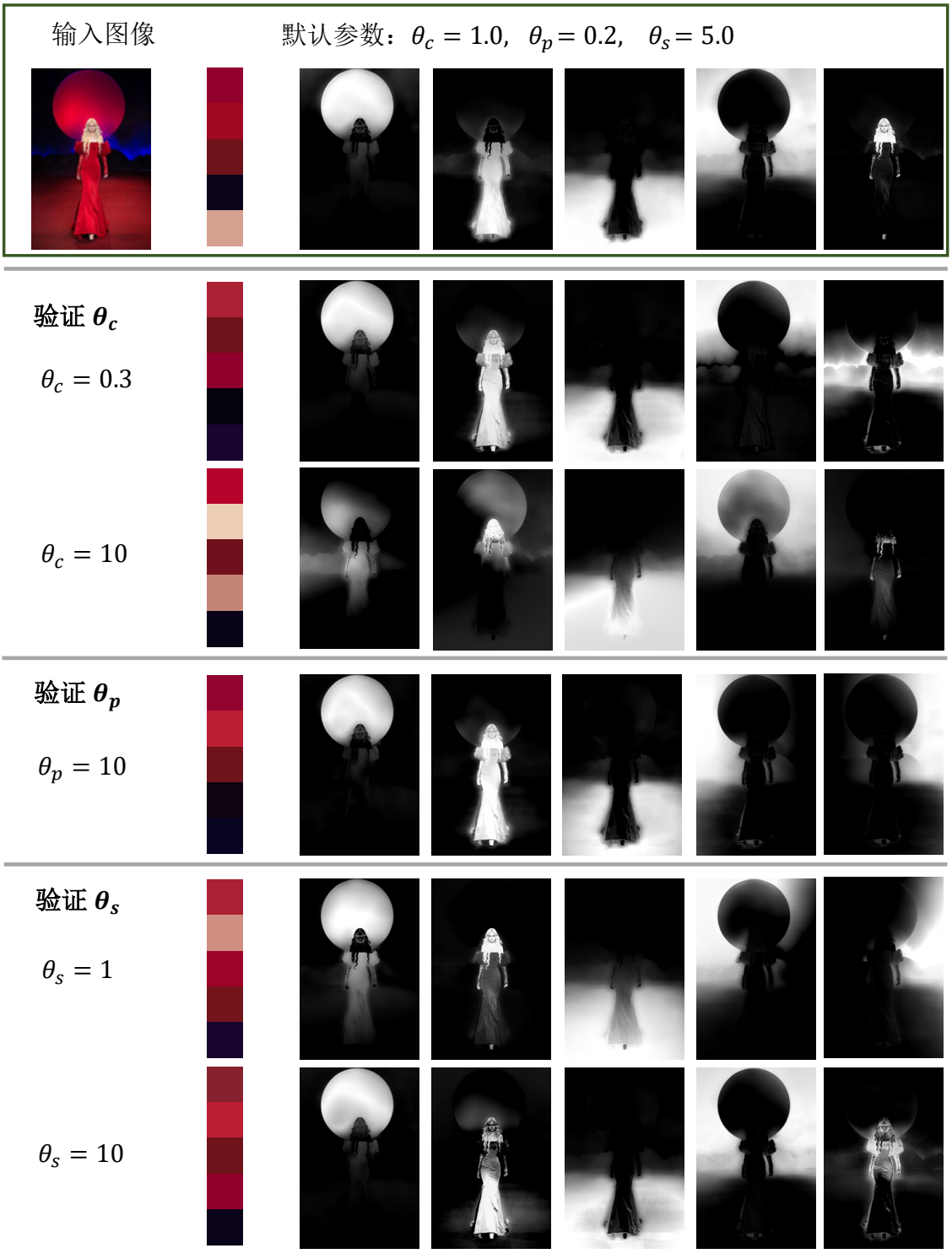
精确地捕捉到图像的语义信息。如第 5 行所示，设置 $\theta_p = 1$ 时，女士和上方的圆形出现在同一个图层（第 5 行第 1 个图层），女士和地毯也出现在同一个图层（第 5 行第 3 个图层），图层的语义属性被淡化。通常 θ_p 取值较大时，图层能更好地反映图像的语义信息。

为了更好地平衡颜色距离、坐标距离和颜色特征距离，本文最终选择 $\theta_c = 1.0$，$\theta_p = 0.2$，$\theta_s = 5.0$ 作为默认的参数设置，并在所有例子中使用这一组参数设置。实验表明，在这组参数设定下，提取的调色板和分解的图层符合用户的直觉，并基于此实现了令人满意的语义编辑效果。

4.9　本章小结

本章提出一种面向图像内容感知重着色的加性图层分解算法，针对给定的输入图像，输出一个调色板以及对应的一组语义相关的图层。进一步，基于提取的调色板和分解的图层，实现了语义级别的图像编辑。传统算法通常将图像投影到 RGB 三维空间，无论是通过聚类的方法还是通过凸包的方法，提取的调色板都只包含颜色信息，后续的图层分解亦如此。在图像重着色任务中，修改调色板颜色时，颜色相似的所有像素的颜色均会发生改变。因此，如果图像中存在多个颜色相同或相似的物体时，现有算法无法针对这些物体中的单一对象进行独立的颜色编辑。不同于现有算法，本文将图像的低级颜色特征、空间特征以及高级的语义特征相结合，并在高维空间提取图像调色板，进一步根据像素与调色板的相似度将图像分解为一组语义相关的图层，并在此基础上实现了内容感知的图像重着色。

提出的方法仍然存在一些缺点有待进一步改进。首先，本文采用 K-means 算法来提取图像的调色板，而这种算法通常需要用户提前指定类别数目。手动指定数目并不准确，且一定程度上增加了用户的使用负担。为此，我们计划通过一些自适应的聚类算法来解决这个问题，比如使用 MeanShift 算法自动地决定聚类的数目。其次，本文算法建立在语义特征检测的基础上，而逐像素的语义特征通常不够准确，这对图层分解和重着色有一些影响。后续我们期望对语义特征进行必要的预处理，使得调色板提取和图层分解更加鲁棒。



(a) 参数设置　(b) 调色板　(c) 分解的图层

图 4.12　式（ 4.12）中权重参数的验证结果

第 5 章　面向视频重着色的时变图层分解

视频编辑涉及人类生产生活的方方面面，相关技术已经在广告制作、实况转播、电影剪辑、在线会议等多个场景广泛应用。随着近年来社交网络尤其短视频平台的快速发展，人们可以随时随地创作、分享和获取视频。视频编辑技术具有广阔的应用前景。然而，大量的应用场景和海量的视频数据也对视频编辑技术的有效性和高效性提出了新的要求和挑战。本章重点关注一类视频重着色技术，具体来说，用户希望通过简单的交互实现直观、自然的视频颜色编辑。这是一个挑战性的问题，目前仍然没有很好的解决方案。我们注意到，针对图像的重着色技术目前已经取得了不错的进展，不仅交互方式简单而且重着色结果令人满意。图像重着色技术的核心在于图层分解，一旦将输入图像分解为多个加性图层，那么后续的颜色编辑就相对简单。但这种技术目前并没有应用到视频场景，针对视频的重着色编辑仍然较为繁琐。为此，本章试图将面向图像重着色的加性图层分解技术拓展到视频场景，并基于此实现简单高效的视频颜色编辑。

5.1　本章概述

本章介绍一种面向视频重着色的时变图层分解算法，将给定的视频分解为一组随时间光滑渐变的图层。与图像的图层分解方法类似，本章技术路线也主要分为两个阶段：一是视频调色板提取，二是时变图层分解。特别地，本文提出使用 4D 的倾斜多面体（Skew Polytope）来表示视频的调色板，并通过广义重心坐标插值的方法对视频进行图层分解。应用到视频编辑任务中，用户只需对视频调色板进行简单地修改就可以实现一些有趣的颜色渐变效果。总体来讲，本章算法的主要贡献体现在：1）首次将面向图像重着色的加性图层分解技术拓展到视频场景，提出的算法能够有效地将给定的视频分解为一组随时间光滑渐变的图层；2）基于此，实现了操作简单、效果自然的视频颜色编辑。

本章剩余部分将按如下方式组织：首先对本章算法涉及到的基于凸包的图像图层分解算法进行回顾；其次对 4D 几何调色板相关的概念进行介绍；然后介绍算法的输入输出并给出问题的形式化定义；接下来对算法的总体框架以及各个步骤涉及的技术细节进行全面、详细的描述；再接下来通过大量对比实验和用户实验对本文算法的有效性进行验证；最后对本文方法进行总结，并对算法存在的缺陷以及可能的改进方向进行讨论。

5.2　相关背景

本文提出的面向视频重着色的时变图层分解技术建立在 Tan 等[12,39]和 Wang 等[40]提出的面向图像的加性图层分解基础之上。为了更好地对本章算法进行描述，本节首先对面向图像重着色的基于凸包的图层分解技术进行简要回顾。

5.2.1　图层的定义

Tan 等[12,39]和 Wang 等[40]提出的加性图层分解算法需要首先从输入图像 $\mathbf{I}$ 中提取调色板 $\mathbf{V}$，然后再根据调色板将图像分解为多个跟输入图像尺寸一致的图层：$L_1, L_2, L_3 ... L_{|\mathbf{V}|}$。并且这些图层需满足：1）任意图层 L_i 中任意位置 $\mathbf{p}$ 的取值 $L_i^{\mathbf{p}} \in [0,1]$；2）各个图层在同一位置的数值之和为 1 即 $\sum_{i=1}^{|\mathbf{V}|} L_i^{\mathbf{p}} = 1$。因此，这里的图层可以简单地视为单通道的灰度图。进一步，可以通过调色板和分解的图层重建输入图像：

$$\mathbf{I}_{\mathbf{p}} = \sum_{i=1}^{|\mathbf{V}|} L_i^{\mathbf{p}} \mathbf{V}_i, \forall \mathbf{p} \in \mathbf{I} \tag{5.1}$$

实际上，上式将图像中的任意像素颜色表示为调色板颜色的凸组合。在图像重着色过程中，图层保持不变，只需将修改后的调色板 V' 再次代入上式即可计算出编辑后的图像：$\mathbf{I}'_{\mathbf{p}} = \sum_{i=1}^{|\mathbf{V}|} L_i^{\mathbf{p}} \mathbf{V}'_i$。

5.2.2　基于凸包的图像调色板

如上文所述，图层分解的首要前提是从输入图像中提取调色板。针对调色板提取，Tan 等[12,39]最早提出将几何凸包作为图像的调色板。具体而言，算法首先将输入图像投影到 RGB 空间，将图像视为 RGB 空间的点集，并计算该点集在 RGB 空间下的凸包。因为初始凸包顶点过多，无法将其作为调色板直接提供给用户使用，为此通过类似网格简化的方法简化初始凸包。最后，将简化后的凸包顶点作为调色板返回给用户。凸包具有很好的几何性质和代数性质。凸包中的任何点都可以通过广义重心坐标插值表示为凸包顶点的凸组合。这里将调色板和图像分别视为 RGB 空间的凸包及其内部的点集，那么，一旦提取到调色板，就可以通过插值的方法自然地将图像分解为多个图层（式（ 5.1））。

针对图像调色板，一般期望它具有三个优良的性质：1）为了方便编辑，调色板应当包含少量的颜色；2）调色板与其对应的图层能够高质量重建输入图像，重建误差要尽可能小；3）调色板颜色应当具有明显的代表性，能够尽可能表示图像的主要颜色分布。针对凸包表示的调色板，高质量的重建意味着凸包需要尽可能将所有像素包裹其中，因为只有凸包内部的像素才能被精确重建；优良的代表性

意味着凸包需要紧致地包裹图像像素，这样调色板颜色才能更接近图像真实的颜色分布，从而才能具有更好的代表性。显然，调色板的重建精度和优良的代表性是相互矛盾的。为了同时兼顾重建精度和调色板的紧致性（代表性），Wang 等[40]通过能量优化的方式进一步对凸包顶点的位置进行优化。具体来讲，他们定义了如下所示的能量函数：

$$\mathcal{L}(\mathrm{V},\mathrm{I}) = \lambda R(\mathrm{V},\mathrm{I}) + C(\mathrm{V},\mathrm{I}) \tag{5.2}$$

能量函数包括两项，其中，$R(\mathrm{V},\mathrm{I})$ 表示重建误差，$C(\mathrm{V},\mathrm{I})$ 表示调色板的代表性误差，λ 用于平衡重建误差和代表性误差对于整个能量函数的贡献。图 5.1 通过 2D（将 RGB 三维空间投影到 RG 二维空间）的示意图展示了凸包优化前后的形状以及能量函数各项的示意。

重建误差定义为所有像素点到凸包表面的距离的平均值：

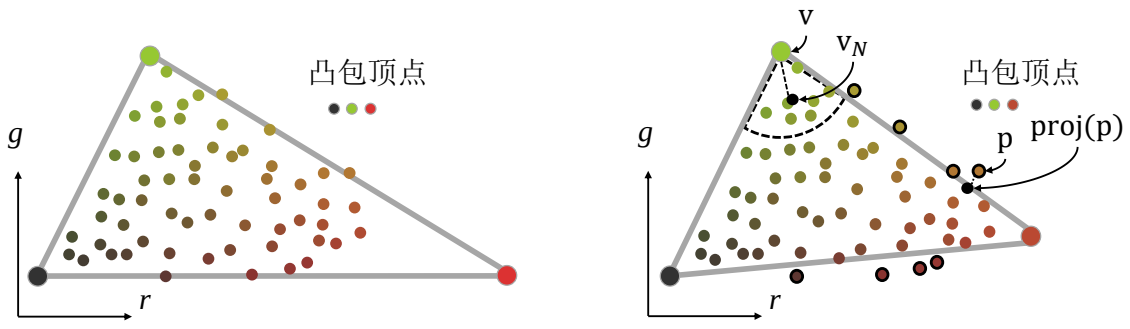
$$R(\mathrm{V},\mathrm{I}) = \frac{1}{|\mathrm{I}|}\sum_{\mathrm{p}\in\mathrm{I}} \|\mathrm{p} - \mathrm{proj}(\mathrm{p})\| \tag{5.3}$$

其中，proj(p) 表示像素点 p 到凸包 V 表面的投影距离。因为凸包内部的像素可以被凸包精确重建，为此，凸包内部的像素点有 p = proj(p)。为了降低计算的复杂性，可以随机采样适当的像素点计算重建误差。

代表性误差定义为凸包顶点到离它最近的若干像素点的中心的平均距离：

$$C(\mathrm{V},\mathrm{I}) = \frac{1}{|\mathrm{V}|}\sum_{\mathrm{v}\in\mathrm{V}} \|\mathrm{v} - \mathrm{v}_N\| \tag{5.4}$$

其中，v_N 表示距离凸包顶点 v 最近的 50 个像素点的几何中心（颜色的平均值）。



(a) 初始凸包　　(b) 优化后的凸包及能量函数示意

图 5.1　Wang 等[40]提出的调色板优化算法示意

优化后的调色板不仅提升了代表性同时保证了较高的重建精度。需要注意的是，Wang 等[40]优化后的多面体已经不再是几何上精确的凸包，但仍然可以看作凸包的近似，为了方便描述，本文依然将其称之为凸包。本章仍沿用上述能量函

数的定义来度量视频中每一帧关于调色板的能量损失。

5.3　4D 几何调色板

为了将 Tan 等[12,39]提出的面向图像的图层分解方法拓展到视频场景，我们首先需要提取输入视频对应的调色板。如 5.2小节所述，针对图像场景，Tan 等将图像看作 RGB 空间中的点集，在 RGB 空间下求解该点集的凸包，并将其作为图像调色板，最后根据调色板对图像进行图层分解。我们可以很自然地将视频当作 RGBT（三维的 RGB 颜色加上一维时间 T）空间中的点集，然后寻找一种 4D 空间中的几何体表示视频的调色板，并进一步根据调色板对视频进行图层分解。

5.3.1　调色板的选取

当把图像调色板拓展到视频场景时，我们期望调色板具有一些优良的性质：1）用户应该能够编辑颜色随时间变化的效果，例如一个延时视频可能描述了树叶从夏天到秋天的变化，或者用户可能希望添加这种效果；2）调色板应该保持时间上的连贯性和一致性，这样才能避免编辑过程中的颜色突变；3）调色板应当易于使用，这意味着调色板的颜色数量不宜过多。

关于视频的几何调色板，有如下 3 种直接的扩展方案：

一是将整个视频看作单张图像，相当于将输入视频的所有帧拼接为一张大图，并采用 Tan 等[12]的方法提取 RGB 3D 凸包作为视频调色板。但这种调色板完全忽略了视频的时变信息，因此，用户无法编辑出颜色随时间光滑渐变的效果。此外，一个共享的全局 RGB 凸包的紧致性通常很差，因为它必须同时包含所有帧中所有像素的颜色。

二是将整个视频看作 RGBT 空间中的点集，并在该 4D 空间提取整个视频所有帧共享的全局 RGBT 4D 凸包作为视频调色板。但这种凸包同样存在紧致性较差的问题，使得调色板颜色远离视频的真实颜色，从而失去了代表性。

三是计算视频每一帧的 RGB 凸包，用户通过修改每一帧对应的图像调色板实现视频颜色编辑。然而，这种方法会产生成百上千个顶点供用户编辑，操作麻烦且难以实现光滑的颜色渐变效果。

上述几种直接的扩展方法提取的调色板，要么不能刻画视频的时变信息，要么代表性较差，因此都不适合用于视频重着色任务。本章实验部分将进一步对这些简单的扩展方案进行对比。

我们的目标是为视频生成一个几何调色板，即每一帧生成一个三维多面体，它近似地包裹该帧的所有像素颜色。本文把每一帧对应的多面体称为该帧的多面体

调色板（Polyhedral Palette）。多面体调色板满足简化凸包的所有优良性质。此外，不同帧对应的多面体调色板应该满足三个条件：1）多面体调色板在相邻帧之间应该是时间上一致的，不允许发生剧烈的突变；2）允许拓扑结构发生变化，例如，某一帧的多面体调色板包含 4 个顶点，下一帧对应的多面体调色板可以包含 5 个顶点，这正好对应了颜色在时间上的出现或消失；3）整个几何调色板应该向用户暴露尽可能少的参数（即顶点和边），否则用户操作会很繁琐，用户界面应该提供一种简单的方法帮助用户同时控制所有多面体调色板。

为了实现这些目标，本文将 4D 几何调色板建立在 4D 倾斜多面体的几何结构基础之上。具体而言，本文将所有视频像素投影到 4D RGBT 空间，将视频视为该空间中的点集，并在此基础上构建尽可能包裹所有视频像素颜色的 4D 倾斜多面体。某一帧的三维多面体调色板则隐式地定义为 4D 倾斜多面体在某一时刻的切片或截面。因此，将 4D 倾斜多面体作为视频调色板，表示非常简洁、直观而紧凑，它只需要存储 4D 倾斜多面体的顶点和边。在视频重着色过程中，用户通过编辑 4D 倾斜多面体的顶点或插入新的顶点来调整视频的颜色。

接下来，本章首先对 4D 倾斜多面体的定义以及相关操作进行描述，然后介绍如何根据 4D 倾斜多面体表示的调色板对视频进行图层分解。

5.3.2　4D 倾斜多面体

定义　一个 RGBT 空间中的 4D 多面体可以定义为：$P = (V, E, F, \Gamma)$，其中，V，E，F 和 Γ 分别表示顶点集合（调色板颜色），边的集合，面的集合和超平面的集合。顶点集合中的任意顶点表示为：$v = (r, g, b, t) \in V$。为简单起见，本文将时间归一化到 $[0, 1]$，即第一帧视频和最后一帧视频分别对应于 $t = 0$ 和 $t = 1$。此外，本文也将所有像素的 RGB 值归一化到 $[0, 1]$。4D 多面体包含两种边：帧内边和帧间边。帧内边连接的两个顶点位于同一帧，即该边的两个端点具有相同的 t 值。帧间边则连接不同视频帧中的两个顶点。

截取　几何上，$n-1$ 维空间的多面体可以通过 n 维空间的超平面截取 n 维空间的多面体得到。例如我们可以使用二维平面（三维空间的超平面）截取三维空间的球得到二维的圆形。所以通过超平面 $t = t_i$ 截取 4D 多面体 P，就可以自然地获取到第 i 帧对应的 3D 多面体调色板：

$$\mathbf{V}_i = S(P, t_i) \tag{5.5}$$

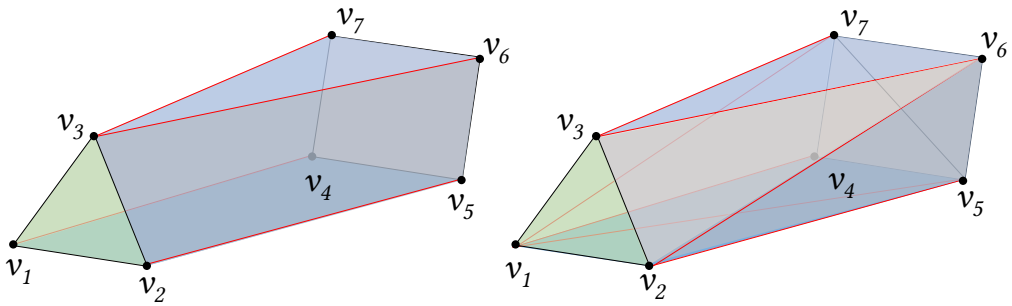
其中，$S(\cdot)$ 表示截取操作，t_i 表示第 i 帧的时间，$\mathbf{V}_i$ 表示截取到的多面体调色板。

4D 倾斜多面体连同切片操作使得我们能够获得随时间光滑变化的 3D 多面体调色板。基于此，才能进一步将视频分解为光滑变化的时变图层，并最终应用到

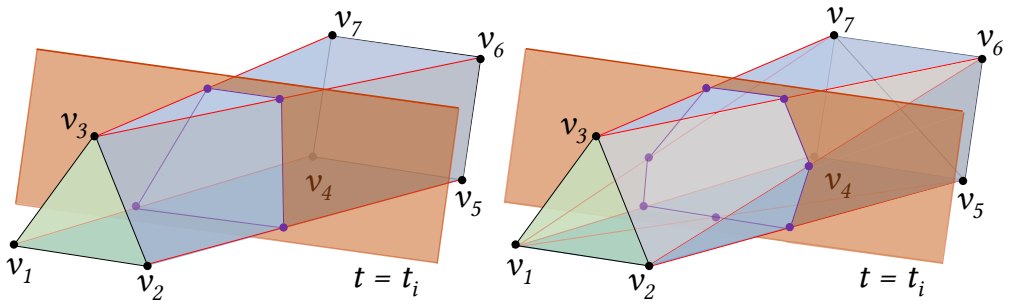
视频重着色任务中。

倾斜多面体与普通多面体　数学上，倾斜多面体是一种广义的多面体，特别地，它允许非平面的面[140]。相反，普通多面体的面则必须是平面。本文使用 4D 倾斜多面体表示视频的调色板主要基于两个原因。首先，倾斜多面体本身包含较少的边，因此它的表示方法更加简单，也更加易于控制。其次，对 4D 倾斜多面体截取生成的 3D 多面体包含较少的顶点和边，拓扑更加简单。相反，对普通 4D 多面体截取生成的 3D 多面体通常包含较多冗余的顶点和边，因此拓扑较为复杂。

因为 4D 多面体难以可视化，图 5.2给出了一个低维的示例（这里用三维的多面体表示四维多面体）。图 5.2(a) 展示了倾斜多面体跟普通多面体的对比，图 5.2(b) 展示了倾斜多面体跟普通多面体的切面对比。给出的示例中，普通多面体包含 15 条边，而倾斜多面体只包含 11 条边。从普通多面体截取的多边形包含 7 条边，而从倾斜多面体截取的多边形只包含 4 条边。相对于普通多面体，倾斜多面体本身及其截面都具有更简单的拓扑结构。



(a) 倾斜多面体 (左) 和普通多面体（右）



(b) 倾斜多面体的截面 (左) 和普通多面体的截面（右）

图 5.2　倾斜多面体和普通多面体的对比

5.3.3　时变图层分解

针对给定的输入视频及其 4D 几何调色板，本节首先通过两阶段插值的方法将视频像素表示为 4D 几何调色板颜色的凸组合，然后再从插值结果中导出随时间光滑渐变的图层。分解的时变图层将直接影响后续的视频颜色编辑。

视频像素关于 4D 几何调色板插值

在任意时刻 t，我们都可以通过截取操作（式（ 5.5））得到该帧对应的多面体调色板 $\mathbf{V}_t$。进一步，可以使用广义重心坐标[13,141-143]插值的方法来计算第 t 帧像素 $\mathbf{p}=(r,g,b,t)$ 相对于多面体调色板 $\mathbf{V}_t$ 的权重 L_p。本文使用均值坐标[144]（Mean Value Coordinates，MVC）来计算权重，即将视频中每一帧的像素颜色表示为对应的多面体调色板颜色的凸组合：

$$\mathbf{p}=L_{\mathbf{p}}\mathbf{V}_t \tag{5.6}$$

其中，$L_{\mathbf{p}}=[L_1^{\mathbf{p}},L_2^{\mathbf{p}},\cdots,L_{|\mathbf{V}_t|}^{\mathbf{p}}]$ 是一个行向量，表示像素点 $\mathbf{p}$ 相对于该帧多面体调色板各颜色的权重。$\mathbf{V}_t=[\mathbf{V}_t^1,\mathbf{V}_t^2,\cdots,\mathbf{V}_t^{|\mathbf{V}_t|}]^{\mathbf{T}}$ 是一个 $|\mathbf{V}_t|$ 行 4 列的矩阵，其中每一行表示该多面体调色板中的一个顶点。需要注意的是，这里的像素 $\mathbf{p}$ 和多面体调色板的顶点集合 $\mathbf{V}_t$ 具有相同的 t 值，因此计算过程中省略 t 值，并直接用三维的 MVC 计算插值权重。

因为多面体调色板是从 4D 几何调色板中截取得到，所以多面体调色板的顶点 $\mathbf{V}_t$ 必然位于 4D 几何调色板的帧间边上。因此，多面体调色板的顶点可以进一步表示为这些边（4D 几何调色板的帧间边）的端点的线性组合：

$$\mathbf{V}_t=L_t\mathbf{V} \tag{5.7}$$

其中，L_t 是一个 $|\mathbf{V}_t|$ 行 $|\mathbf{V}|$ 列的矩阵，表示多面体调色板顶点 $\mathbf{V}_t$ 关于 4D 几何调色板 $\mathbf{V}$ 的插值权重，$\mathbf{V}$ 是一个 $|\mathbf{V}|$ 行 4 列的矩阵，它的每一行表示 4D 几何调色板的一个顶点。

由于这两个步骤都是线性的，它们可以通过矩阵乘法结合起来。这样，视频中的任意像素点 $\mathbf{p}=(r,g,b,t)$ 就可以直接通过 4D 几何调色板顶点的线性组合表示：

$$\mathbf{p}=L_{RGBT}\mathbf{V} \tag{5.8}$$

其中，$L_{RGBT}=L_{\mathbf{p}}L_t$。接下来，我们将描述如何从上述插值权重中导出视频对应的时变图层。

从插值结果导出时变图层

针对输入视频中的某一帧 $\mathbf{I}_t$，假设视频的分辨率为 $\omega\times h$，那么该帧所有像素的颜色按上述插值方式可以表示为：

$$\mathbf{I}_t=L_{RGBT}^t\mathbf{V_t} \tag{5.9}$$

其中，$\mathbf{I}_t$ 是一个 ωh 行 4 列的矩阵，每一行表示该帧中一个像素的颜色。L_{RGBT}^t 是权重矩阵，行数为 ωh，列数为该帧对应的多面体调色板的颜色数目 $|\mathbf{V_t}|$，每一

行表示该帧中一个像素点关于所有 4D 几何调色板顶点的插值权重，且每一行的权重之和为 1。可以将 L^t_{RGBT} 当作一个二维张量，进一步将它变形为维度分别为 $\omega, h, |\mathbf{V_t}|$ 的三维张量。那么这个三维张量就可以认为是一个宽度为 ω，高度为 h，通道数为 $|\mathbf{V_t}|$ 的图像，它的每一个通道就代表了一个图层。为了方便，本文将视频中第 t 帧的第 i 个图层 L^i_t 表示为：

$$L^i_t = L^t_{RGBT}.\text{Reshape}(\omega, h, |\mathbf{V_t}|).\text{Channel}(i) \tag{5.10}$$

上式右侧的表达式按从左至右的顺序执行。其中，$\text{Reshape}(\omega, h, |\mathbf{V_t}|)$ 函数将二维矩阵 L^t_{RGBT} 转换为维度分别为 $\omega, h, |\mathbf{V_t}|$ 的三维张量，并将其视为一张多通道的图像。$\text{Channel}(i)$ 函数表示获取该图像的第 i 个通道。总体来说，视频中的每一帧可以分解为多个时变图层，且图层数量跟该帧对应的多面体调色板的颜色数目一致。不同帧对应的多面体调色板的颜色数目不尽相同，因此不同帧的图层数目可能并不一致。

5.4　问题的形式化定义

5.4.1　输入输出

我们的目标是将输入的视频分解为一组随时间变化的图层，并借此实现高效的视频颜色编辑。跟面向图像重着色的图层分解方法类似，提出的算法首先从视频中提取 4D 几何调色板，然后再根据调色板对视频进行图层分解。简而言之，算法的输入和输出分别为：

- 输入：一个包含 N 帧的视频 $\mathbf{I}$。
- 输出：一个 4D 几何调色板 $P = (V, E, F, \Gamma)$ 和一组随时间光滑渐变的图层。

5.4.2　能量函数

为了将视频分解为一组时变图层，首要任务是要从输入视频中提取 4D 几何调色板，然后再通过 5.3.3小节所示的方法进行图层分解。为了得到光滑渐变的图层，以及满意的视频重着色效果，首先需要提取得到高质量的 4D 几何调色板。仿照 Wang 等[40]提取调色板的工作，本文仍使用式（ 5.3），式（ 5.4）和式（ 5.2）来定义视频中每一帧的重建损失、紧致性损失和整体损失（即重建和紧致性损失的组合）。因此，4D 几何调色板关于输入视频的整体能量函数定义为：

$$\mathcal{L}_v(P, \mathrm{I}) = \frac{1}{N}\sum_{i=1}^{N} \theta_i \mathcal{L}\left(S\left(P, t_i\right), \mathrm{I}_i\right) \tag{5.11}$$

其中，i 是视频帧的索引，$S(P,t_i)$（式（ 5.5））表示 P 在时间 t_i（或第 i 帧）的截面即三维多面体调色板，$\mathbf{I}_i$ 和 θ_i 分别表示输入视频的第 i 个帧及其对应的权重。

本节介绍了算法的输入输出并给出了 4D 几何调色板关于输入视频的能量函数。本章后续部分将详细阐述如何在上述能量函数的指导下从给定视频中提取高质量的 4D 几何调色板。这是对输入视频进行时变图层分解的重要前提，也是执行视频颜色编辑的基础。

5.5　算法总体框架

从视频中生成 4D 多面体最直接的方法是在 RGBT 空间中求解视频像素对应的简化 4D 凸包。然而，这样产生的 4D 凸包往往沿时间维度包围了大量的空白空间，不符合本文对调色板紧致性的要求。

本文提出了一种渐进的方法从视频中提取 4D 倾斜多面体，并将其作为视频的 4D 几何调色板。提出的方法主要包括以下 4 步：

1. **构建初始 4D 倾斜多面体** 将相邻视频帧的 3D 简化凸包粘连在一起，生成一个初始的 4D 倾斜多面体。初始的 4D 倾斜多面体拓扑非常复杂，包含大量的顶点和边，它是后续拓扑优化和网格简化的基础。这部分内容参见 5.6 小节。
2. **帧块合并** 在上述初始化阶段，每个视频帧都独立地生成 3D 简化凸包。因此，相邻帧的凸包很可能具有不同的拓扑结构。然而帧间拓扑结构的频繁变化往往是不必要的，并且会使得后续的简化操作非常困难。因此引入帧块合并算法用于减少帧间拓扑变化。这部分内容参见 5.7 小节。
3. **顶点删除** 为了进一步降低 4D 倾斜多面体的复杂性，本文通过迭代的方法删除冗余顶点来简化 4D 倾斜多面体，直到整体损失达到设定的阈值。这部分内容参见 5.8 小节。
4. **顶点优化** 在上述简化步骤之后，本文继续进行顶点优化，以进一步减少整体损失并增加时间上的一致性。这部分内容参见 5.9 小节。

完整的调色板提取算法流程如图 5.3 所示。为了更好地理解算法，图中使用 3D 的形状表示 4D 的倾斜多面体。

5.6　构建初始 4D 倾斜多面体

初始的 4D 倾斜多面体构建分为三步：1）为每一帧生成 3D 多面体调色板；2）利用相邻两帧的 3D 多面体生成横跨两帧的 4D 倾斜多面体；3）将步骤 2 中生成的所有 4D 倾斜多面体连接起来构成初始的覆盖所有帧的 4D 倾斜多面体。

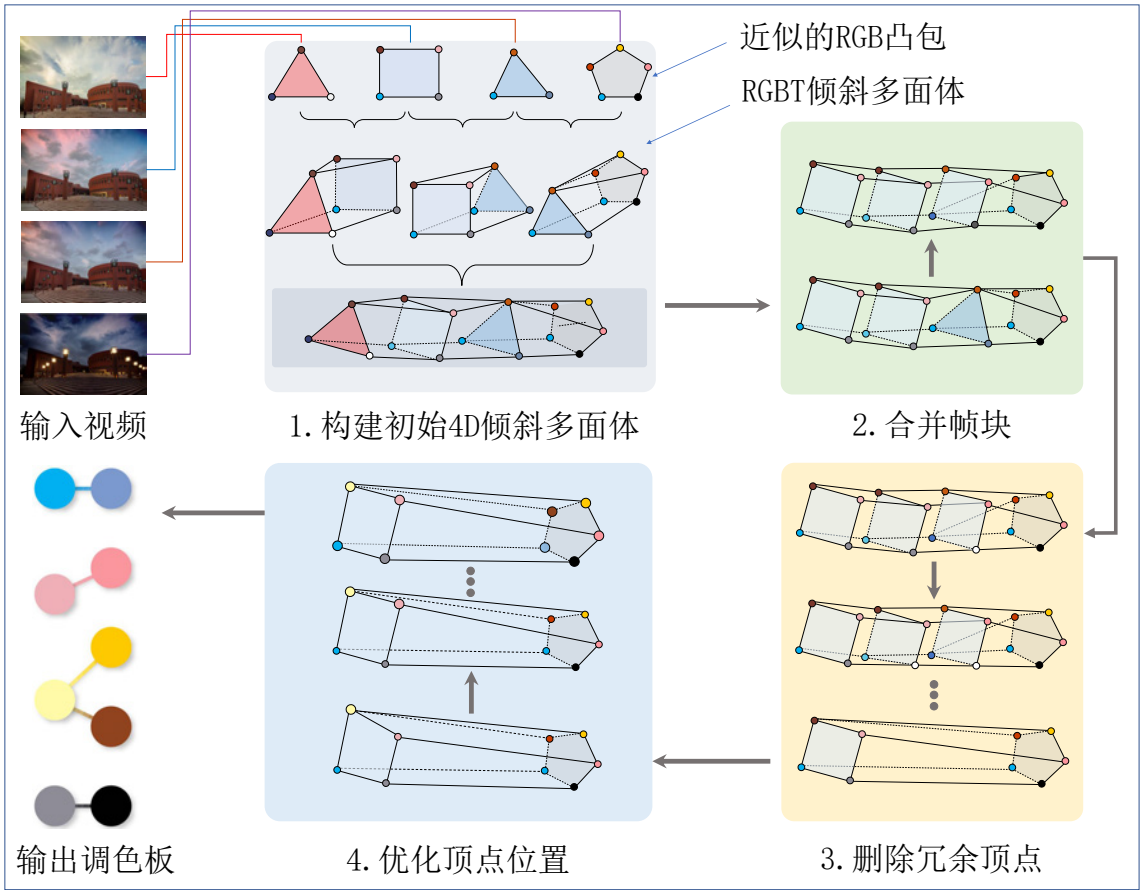


图 5.3　调色板提取的算法流程

生成逐帧多面体调色板

对于每一帧视频，本文首先使用 Tan 等[12]提出的方法来提取一个简化的 3D RGB 凸包，然后使用 Wang 等[40]提出的方法优化顶点位置以减少整体损失（式（5.2））。最终，为每一帧视频生成了一个近似的、紧致性较好的 RGB 凸包，作为该帧的多面体调色板。

构造横跨两帧的 4D 倾斜多面体

对于每一对相邻的视频帧，我们需要连接它们对应的多面体调色板的顶点，以形成一个横跨两帧的 4D 倾斜多面体。假设相邻两帧对应的多面体调色板的顶点集分别表示为 $\mathbf{V}=\{v_i\}$ 和 $\mathbf{U}=\{u_j\}$，那么我们的目标是生成一组帧间边 $\mathbf{E}=e_k$，其中，每条边 e_k 的一个端点属于集合 $\mathbf{V}$，另一个端点属于集合 $\mathbf{U}$。

本文引入了 4 条规则以确保生成的横跨两帧的 4D 倾斜多面体具有正确的拓扑和简明的对应关系。前两个规则确保集合 $\mathbf{V}$ 中的任意顶点都能在集合 $\mathbf{U}$ 中找到对应顶点。

- **帧间度数规则 1**　我们将一个顶点的帧间度数定义为它所连接的帧间边的数

量，并要求 **V** 和 **U** 中的每个顶点的帧间度数大于等于 1，不允许存在帧间度数为 0 的顶点。

- **帧间度数规则 2** 对于每条帧间边，最多允许其中一个顶点的帧间度数 $\geqslant 2$。

后两个规则确保某一帧中多面体调色板的任意边可以对应到另一帧中的一条边（或一个顶点），使得该 4D 倾斜多面体具有简单的拓扑连接。

- **拓扑连接规则 1** 对于每个有多条帧间边的顶点 v，假设它连接到两个不同的顶点 u_1 和 u_2，那么 u_1 和 u_2 必须已经通过一条帧内边连接。反之，对于同时连接到 v_1 和 v_2 的 u 也是如此。
- **拓扑连接规则 2** 对于每条帧内边 (v_1, v_2)，假设 v_1 连接到 u_1，v_2 连接到 u_2，本文要求 u_1 和 u_2 必须已经通过帧内边连接，或 $u_1 = u_2$（同一顶点）。

图 5.4 给出了这些规则的示意。图 5.4 (a) 中 u_4 是一个孤立的顶点，帧间度数为 0，这违反了帧间度数规则 1；图 5.4 (b) 中帧间边 (v_2, u_3) 和 (v_3, u_3) 的两个端点的度数都是 2，这违反了帧间度数规则 2；图 5.4 (c) 中帧内边 (v_1, v_2) 在另一帧中没有对应的顶点和边，这违反了拓扑连接规则 2；图 5.4 (d) 所示的多面体满足所有上述规则，在这个例子中，v_3 是一个分裂顶点。

上述规则有助于使多面体调色板之间的拓扑对应或变化更有意义。总体上，相邻帧的多面体调色板的顶点只允许存在三种对应关系：

1）一对一：**V** 中的一个顶点 v 只与 **U** 中的一个顶点 u 连接（反之亦然）。

2）多对一：**V** 中的多个顶点连接到 **U** 中的一个顶点 u；将 u 称为合并顶点。

3）一对多：**V** 中的一个顶点 v 连接到 **U** 中的多个顶点；将 v 称为分裂顶点。

需要注意的是，多对多的顶点对应关系是不允许的，否则将导致生成的 4D 倾斜多面体的拓扑连接异常复杂，不利于后续的简化。

除了这些要求外，本文还希望 4D 倾斜多面体在时间维度上是平滑的。例如，如果第 1 帧和第 2 帧都有 1 个红色和 1 个绿色的顶点，那么连接红与红、绿与绿显然比连接红与绿、绿与红更好。从形式上看，本文需要寻求一组帧间边 **E**，在上述条件下使得 $\sum_{(u,v)\in E} \left\| u_{RGB} - v_{RGB} \right\|$ 取得最小值。为此，本文首先在 $\mathbf{V} \bigcup \mathbf{U}$ 的顶点集合上计算出一个 4D 凸包，然后在该 4D 凸包的所有帧间边的集合上搜索满足上述条件的 4D 倾斜多面体。

构造整体的初始 4D 倾斜多面体

最后我们将所有通过相邻帧对创建的 4D 倾斜多面体粘在一起，形成初始的覆盖整个视频的 4D 倾斜多面体。需要注意的是，初始的 4D 倾斜多面体包含较多的顶点和边，并且具有复杂的拓扑结构。为了得到拓扑简单的视频调色板，针对初始 4D 倾斜多面体的简化操作必不可少，下文将对拓扑简化操作进行详细的描述。

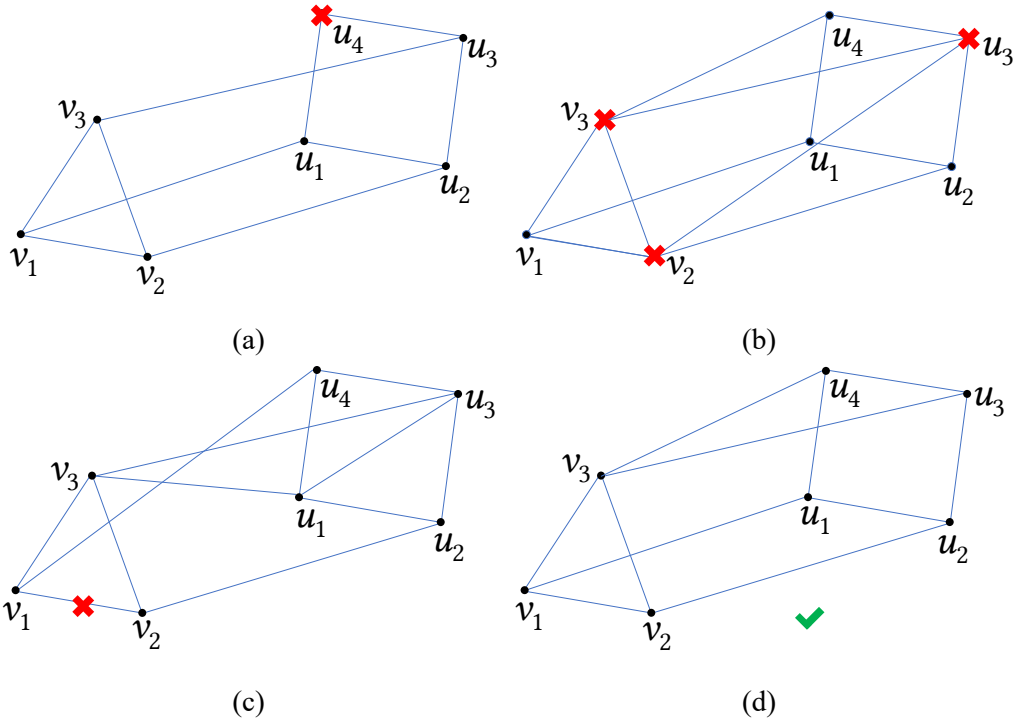


图 5.4　帧间度数规则和拓扑连接规则

5.7　帧块合并

初始的 4D 倾斜多面体具有复杂的拓扑结构，通常包含数百个顶点和边。它需要被进一步简化才能作为 4D 几何调色板供用户使用。如果把这个 4D 倾斜多面体看作网格的话，似乎可以通过顶点删除操作来对网格进行简化。然而，直接简化最初的 4D 倾斜多面体会导致截面即多面体调色板出现频繁的拓扑变化。

给定相邻的两帧，如果满足：1）它们的多面体调色板具有相同的拓扑结构，即相同的顶点数量和相同的帧内连接；2）它们的任意帧间边连的两个顶点都是一一对应的。那么就称这两帧在拓扑上一致。初始情况下，许多帧对的拓扑是不一致的，因为每一帧的多面体调色板都是独立生成的，在这个过程中并没有考虑时间上的一致性。为了解决这个问题，本文提出了一种合并算法来创建连续的、拓扑上一致的帧块。

5.7.1　帧块及其合并操作

本文将帧块定义为一组连续的帧，其中每一对相邻的帧对应的多面体调色板具有完全一致的拓扑，所有的帧间边连接的顶点都是一一对应的。一个帧块可以包含单帧（当它与相邻的两个帧拓扑不一致时）或多个帧。因此，整个视频（或 4D 倾斜多面体）可以被看作是一连串的帧块 $B = \{b_i\}$ 集合，为了方便描述，本文将其称之为帧块配置。

两个相邻的帧块可以被合并。如果将一个帧块 b_2 合并到相邻的帧块 b_1 中，那么它将产生一个包含 b_1 和 b_2 所有帧的新帧块 b。同时，原本属于 b_2 的帧将丢弃它们的多面体调色板，并采用与 b_1 中的帧一致的多面体调色板。

不失一般性，假设 b_1 在 b_2 之前，且我们期望将 b_2 合并到 b_1。假设 b_1 的最后一帧为 s，对应的多面体调色板为 $\mathbf{V}_s$，帧块 b_2 包含的帧分别为 $s+1,\cdots,s+n$。根据要求，我们需要为 b_2 中的所有帧计算新的多面体调色板 $\mathbf{V}_{s+1},\cdots,\mathbf{V}_{s+n}$。如图 5.5 所示，我们从 $s+1$ 帧开始按从前到后的顺序迭代地计算 b_2 中所有帧的多面板调色板。这一计算过程可以形式化地表示为如下的形式：

$$\mathbf{V}_k = \text{VertexRefine}(\mathbf{V}_{k-1}, \mathbf{I}_k), \quad k = s+1,\cdots,s+n. \tag{5.12}$$

为了求当前帧的多面体调色板 $\mathbf{V}_k$，本文首先将上一帧的多面体调色板 $\mathbf{V}_{k-1}$ 拷贝到第 k 帧，然后通过 Wang 等[40]提出的方法（VertexRefine 函数）迭代地优化该多面体调色板的顶点，使得它能在损失函数的约束下更好地匹配当前视频帧 $\mathbf{I}_k$。b_2 中的多面体调色板的拓扑结构直接取自 b_1。合并后的块 b 中所有帧对应的多面体具有一致的拓扑，且相邻帧的顶点一一对应。需要注意的是，帧块的合并是不可交换的，尽管将 b_2 合并到 b_1 与将 b_1 合并到 b_2 都会得到一致的拓扑，但两种合并方式得到的拓扑结构完全不同。

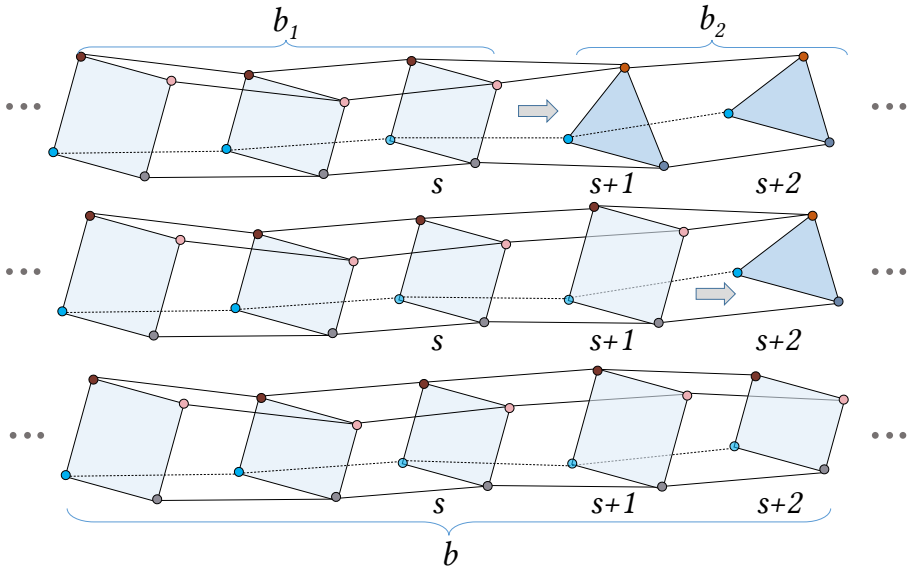


图 5.5　相邻帧块的合并示意

5.7.2　帧块的度量

如上文所述，整个视频可以视为一个帧块序列。为了得到拓扑简单的 4D 几何调色板，本文希望减少帧块序列中帧块的数量同时保持 4D 倾斜多面体的重建精度和良好的紧致性。因此本文使用式（ 5.11）表示的总体损失函数来衡量一个帧块

序列的质量，其中任意帧的权重 θ_i 定义为：

$$\theta_i = \left(1 + \alpha \frac{n_j}{N}\right)\left(1 + \beta m_j\right) \tag{5.13}$$

其中，第 i 帧属于帧块 b_j，n_j 表示 b_j 包含的帧的总数，N 表示整个视频包含的帧的总数，m_j 表示 b_j 中每帧对应的多面体调色板的顶点数目。上式包含两项，其中，第一项 $\left(1 + \alpha n_j / N\right)$ 用于对包含帧较多的大帧块进行惩罚，其目的在于鼓励较小的帧块首先被合并。第二项 $(1 + \beta m_j)$ 用于惩罚那些包含大量顶点的多面体调色板，其目的在于鼓励生成数目较少且具有明显区分度的多面体调色板。α 和 β 用于控制这两项的贡献强度，根据经验，本文设定 $\alpha = 10$，$\beta = 1$。

5.7.3　帧块合并算法

本文通过迭代地合并相邻帧块的方式对初始的 4D 倾斜多面体进行拓扑优化，直到式（ 5.11）所示的损失超过设定的阈值。具体而言，给出的帧块合并算法包含以下四步：

1. 帧块合并算法从初始帧块配置 $\boldsymbol{B} = \{b_i\}$ 开始，根据式（ 5.11）和式（ 5.13）计算初始帧块配置的总体能量损失 $\mathcal{L}_b^0$。
2. 对于帧块配置 $\boldsymbol{B}$ 中的每一对相邻的帧块 b_1 和 b_2，本文计算两种可能的帧块合并操作（将 b_1 合并到 b_2，以及将 b_2 合并到 b_1）对应的损失。
3. 在步骤 2 计算的所有可能的合并以及它们的损失中，本文贪心地选择具有最小损失的帧块合并方式，并相应地对帧块配置 $\boldsymbol{B}$ 进行更新。
4. 重复步骤 2 和 3，直到当前帧块配置的损失 $\mathcal{L}_b$ 超过某个设定的阈值，即 $\mathcal{L}_b > \eta_1 \mathcal{L}_b^0$ 或当前只剩一个唯一的帧块时，帧块合并操作结束。η_1 是一个预定义的阈值，实验中，本文将其设置为 $\eta_1 = 3.5$。

经过帧块合并操作之后，帧块的数量大大减少，更新后的 4D 倾斜多面体包含更简单的拓扑连接。当然，简化的 4D 倾斜多面体仍然包含一些拓扑变化，如某些顶点的帧间度数大于 2，表现为 4D 倾斜多面体上顶点的拆分和帧间边的交汇。这是允许的，因为这正好对应了视频中突然出现或消失的颜色。

5.8　顶点删除

帧块合并之后，4D 倾斜多面体的拓扑结构更加简单，但仍包含较多的顶点，还不能将其作为视频调色板供用户使用。同时我们也注意到经过帧块合并后的 4D 倾斜多面体包含大量冗余的、不必要的顶点。比如 4D 倾斜多面体上首尾相接的 k 个帧间边 $\{e_1, e_2, \cdots, e_k\}$ 近似或精确位于某一条直线上，那么这些帧间边的端点大

部分都是冗余且不必要的，因为大部分顶点都可以通过线性插值得到，所以只需要存储 e_1 中时间值较小的端点和 e_k 中时间值较大的端点即可。为此，本节通过顶点删除操作来进一步降低 4D 倾斜多面体的复杂性。

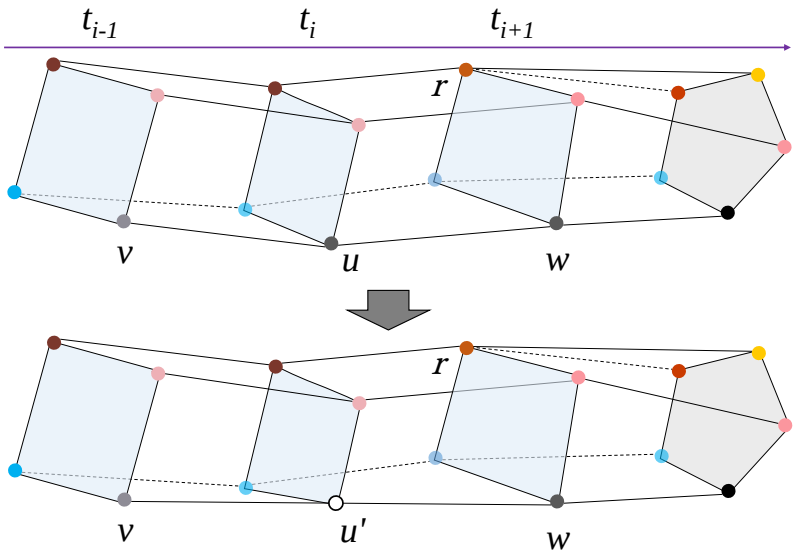


图 5.6　4D 倾斜多面体冗余顶点删除示意

删除单个顶点

在删除 4D 倾斜多面体的某个顶点后，本文引入一个虚拟的 ghost 顶点来暂时代替移除的顶点，并且本文要求该 ghost 顶点必须位于非 ghost 顶点连接的帧间边上。如图 5.6 所示，当顶点 u 被删除后，顶点 v 和顶点 w 被连接起来，形成一条新的帧间边 vw，产生的 ghost 顶点 u' 被强制要求位于线段 vw 上。具体来说，它是超平面 $t = t_i$ 和边 vw 的交点。

本文顶点删除方案有两个限制。首先，分裂和合并的顶点不能被删除，因为删除这些顶点会改变帧块合并产生的拓扑结构。如图 5.6中的顶点 r 就不能被移除，因为它是一个分裂顶点。其次，第一帧和最后一帧的顶点不能被删除，因为首尾帧的顶点不能通过其它顶点插值得到，同时删除首尾帧的顶点将可能改变视频时长，所以这是不允许的。

迭代删除算法

与帧块合并过程类似，这里仍然使用式（ 5.11）所示的总体损失函数来衡量简化的 4D 倾斜多面体的质量，并将所有帧的权重设为 $\theta_i = 1$。本文迭代地删除 4D 倾斜多面体的顶点，直到整体损失达到给定的阈值。具体而言，给出的算法包含以下 4 步：

1. 迭代删除算法从帧块合并后的 4D 倾斜多面体开始，首先根据式（ 5.11）计

算该 4D 倾斜多面体的初始损失 $\mathcal{L}_v^0$。

2. 对于每个可能删除的顶点，计算删除该顶点后的总体损失。为了减少搜索时间，本文只考虑那些最接近其潜在 ghost 顶点位置的 10 个顶点作为删除的候选对象。某个顶点 u 跟潜在的 ghost 顶点 u' 接近意味着 u 连接的两条帧间边近似位于一条直线上，u' 替代 u 后只会带来微小的能量损失。
3. 针对所有的候选顶点，对它们删除后的损失进行排序，本文贪心地选择删除后具有最小损失的顶点，并将其从 4D 倾斜多面体中删除。
4. 重复步骤 2 和 3，直到当前的总体损失大于初始总体损失的某个倍数（$\mathcal{L}_v > \eta_2 \mathcal{L}_v^0$）或者已经没有其他顶点可以被删除时，算法停止。

经过顶点删除操作之后，4D 倾斜多面体中大量的冗余顶点被删除，最终只保留了那些重要的、必要的顶点，因此，4D 倾斜多面体的拓扑更加简明，表达更加紧凑。

5.9　顶点优化

在简化 4D 倾斜多面体后，本节继续通过顶点优化来进一步减少 4D 倾斜多面体的整体损失。此外，本文还期望 4D 倾斜多面体在时间上是一致的，以便多面体调色板沿着时间线平滑地变化。为了度量 4D 倾斜多面体 P 的平滑程度，本节定义了一个额外的平滑项：

$$K(P) = \frac{1}{|E|} \sum_{(u,v)\in E} \left| \frac{u_{RGB} - v_{RGB}}{u_t - v_t} \right| \tag{5.14}$$

其中，u_{RGB} 和 v_{RGB} 分别表示顶点 u 和 v 的颜色分量，u_t 和 v_t 分别表示顶点 u 和 v 的时间值。我们枚举 4D 倾斜多面体帧间边集 E 中的每条边 (u, v)，并计算每条边上颜色相对于时间的变化率之和，然后将其平均值作为该 4D 倾斜多面体的平滑度。因此，顶点优化的损失函数定义为总视频损失（式（ 5.11））加上该平滑项：

$$\mathcal{L}_v'(P, I) = \mathcal{L}_v(P, \mathbf{I}) + \gamma K(P) \tag{5.15}$$

其中，权重 γ 被设为 0.01，并将视频损失中的帧权重 θ_i 被设为 1。

本文采用迭代的方式优化每个顶点的位置，以降低式（ 5.15）所示的能量损失。具体来说，每次迭代仅选择一个顶点进行局部调整，并固定所有其他顶点的位置。本文使用 NLopt 库[100]在 4D RGBT 空间中局部调整顶点 v_{RGBT} 的位置，并将搜索空间限定在 $[v_R - 0.1, v_R + 0.1] \times [v_G - 0.1, v_G + 0.1] \times [v_B - 0.1, v_B + 0.1] \times [v_b^s, v_b^e]$。其中，$v_R$，$v_G$ 和 v_B 分别表示 v 点的三个颜色分量，v_b^s 和 v_b^e 是 v 所在帧块的开始和结束时间。为了保持帧块结构，第一帧和最后一帧中的顶点以及分割和合并顶点将不允许修改 t 值。对于上述迭代优化过程，通常需要循环遍历所有顶点数次，

直到收敛（通常需要 3 次循环就能达到满意的结果）。

到此为止，针对给定的视频，整个 4D 几何调色板的提取算法的所有步骤已经描述完毕。获取到 4D 几何调色板后，通过 5.3.3 小节介绍的方法就可以将输入视频分解为一组时变的图层。最后，根据视频的图层分解结果，用户通过修改 4D 几何调色板来调整或修改视频的颜色。

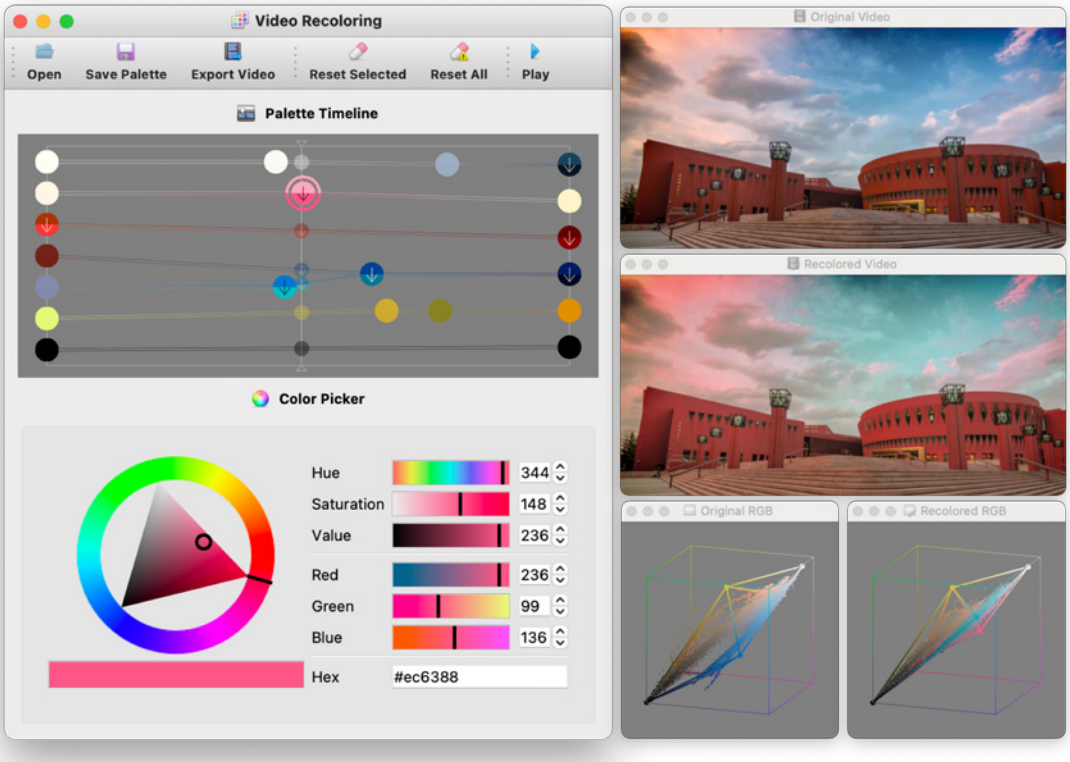


图 5.7　视频重着色编辑用户界面

5.10　实验

这一部分通过实验对本章算法的有效性进行验证，主要包括参数评估、方法对比、图层分解结果、视频编辑结果、用户实验五部分。本章实验环境为：

- 操作系统：Windows 10；
- 处理器：Intel i9-9900K 3.6 GHz CPU、16 核；
- 内存：16 GB RAM；
- 编程环境：Microsoft Visual Studio 2019。

本章算法通过 C++ 语言实现，用到的第三方库有 NLopt[100] 和 OpenCV[105]，并使用 Tan 等[12]开源的 Python 代码求解视频帧的多面体调色板，以及 Wang 等[40]开源的代码对提取的多面体调色板进行顶点优化。

本文使用 Qt 实现了一个视频颜色编辑的图形用户界面如图 5.7 所示。该图形

用户界面以视频编辑软件中常见的时间线视图呈现 4D 几何调色板，并有一个颜色选择器用于修改调色板的颜色。为了方便比较，我们将当前时刻的视频帧重着色前后的图像进行对照显示。同时，用户也可以观察当前帧编辑前后的多面体调色板在 RGB 空间中的形状。

5.10.1　参数评估

本小节首先评估帧块合并和顶点删除算法中使用的参数对最终 4D 几何调色板的影响，验证的参数包括：1）在帧块合并（ 5.7 小节）中使用的三个参数：α，η_1，β；2）在顶点删除（ 5.8 小节）中使用的参数 η_2。然后对顶点优化（ 5.9 小节）步骤的必要性进行验证。

参数 α 和 η_1

在帧块合并中，参数 α（式（ 5.13））用于控制损失对帧块大小的依赖程度，而 η_1 控制迭代过程何时停止（即当帧块配置的能量损失 $\mathcal{L}_b > \eta_1 \mathcal{L}_b^0$）。这两个参数共同控制了帧块合并后帧块的数目。通常，较大的 α 和较小的 η_1 会导致更多的帧块，从而使 4D 倾斜多面体的拓扑结构更加复杂，而较小的 α 和较大的 η_1 则会使得合并后的 4D 倾斜多面体包含较少的帧块和更简单的拓扑结构。

本文用 α 和 η_1 的不同组合在几个例子中测试了帧块合并算法，并记录了每一组参数配置下帧块合并后的帧块数量。统计结果如表 5.1 所示。实验结果表明，$\alpha = 10$ 和 $\eta_1 = 3.5$ 对所有的例子都能产生适当数目的帧块，从而在 4D 几何调色板的拓扑结构的复杂性和控制的便利性之间取得良好的平衡。

表 5.1　α 和 η_1 的不同取值对帧块合并的影响

示例名称	初始帧块数目	η_1 ($\alpha = 5$)			η_1 ($\alpha = \mathbf{10}$)			η_1 ($\alpha = 15$)		
		1.5	3.5	6.0	1.5	3.5	6.0	1.5	3.5	6.0
森林 (图 5.13 (a))	61	3	1	1	3	2	1	5	2	1
街市 (图 5.13 (b))	75	8	3	1	12	3	2	13	4	2
云彩 (图 5.14 (a))	77	1	1	1	2	1	1	5	1	1
夜晚 (图 5.14 (c))	74	1	1	1	6	3	1	6	2	1
水域 (图 5.14 (d))	98	5	2	1	9	3	2	12	5	2

参数 β

在帧块合并中，较大的参数值 β（式（ 5.13））鼓励生成顶点数量较少的多面体调色板。当 $\beta = 0$，合并两个含有不同顶点数目的多面体调色板的块时，算法将

倾向于保留顶点数较多的多面体调色板，而丢弃顶点数较少的调色板，因为具有更多顶点的多面体调色板有更多的自由度可以优化，它们通常会产生较小的整体帧损失。

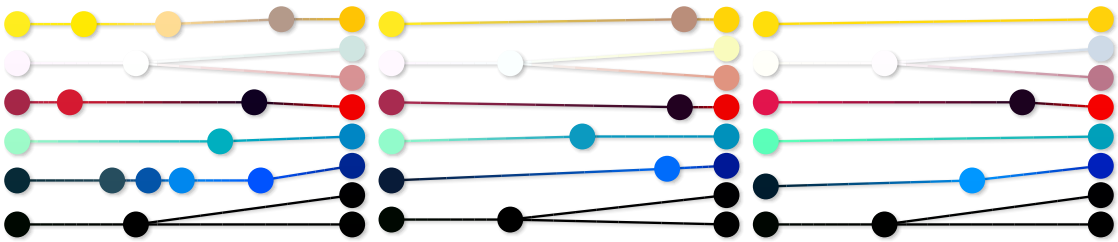
本文用不同的 β 值测试了帧块合并算法，并记录了帧块合并后所有帧的多面体调色板的平均顶点数目，统计结果如表 5.2 所示。从结果可以看出，与设置 $\beta = 0$ 相比，设置 $\beta > 0$ 通常会减少多面体调色板的平均顶点数目。实验结果表明，$\beta = 1.0$ 产生的多面体调色板的顶点数目适中，本章后续实验也将统一使用这一设定。

表 5.2　β 的不同取值对多面体调色板的影响

示例名称	$\beta = 0$	$\beta = \mathbf{1.0}$	$\beta = 2.0$	$\beta = 5.0$
森林 (图 5.13 (a))	8.0	7.4	7.4	7.4
街市 (图 5.13 (b))	8.0	8.0	8.0	8.0
云彩 (图 5.14 (a))	8.0	7.0	7.0	7.0
夜晚 (图 5.14 (c))	7.0	7.0	6.6	6.6
水域 (图 5.14 (d))	7.4	7.0	7.0	7.0

参数 η_2

在删除顶点的过程中，阈值 η_2 间接地控制了迭代过程何时停止（即当 $\mathcal{L}_v > \eta_2 \mathcal{L}_v^0$ 时，算法停止），因此决定了 4D 倾斜多面体的复杂性（即顶点数量）。通常，4D 几何调色板的顶点数目越少，越利于用户控制，但往往会导致更高的整体损失。反过来，顶点数目较多的 4D 几何调色板的整体损失更小，但难以控制和交互。为此，本文需要在交互的便利性和拓扑的复杂性之间平衡。图 5.8(a,b,c) 显示了使用不同的 η_2 值生成的几个 4D 几何调色板。可以清楚地看到，较大的 η_2 值会导致较少的顶点，而较少的顶点则会导致较大的整体视频损失。实验中，本文设定 $\eta_2 = 3.0$ 实现了较好的平衡。

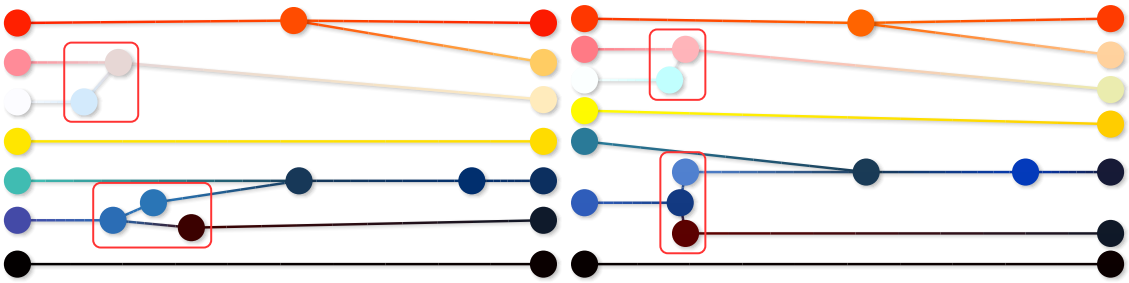


(a) $\eta_2 = 1.5$，26 个顶点　　(b) $\eta_2 = 3.0$，20 个顶点　　(c) $\eta_2 = 6.0$，18 个顶点

图 5.8　η_2 的不同取值对生成的几何调色板的影响

顶点优化的必要性

作为计算视频调色板的最后一步，顶点优化用于进一步减少整体视频损失并提高时间一致性。图 5.9比较了是否实施顶点优化产生的调色板。从结果可以看出，顶点优化进一步减少了尖锐的帧间边，使得帧间边更加平滑，从而提升了 4D 几何调色板的时间一致性。



(a) 实施顶点优化的结果　　(b) 未实施顶点优化的结果

图 5.9　顶点优化前后的调色板对比

5.10.2　视频重着色对比

现有的颜色编辑的方法如风格迁移或编辑传播，要么提供较少的用户控制（风格迁移），要么需要更多的用户交互（编辑传播），这些方法与本文方法的交互方式完全不同。因此这里只将本章方法与基于调色板的编辑方法进行对比，包括基于 RGB 凸包方法[12,39-40]和一个基于聚类的方法[11]。因为现有基于调色板的方法都是为图像重着色而设计的，所以本节先将这些方法扩展到视频场景然后再跟本文算法进行对比。

对于 RGB 凸包方法，为了从视频中提取调色板，我们将所有帧中的所有像素颜色集中起来，并计算出一个简化的 RGB 凸包[12,39]，然后进行局部顶点优化[40]。对于视频重着色任务，本文提供两种方式来扩展这个方法。一是在所有视频帧使用同一个共享的调色板。二是将第一帧和最后一帧视为两个关键帧，用户可以分别调整这两个关键帧的调色板，中间帧的调色板则通过第一帧和最后一帧的调色板线性插值得到。

对于聚类的方法[11]，我们以类似的方式将其拓展到视频重着色任务中。我们首先从所有帧的所有像素颜色中一起提取一个调色板，然后同样将第一帧和最后一帧视为两个关键帧，在关键帧上线性插值得到中间帧的调色板，依次实现随时间变化的视频颜色编辑效果。

此外，本文还将 RGB 凸包方法[12]直接扩展到 4D RGBT 空间。我们首先在 RGBT 空间中构建一个包裹所有视频像素的 4D 凸包。然后迭代地简化 4D RGBT

凸包，直到凸包的顶点数量减少到预定的阈值。类似地，我们也在每一帧对应的时刻对该 4D 凸包进行切片，以获得每一帧对应的多面体调色板。

图 5.10展示了本文方法与 RGB 凸包方法以及 RGBT 凸包方法的对比。输入视频“季节”展示了一个小树林一年四季的色彩变幻。颜色编辑的意图是使春天的大地更暗，夏天的树叶更绿，秋天的树叶更红，冬天的落叶更白。本文方法生成的几何调色板很好地捕捉了某些物体随时间变化的颜色，例如树叶的颜色从绿色到黄色再到橙色的变化。本文方法通过调整几个顶点的颜色成功实现了重着色的意图。相比之下，现有其它方法未能按预期调整视频中的树叶颜色。此外，与本文方法生成的多面体调色板相比，RGBT 凸包方法生成的多面体调色板（图 5.10倒数第二行）极其复杂且可解释性较差。

图 5.11展示了另一个视频“建筑”的重着色对比。对于该视频，用户期望将日落时分的云层调整得更红（第 60 帧处），并将天空调成墨绿色，但在日落之外（白天和晚上）保持天空的颜色不变。此外，用户还希望将视频开始处的建筑调得更亮，而将视频结束时的天空调得更暗。本文方法成功地实现了该编辑意图。其他方法由于缺乏对时间的精确控制而失败。例如，RGB 凸包方法在编辑中间帧的天空颜色时不可避免地改变了第一帧的天空颜色。

对视频“季节”（图 5.10）和“建筑”（图 5.11）计算的损失也验证了本文方法生成的调色板更加紧致。本文算法提取的 4D 几何调色板产生的紧致性损失分别为 0.036 和 0.1255，RGB 凸包的紧致性损失分别为 0.0649 和 0.3201，RGBT 凸包的紧致性损失为 0.2635 和 0.3515，相比之下，本文算法生成的调色板具有更好的紧致性。

5.10.3　图层分解结果

这小节展示视频的时变图层分解结果。如 5.3.3 小节所述，一旦获取到输入视频的 4D 几何调色板，就可以通过广义重心坐标插值的方法导出随时间变化的图层。具体来讲，首先在 4D 几何调色板中截取某一帧对应的 3D 多面体调色板，然后在 3D 多面体调色板内部对像素进行插值，最后导出每一帧对应的多个图层。因为本文提取的 4D 调色板具有很好的时间一致性，截取出的 3D 多面体调色板具有很好的连续性，因此保证了分解的图层在时间维度上光滑渐变。

图 5.12给出了两个视频的时变图层分解结果。针对每个例子，第一行是输入视频帧，第二行是对应的 4D 几何调色板，第三行是分解到的图层。我们知道，视频中的每一帧将会分解出多个图层，图层数目跟多面体调色板的颜色数目一致（每个多面体调色板的颜色对应一个图层），因此整个视频将分解得到数百个甚至上千个图层。为了便于观察，这里只展示了 4D 几何调色板中某条边上连续变化的颜色

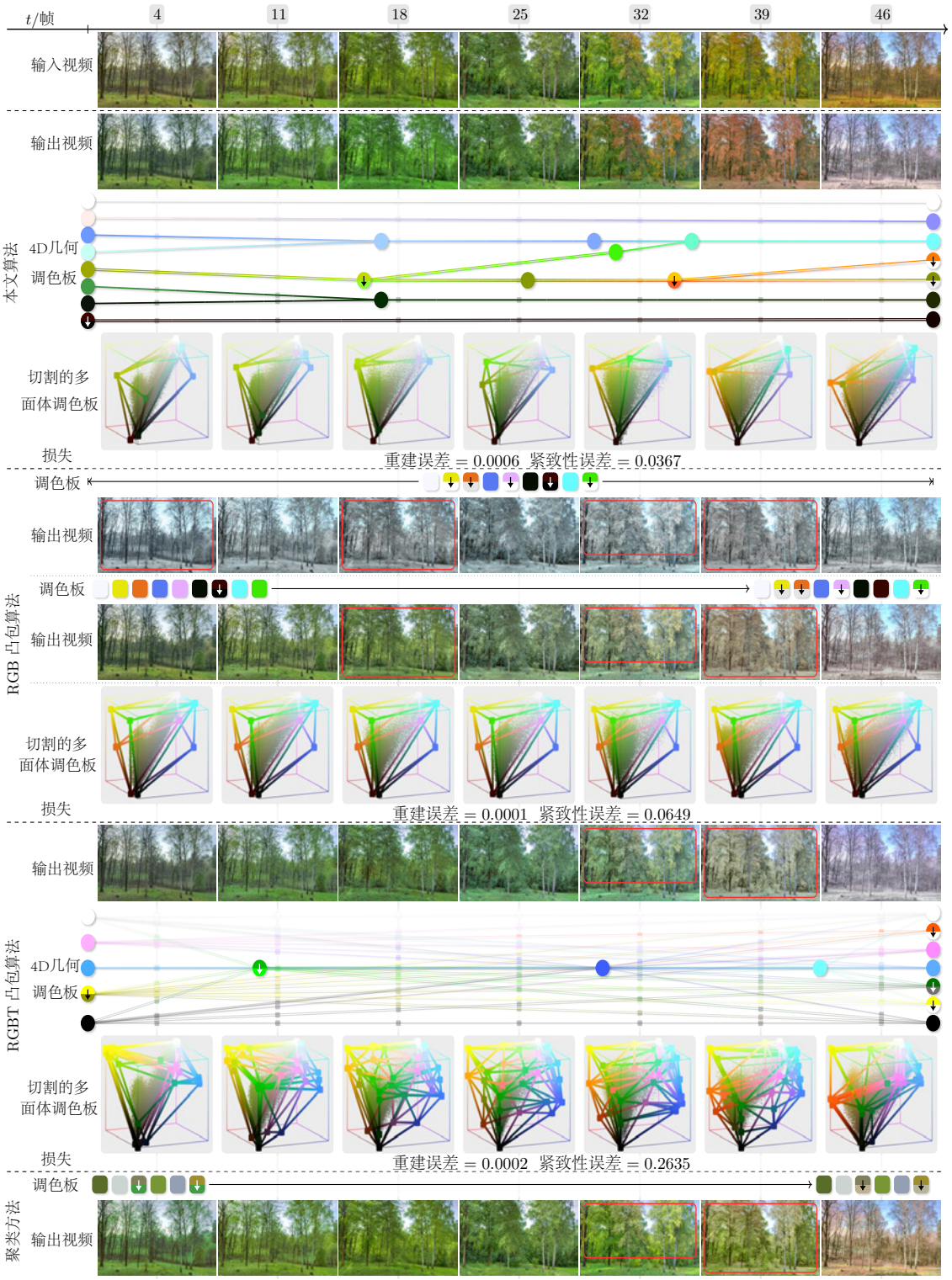


图 5.10　本文方法跟其它几种扩展的视频重着色算法比较 1

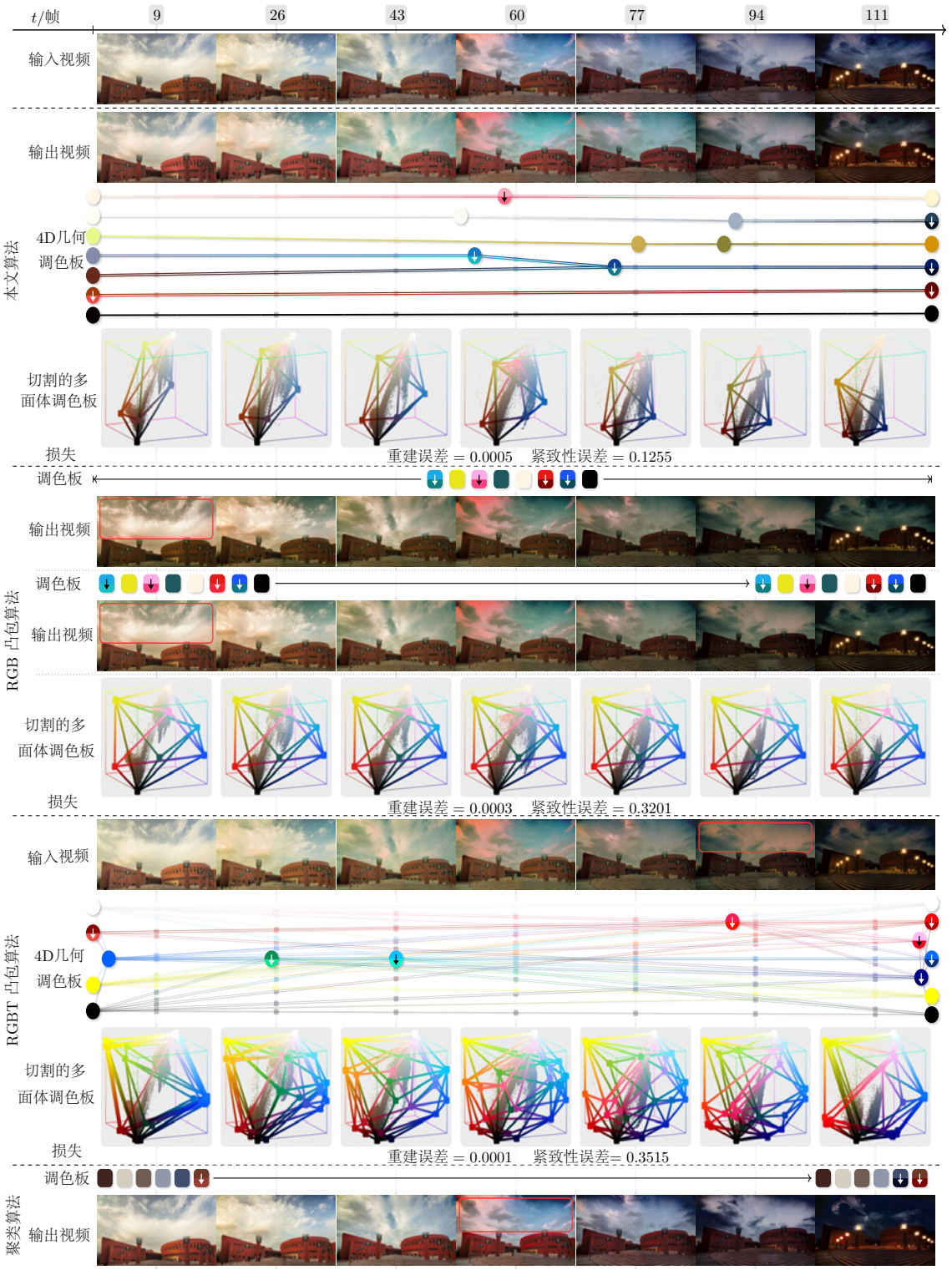


图 5.11　本文方法跟其它几种扩展的视频重着色算法比较 2

对应的各帧的图层。

图 5.12(a) 展示了建筑物从白昼到黑夜的延时效果，对应的 4D 几何调色板的第 6 行的帧间边有效地捕捉了建筑物从亮红到暗红的渐变过程。我们在这条边上均匀采样顶点（3D 多面体调色板的顶点），并展示了对应的各帧的图层。可以看出这些图层很好地覆盖了砖红色的建筑物，并且在视频的末尾部分，图层还零星地覆盖了天空部分，这对应于建筑物在夜晚灯光的辉映下出现的色溢效果。

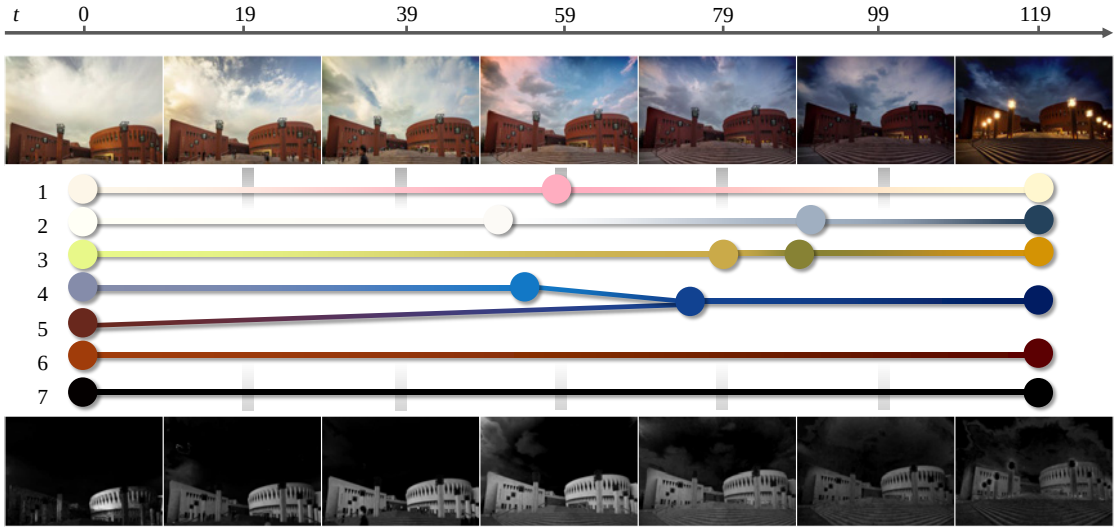
图 5.12(b) 展示了云彩的色彩变换过程，对应的 4D 几何调色板的第 5 行的帧间边有效地捕捉了云彩从墨绿到深蓝再到淡蓝的变化过程。我们也在这条边上均匀采样顶点，并展示了对应的各帧的图层。可以看出，这些图层准确地反映了图中蓝色部分从无到有，从亮到暗，面积从大到小的渐变过程。

5.10.4　视频编辑结果对比

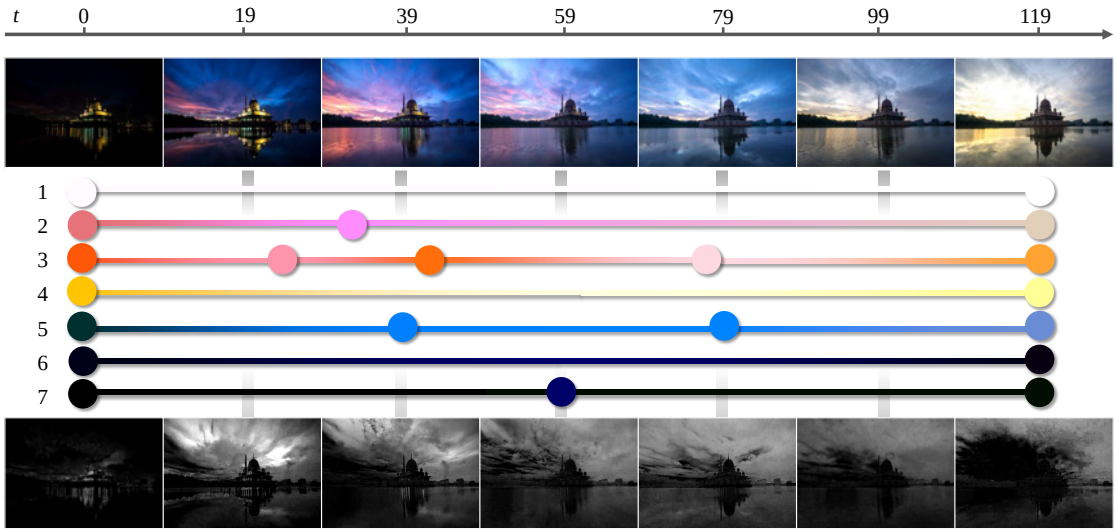
基于提取的 4D 几何调色板和分解的时变图层，本文做了大量的视频重着色实验。这里提供的大多数示例视频都是延时拍摄的，以便清晰地观察随时间变化的颜色。总体上，这些视频包含复杂和相对快速的颜色变化，因此，对这类视频进行重着色是一个挑战性的任务。图 5.13和 5.14中展示了大量的视频重着色结果。针对每一个示例，我们展示了视频中若干帧重着色前后的效果，左侧：第一行的时间轴上的数字对应了视频帧的索引，第二行是输入视频，第三行是输出视频；右侧：4D 几何调色板，每一个调色板的颜色用一个圆形表示，其中编辑过的颜色用两个半圆显示，上半部分是原始的颜色，下半部分是编辑后的颜色。

图 5.13(a) 所示的视频展示了秋天层林尽染的美景。这里将调色板中左侧的红色和黄色改为绿色，将右侧的绿色改为黄色，最终生成了森林从春天到秋天的渐变效果。图 5.13(b) 所示的视频展示了城市中琳琅满目的商标。用户通过对调色板的修改，将视频的色调统一调整为绿色。在图 5.13(c) 中，用户将天空的颜色从橙色调整为紫色。在图 5.13(d) 中，用户调整了天空和霞光的颜色。在图 5.13(e) 中，重着色后，光线的颜色逐渐从绿色变为黄色，看起来比原始视频更温暖。在图 5.13(f) 中，用户修改了大雁塔的灯光效果，使塔的颜色变化与喷泉的颜色变化一致。

图 5.14(a) 所示的视频展示了天空五彩斑斓的色彩变化。用户对云和寺庙施加了更多的色彩变化，使得颜色变化更加丰富多彩。图 5.14(b) 所示的视频展示了城市从天亮到日出的延时效果。用户修改了城市的灯光以及阳光的颜色，使之看起来呈现出一种朦胧效果。图 5.14(c) 所示的视频展示了滨海小镇傍晚的风光。通过对调色板编辑，使得重着色后的视频呈现出冷色调的效果。图 5.14(d) 所示的视频展示了“日出江花红胜火”的日出美景。用户通过对调色板编辑，实现了云彩和水面从暗到亮的渐变效果。



(a) 视频“建筑”的时变图层分解结果



(b) 视频“云彩”的时变图层分解结果

图 5.12　两个视频的时变图层分解结果

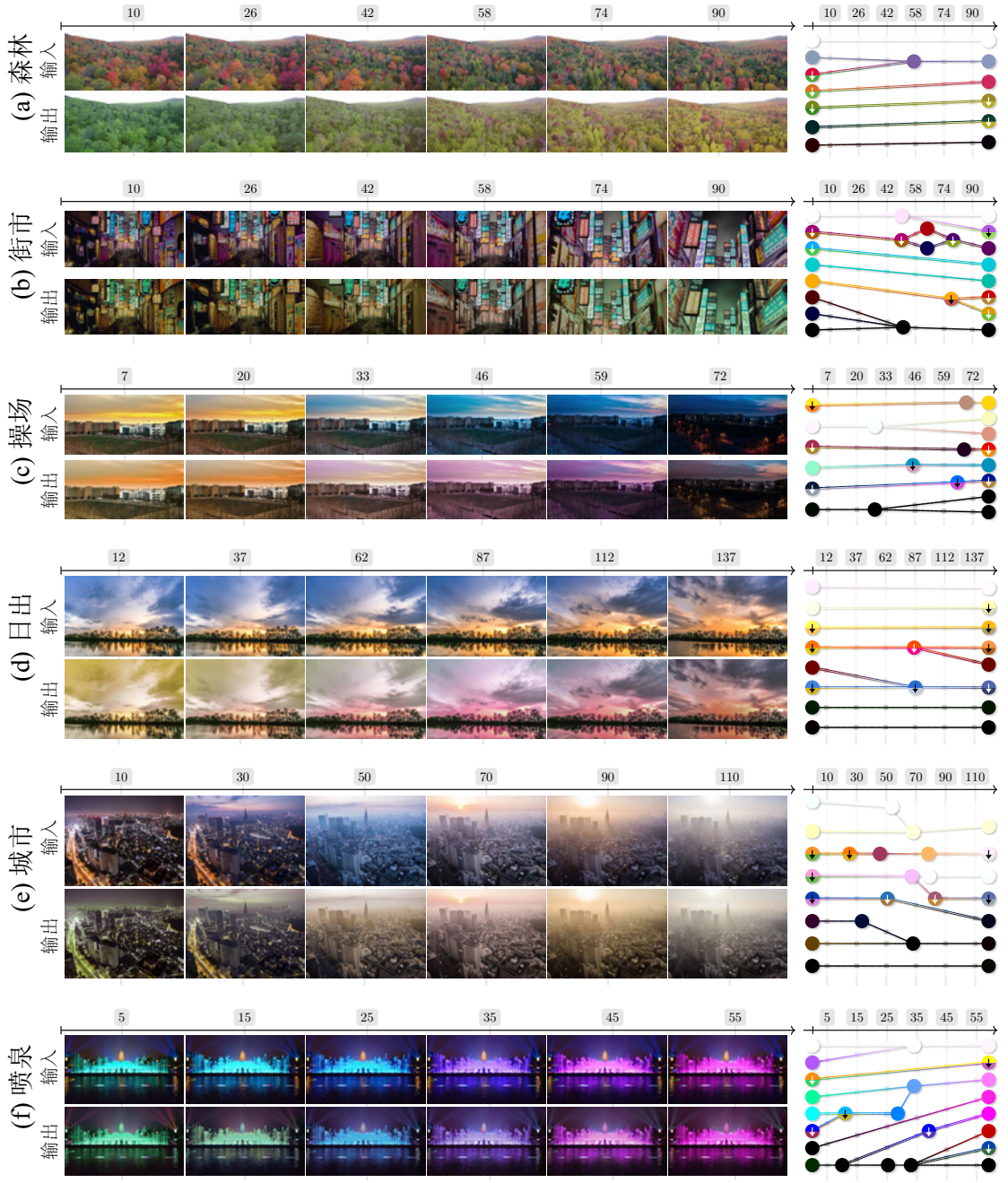


图 5.13　视频重着色结果展示 1

本文提供的图像用户界面还支持向几何调色板添加新的关键帧（即颜色），以实现更复杂的非线性的颜色变化。图 5.15中展示了这样一个例子，通过增加新的颜色，引入了输入视频中不存在的非线性的颜色变化，使得这里的天空、云和山从淡紫色变为蓝色，然后再回到淡紫色。

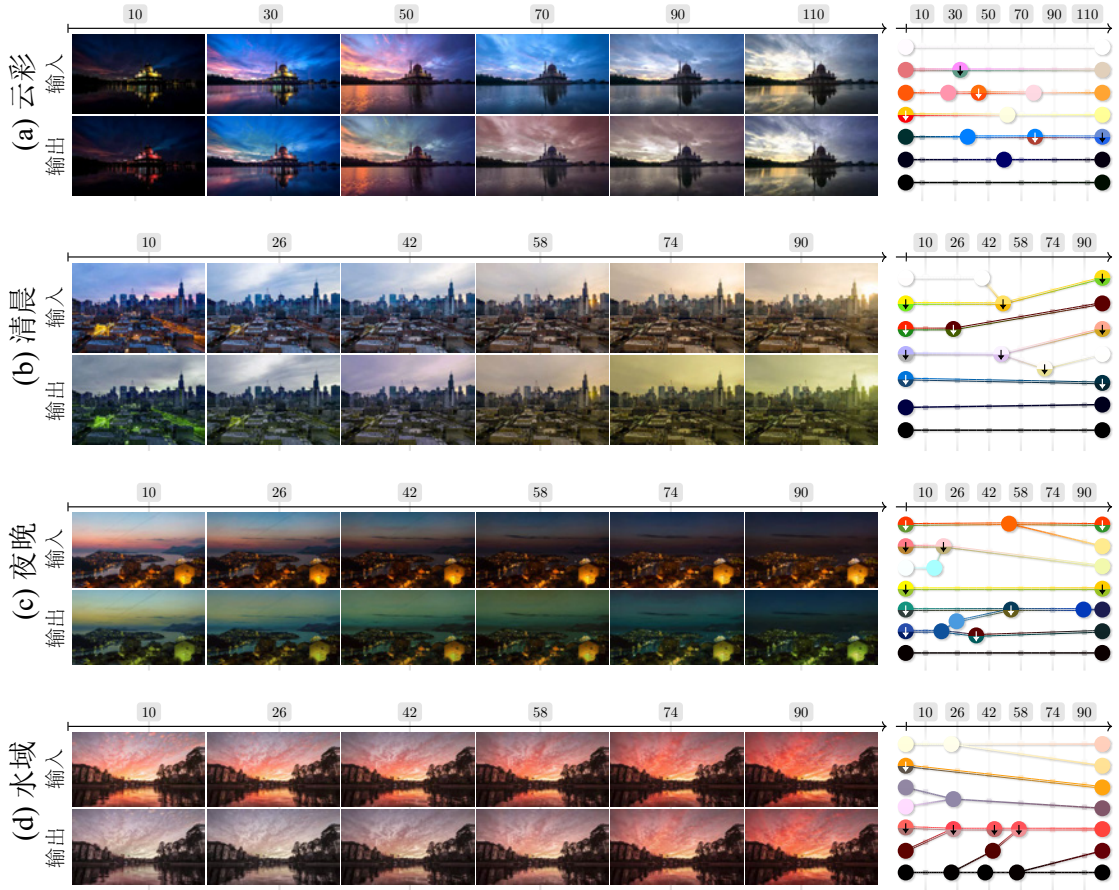


图 5.14　视频重着色结果展示 2

图 5.15　添加关键帧（第一行）与不添加关键帧（第二行）的重着色效果对比

5.10.5　性能统计

表 5.3提供了所有例子的统计数据，包括帧数、视频包含的像素数目（M 表示百万）、提取的 4D 几何调色板的顶点数目（颜色数目）、预处理时间（为所有帧提取初始多面体调色板并执行顶点优化的时间）和本文几何调色板提取方法的运行时间。一旦提取到给定输入视频的几何调色板，我们就可以提供实时的重着色编辑并产生自然的重着色效果。

表 5.3　本章各示例的统计结果

示例名称	帧数	像素数目	顶点数目	预处理用时	提取调色板用时
季节 (图 5.10)	50	13M	23	18 分钟	12 分钟
建筑 (图 5.11)	120	96M	20	36 分钟	40 分钟
森林 (图 5.13(a))	100	46M	14	48 分钟	33 分钟
街市 (图 5.13(b))	100	86M	23	42 分钟	48 分钟
操场 (图 5.13(c))	80	64M	20	33 分钟	35 分钟
日出 (图 5.13(d))	150	52M	17	35 分钟	52 分钟
城市 (图 5.13(e))	120	58M	26	22 分钟	55 分钟
喷泉 (图 5.13(f))	60	17M	24	26 分钟	12 分钟
云彩 (图 5.14(a))	120	117M	22	30 分钟	61 分钟
清晨 (图 5.14(b))	100	35M	18	38 分钟	40 分钟
夜晚 (图 5.14(c))	100	92M	22	26 分钟	56 分钟
水域 (图 5.14(d))	100	92M	23	26 分钟	45 分钟
雪山 (图 5.15)	120	55M	20	31 分钟	45 分钟

5.10.6　用户实验

本文还进行了一项用户实验，以评估视频调色板的质量，重着色效果以及图形用户界面的便捷性。我们共邀请了 16 名用户（有 7 名男性和 9 名女性，年龄在 21 至 26 岁之间）参与实验。其中，9 名用户来自于清华大学美术学院，他们中很多人有丰富的视频编辑经验。每个用户被要求执行四个视频重着色任务。其中，两项任务有具体的重着色目标：1）使“建筑”视频中的音乐厅看起来更老旧，使“季节”视频中的四季更加分明。另外两项任务是开放式的，用户可以用他们喜欢的方式自由地编辑两个给定的视频。在重着色任务之前，我们提供了简单的教程和一个试验，让参与者学习使用本文的重着色工具。待所有任务完成后，我们给用户发放问卷，并要求他们对调色板的准确性、重着色的质量和用户体验等几方面

问题进行评估。

总体上，用户实验得到了非常积极的反馈。例如，100％的参与者对用户界面感到满意或非常满意，87.5％的用户认为本文提取的几何调色板很好地反映了视频的颜色分布，87.5％的参与者认为可以通过本文提供的重着色工具轻松地实现编辑意图，93.5％的参与者认为视频重着色的结果符合他们的期望。许多参与者都熟悉 DaVinci Resolve，Adobe After Effects 等专业视频编辑软件的颜色编辑操作。通过对比，受试者认为本文提供的重着色工具更加灵活和方便。用户实验进一步说明了本文算法的有效性以及重着色工具的易用性。

5.11　本章小结

本章首次将面向图像重着色的图层分解算法拓展到视频场景，并成功将其应用到视频重着色任务中。提出的方法以 4D RGBT 空间中的倾斜多面体为核心，它能够准确地捕捉视频的颜色变化，同时将其作为调色板供用户使用，具有非常友好的体验。大量对比实验和用户实验表明，提出的方法能够有效地从视频中抽取高质量的调色板，并进一步将视频分解为一组随时间光滑渐变的图层。在此基础上，实现了高效、自然的视频颜色编辑。

提出的方法仍然存在一些缺陷。1）虽然本文方法能够支持常用的颜色编辑，但基于线性混合权重的方法不能处理非线性的颜色编辑如色调映射等操作。2）提取的 4D 几何调色板只能捕捉整体的颜色变化但不能跟踪特定物体的颜色变化，因此无法做到语义层次的局部编辑。3）本文算法对于拍摄镜头剧烈变化的视频可能无法分解出光滑渐变的图层，因为本章假设色彩需要随时间平滑变化。

当前算法可以在几个方面进一步优化。首先，本章算法在视频预处理和调色板提取阶段均需花费大量时间，后续考虑在视频的关键帧上构建初始的 4D 倾斜多面体来降低算法的开销并尝试通过并行计算来加速调色板的提取过程。其次，在帧块合并中，当前算法每次会枚举所有相邻的帧块并计算它们的能量损失，这将耗费大量时间，后续考虑对这个枚举过程进行优化以降低时间复杂度。未来我们还希望将 4D RGBT 几何调色板的分段线性性质进一步泛化，以允许使用曲线甚至是曲面来表示图像或视频的颜色分布（如 Shugrina 等[145]提出的非线性调色板）。

第 6 章　总结与展望

本文主要围绕图层分解技术展开研究，针对不同的应用场景，分别提出了面向图像矢量化的半透明图层分解算法，面向图像内容感知重着色的加性图层分解算法和面向视频重着色的时变图层分解算法。提出的算法取得了一系列进展和成果，促进了相关领域的发展。接下来，我们对本文研究工作进行简要总结，指出本文算法存在的缺陷并对未来可能的改进方向进行展望。

6.1　本文工作总结

本文第三章提出了一种面向图像矢量化的半透明图层分解算法。针对输入的光栅图像和对应的区域分割结果，算法输出一组空间上相互重叠的半透明矢量图层，并且这些图层经过 Alpha 混合可以完美地重建出输入图像。针对给定光栅图像的半透明图层分解是一个极具挑战性的问题。首先，它是一个欠约束的问题，理论上存在无穷多种可能的图层分解方案。其次，针对每一种可能的图层分解方案，需要同时预测图层配置（图层的数目，图层的重叠顺序以及每个图层的蒙板）和图层参数（每个图层的线性渐变参数）。其中，图层配置参数是离散的，图层参数是连续的，二者难以统一优化求解。提出的算法基于两个重要的发现：第一，图层配置可以形式化为一棵区域支持树；第二，通过枚举生成树可以从区域邻接图中导出所有可能的区域支持树。为此，本文提出一种基于图的半透明图层分解算法。首先根据区域分割图像构建区域邻接图，并通过一些感知规则对其简化以排除大量不合理的图层分解方案。然后通过穷举的方式搜索所有可能的生成树（图层配置）。接下来对每一个图层配置执行必要的图层合并，并通过优化的方法估计图层参数。最后将质量最好的图层分解方案返回给用户。提出的方法在给定输入的情况下，实现了全自动的半透明图层分解。定性的对比和用户实验表明：本文算法分解的图层能够更好地反映图像本身的形状结构，更加方便用户实施后续编辑。

本文第四章提出了一种面向图像内容感知重着色的加性图层分解算法。算法的输入是一张图像，输出是一个调色板和一组语义相关的图层。现有方法通常将图像视为 RGB 空间或 RGBXY 空间的点集，通过聚类或计算凸包的方法提取调色板并进行图层分解。由于缺乏语义信息，在颜色编辑过程中，现有方法无法区分颜色相似的不同物体，当用户修改调色板中的某个颜色时，所有与之颜色相似的像素均会发生改变。为了解决这个问题，本文将图像的低级颜色特征同高级语义特征结合起来，将图像看作高维空间的点集，并在高维空间提取调色板和分解图层。

在调色板提取阶段，本文使用改进的 K-means 算法对输入图像对应的高维点集进行聚类，最后将收敛的聚类中心作为图像的调色板。在图层分解阶段，通过径向基函数定义调色板中每个条目的相似度函数，然后根据输入图像中每个像素跟调色板条目的相似度将图像分解为多个图层。实验结果表明，本文算法分解的图层很好地反映了图像的语义信息。在颜色编辑任务中，提出的方法能够有效地将图像中颜色相似的不同物体区分开，实现了语义级别的、内容感知的图像重着色，为用户实施针对性的局部编辑提供了极大的便利。

本文第五章提出了一种面向视频重着色的时变图层分解算法。提出的算法首次将基于凸包的加性图层分解算法拓展到视频场景。针对给定的视频，该算法输出一个 4D 几何调色板和一组随时间光滑渐变的图层。在视频编辑任务中，用户通过修改 4D 几何调色板控制视频的颜色变化。提出的算法主要分为两步即 4D 几何调色板提取和时变图层分解。针对 4D 几何调色板提取，算法首先在每一个视频帧构建 3D RGB 凸包，并将所有相邻帧的 3D RGB 凸包“粘连”到一起，构建初始的 4D RGBT 倾斜多面体。然后通过帧块合并、顶点删除等一系列操作迭代地简化初始 4D RGBT 倾斜多面体的拓扑结构。接下来逐个优化 4D RGBT 倾斜多面体的顶点以进一步降低能量误差，并将优化后的 4D RGBT 倾斜多面体作为视频的 4D 几何调色板。最后通过均值坐标插值的方法将视频分解为一组时间上光滑渐变的图层。实验结果表明，本文算法提取的调色板有效地捕捉了视频中主要颜色的变化，分解的图层在时间轴上具有很好的连续性。在此基础上，本文实现了操作便捷、效果自然的视频颜色编辑。

6.2　未来工作展望

针对面向图像矢量化的半透明图层分解，提出的方法仍有一些不足之处值得进一步改进。首先，为了降低求解难度，算法需要用户提供额外的图像分割结果作为输入。然而这对于没有经验的用户来讲相对繁琐，一定程度上加重了用户的使用负担。图像分割目前仍然是计算机视觉领域的开放问题，尤其对于色彩变化较为丰富的图像，现有技术难以实现高质量的图像分割。近期，Kirillov 等[106]提出的 SAM 和 Wang 等[107]提出的 SegGPT 等通用分割模型取得了突破性的进展，未来我们期望在这些算法基础之上实现无需区域分割作为输入的半透明图层分解。其次，为了编辑的便捷性，目前只考虑了一种线性渐变的图层，因此颜色拟合能力比较有限。将来期望考虑径向渐变、二次渐变等更多样化的图层以模拟更复杂的颜色变化。再次，多层矢量化问题本质上存在无穷解，提出的方法通过感知规则裁减了搜索空间，因此只能枚举一些符合感知规则的图层分解方案。未来我们希望

利用更多的感知规则以满足不同用户的个性化需求。最后，针对复杂的例子，提出的算法计算开销仍然较大。为此，我们期望对求解过程进一步优化，以提升算法的运行效率。

针对面向图像内容感知重着色的加性图层分解，提出的算法有两点缺陷可以进一步改进。首先，我们通过 K-means 算法提取图像的调色板，而这个过程需要用户手动指定聚类中心的数目。指定的数目过多将导致调色板包含过多的颜色，不利于后续的颜色编辑。指定的数目过少，将使得分解的图层无法有效地区分图像中不同语义的物体，难以对图像中的局部物体实施针对性的编辑。为此，我们期望通过一些自适应的聚类算法如 DBSCAN，MeanShift 等实现自动化的调色板提取。其次，提出的图层分解算法依赖深度卷积神经网络的语义预测结果，对于很多例子，当前网络预测的语义特征仍然很不准确，因此可能影响到后续的图层分解和图像重着色结果。将来我们期望对语义预测网络进行改进并对语义预测结果进行必要的优化以提升语义预测的准确性。

针对面向视频重着色的时变图层分解，提出的算法主要存在以下两个缺陷。首先，提出的算法不够高效。在预处理阶段，需要花费较多时间提取视频帧的 RGB 凸包。在调色板提取阶段，无论是帧块合并还是顶点删除都涉及大量的迭代操作，因此非常耗时。针对分辨率较高的视频，调色板提取往往需要花费数十分钟。我们期望在两方面改进这一点：第一，通过提取视频关键帧，并在关键帧上构建初始 4D 倾斜多面体来降低初始调色板的拓扑复杂性，进而降低帧块合并和顶点删除等后续操作的时间复杂度，提升算法的运行效率；第二，我们期望探索更多的视频调色板表示方式，比如使用 RGBT 4D 空间的 Bezier 曲线或曲面表示调色板，并最大程度地避免大量的迭代计算。其次，本文算法分解的图层虽然在时间上具有很好的一致性但不具备空间一致性，即无法追踪视频中具体物体的颜色变化，因此无法实现语义级别的颜色编辑。未来我们期望将第 4 章介绍的面向图像内容感知重着色的加性图层分解技术拓展到视频场景，实现时空一致的图层分解，并在此基础上实现内容感知的、语义级别的视频重着色。

参考文献

[1] Sun J, Liang L, Wen F, et al. Image vectorization using optimized gradient meshes[J]. ACM Transactions on Graphics (TOG), 2007, 26(3): 1-8.

[2] Lai Y K, Hu S M, Martin R R. Automatic and topology-preserving gradient mesh generation for image vectorization[J]. ACM Transactions on Graphics (TOG), 2009, 28(3): 1-8.

[3] Orzan A, Bousseau A, Winnemöller H, et al. Diffusion curves: a vector representation for smooth-shaded images[J]. ACM Transactions on Graphics (TOG), 2008, 27(3): 1-8.

[4] Xie G, Sun X, Tong X, et al. Hierarchical diffusion curves for accurate automatic image vectorization[J]. ACM Transactions on Graphics (TOG), 2014, 33(6): 1-11.

[5] Prévost R, Jarosz W, Sorkine-Hornung O. A vectorial framework for ray traced diffusion curves [J]. Computer Graphics Forum (CGF), 2015, 34(1): 253-264.

[6] Li Y, Zhai X, Hou F, et al. Vectorized painting with temporal diffusion curves[J]. IEEE Transactions on Visualization and Computer Graphics, 2021, 27(1): 228-240.

[7] Xia T, Liao B, Yu Y. Patch-based image vectorization with automatic curvilinear feature alignment[J]. ACM Transactions on Graphics (TOG), 2009, 28(5): 1-10.

[8] Richardt C, Lopez-Moreno J, Bousseau A, et al. Vectorising bitmaps into semi-transparent gradient layers[J]. Computer Graphics Forum (CGF), 2014, 33(4): 11-19.

[9] Favreau J D, Lafarge F, Bousseau A. Photo2clipart: Image abstraction and vectorization using layered linear gradients[J]. ACM Transactions on Graphics (TOG), 2017, 36(6): 1-11.

[10] Porter T, Duff T. Compositing digital images[C]//Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques. 1984: 253-259.

[11] Chang H, Fried O, Liu Y, et al. Palette-based photo recoloring.[J]. ACM Transactions on Graphics (TOG), 2015, 34(4): 139-1.

[12] Tan J, Lien J M, Gingold Y. Decomposing images into layers via RGB-space geometry[J]. ACM Transactions on Graphics (TOG), 2016, 36(1): 1-14.

[13] Floater M S. Generalized barycentric coordinates and applications[J]. Acta Numerica, 2015, 24: 161-214.

[14] Browne C B, Powley E, Whitehouse D, et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games, 2012, 4(1): 1-43.

[15] 林生佑, 潘瑞芳, 杜辉, 等. 数字抠图技术综述[J]. 计算机辅助设计与图形学学报, 2007, 19 (4): 473-479.

[16] Wang J, Cohen M F, et al. Image and video matting: a survey[J]. Foundations and Trends® in Computer Graphics and Vision, 2008, 3(2): 97-175.

[17] Chuang Y Y, Curless B, Salesin D H, et al. A bayesian approach to digital matting[C]//2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2001: 1-8.

[18] Hillman P, Hannah J, Renshaw D. Alpha channel estimation in high resolution images and image sequences[C]//2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2001: 1-6.

[19] Rother C, Kolmogorov V, Blake A. “GrabCut”: Interactive foreground extraction using iterated graph cuts[J]. ACM Transactions on Graphics (TOG), 2004, 23(3): 309-314.

[20] Wang J, Cohen M F. Optimized color sampling for robust matting[C]//2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2007: 1-8.

[21] Gastal E S, Oliveira M M. Shared sampling for real-time alpha matting[J]. Computer Graphics Forum (CGF), 2010, 29(2): 575-584.

[22] He K, Rhemann C, Rother C, et al. A global sampling method for alpha matting[C]//2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2011: 2049-2056.

[23] Shahrian E, Rajan D. Weighted color and texture sample selection for image matting[C]//2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2012: 718-725.

[24] Shahrian E, Rajan D, Price B, et al. Improving image matting using comprehensive sampling sets[C]//2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2013: 636-643.

[25] Sun J, Jia J, Tang C K, et al. Poisson matting[J]. ACM Transactions on Graphics (TOG), 2004, 23(3): 315-321.

[26] Levin A, Lischinski D, Weiss Y. A closed-form solution to natural image matting[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 30(2): 228-242.

[27] Levin A, Rav-Acha A, Lischinski D. Spectral matting[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008, 30(10): 1699-1712.

[28] Chen Q, Li D, Tang C K. KNN matting[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(9): 2175-2188.

[29] Cho D, Tai Y W, Kweon I. Natural image matting using deep convolutional neural networks[C]// Proceedings of the 14th European Conference on Computer Vision (ECCV). 2016: 626-643.

[30] Shen X, Tao X, Gao H, et al. Deep automatic portrait matting[C]//Proceedings of the 14th European Conference on Computer Vision (ECCV). 2016: 92-107.

[31] Xu N, Price B, Cohen S, et al. Deep image matting[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017: 2970-2979.

[32] Zhu B, Chen Y, Wang J, et al. Fast deep matting for portrait animation on mobile phone[C]// Proceedings of the 25th ACM international Conference on Multimedia. 2017: 297-305.

[33] 王宸昊, 张慧. 粗糙语义引导的同时预测前背景的透明度图估计算法[J]. 计算机辅助设计与图形学学报, 2022, 34(9): 1432-1440.

[34] 冉清, 冯结青. 人体前景的自动抠图算法[J]. 计算机辅助设计与图形学学报, 2020, 32(2): 277-286.

[35] Lutz S, Amplianitis K, Smolic A. AlphaGAN: Generative adversarial networks for natural image matting[J/OL]. CoRR, 2018, abs/1807.10088[2023-03-01]. https://arxiv.org/abs/1807.10088.

[36] Ren X, Liu Y, Song C. A generative adversarial framework for optimizing image matting and harmonization simultaneously[C]//2021 IEEE International Conference on Image Processing (ICIP). 2021: 1354-1358.

[37] Park G, Son S, Yoo J, et al. Matteformer: Transformer-based image matting via prior-tokens [C]//2022 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2022: 11696-11706.

[38] Cai H, Xue F, Xu L, et al. TransMatting: Enhancing transparent objects matting with transformers[C]//Proceedings of the 17th European Conference on Computer Vision (ECCV). 2022: 253-269.

[39] Tan J, Echevarria J, Gingold Y. Efficient palette-based decomposition and recoloring of images via rgbxy-space geometry[J]. ACM Transactions on Graphics (TOG), 2018, 37(6): 1-10.

[40] Wang Y, Liu Y, Xu K. An improved geometric approach for palette-based image decomposition and recoloring[J]. Computer Graphics Forum (CGF), 2019, 38(7): 11-22.

[41] Bock H H. Clustering methods: A history of k-means algorithms[M/OL]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007: 161-172. https://doi.org/10.1007/978-3-540-73560-1_15.

[42] Zhang Q, Xiao C, Sun H, et al. Palette-based image recoloring using color decomposition optimization[J]. IEEE Transactions on Image Processing, 2017, 26(4): 1952-1964.

[43] Shen L, Yeo C. Intrinsic images decomposition using a local and global sparse representation of reflectance[C]//2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2011: 697-704.

[44] Chen Q, Koltun V. A simple model for intrinsic image decomposition with depth cues[C]// Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2013: 241-248.

[45] Bi S, Han X, Yu Y. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition[J]. ACM Transactions on Graphics (TOG), 2015, 34(4): 1-12.

[46] Laffont P Y, Bazin J C. Intrinsic decomposition of image sequences from local temporal variations[C]//Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2015: 433-441.

[47] Bonneel N, Kovacs B, Paris S, et al. Intrinsic decompositions for image editing[J]. Computer Graphics Forum (CGF), 2017, 36(2): 593-609.

[48] Janner M, Wu J, Kulkarni T D, et al. Self-supervised intrinsic image decomposition[C]// Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS). 2017: 5938-5948.

[49] Lettry L, Vanhoey K, Van Gool L. Unsupervised deep single-image intrinsic decomposition using illumination-varying image sequences[J]. Computer Graphics Forum (CGF), 2018, 37 (7): 409-419.

[50] Bi S, Kalantari N K, Ramamoorthi R. Deep hybrid real and synthetic training for intrinsic decomposition[J/OL]. CoRR, 2018, abs/1807.11226[2023-03-01]. https://arxiv.org/abs/1807.11226.

[51] Koyama Y, Goto M. Decomposing images into layers with advanced color blending[J]. Computer Graphics Forum (CGF), 2018, 37(7): 397-407.

[52] Horita D, Aizawa K, Suzuki R, et al. Fast nonlinear image unblending[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2022: 2051-2059.

[53] Aharoni-Mack E, Shambik Y, Lischinski D. Pigment-based recoloring of watercolor paintings [C]//Proceedings of the Symposium on Non-Photorealistic Animation and Rendering. 2017: 1-11.

[54] Kubelka P, Munk F. An article on optics of paint layers[J]. Z. Tech. Phys, 1931, 12(593-601): 259-274.

[55] Tan J, DiVerdi S, Lu J, et al. Pigmento: Pigment-based image analysis and editing[J]. IEEE Transactions on Visualization and Computer Graphics, 2018, 25(9): 2791-2803.

[56] Gordon W J, Riesenfeld R F. B-spline curves and surfaces[M/OL]//Computer Aided Geometric Design. 1974: 95-126. https://www.sciencedirect.com/science/article/pii/B9780120790500500114. DOI: https://doi.org/10.1016/B978-0-12-079050-0.50011-4.

[57] Catmull E, Clark J. Recursively generated b-spline surfaces on arbitrary topological meshes[J]. Computer-Aided Design, 1978, 10(6): 350-355.

[58] Piegl L. On NURBS: a survey[J]. IEEE Computer Graphics and Applications, 1991, 11(01): 55-71.

[59] 张礼林, 王国瑾. 带 B 样条曲率线的 NURBS 曲面设计[J]. 计算机辅助设计与图形学学报, 2018, 30(9): 1692-1698.

[60] 王文涛, 汪国昭. 带形状参数的双曲多项式均匀 B 样条[J]. 软件学报, 2005, 16(4): 625-633.

[61] 王文涛, 汪国昭. 带形状参数的三角多项式均匀 B 样条[J]. 计算机学报, 2005, 28(7): 1192-1198.

[62] Prautzsch H, Boehm W, Paluszny M. Bézier and B-spline techniques[M]. Springer, 2002.

[63] Böhm W, Farin G, Kahmann J. A survey of curve and surface methods in cagd[J]. Computer Aided Geometric Design, 1984, 1(1): 1-60.

[64] 吴晓勤, 韩旭里. 三次 Bézier 曲线的扩展[J]. 工程图学学报, 2005, 26(6): 98-102.

[65] Lecot G, Levy B. ARDECO: Automatic region detection and conversion[C]//17th Eurographics Symposium on Rendering (EGSR'06). 2006: 349-360.

[66] Chen K W, Luo Y S, Lai Y C, et al. Image vectorization with real-time thin-plate spline[J]. IEEE Transactions on Multimedia, 2019, 22(1): 15-29.

[67] Zhu H, Cao J, Xiao Y, et al. TCB-spline-based image vectorization[J]. ACM Transactions on Graphics (TOG), 2022, 41(3): 1-17.

[68] Ailisto H, Pietikainen M. Ferguson patch based method for representation of 3-d scenes[C]// Proceedings of 1987 IEEE International Conference on Robotics and Automation. 1987: 1737-1740.

[69] Xiao Y, Wan L, Leung C S, et al. Example-based color transfer for gradient meshes[J]. IEEE Transactions on Multimedia, 2012, 15(3): 549-560.

[70] Wan L, Xiao Y, Dou N, et al. Scribble-based gradient mesh recoloring[J]. Multimedia Tools and Applications, 2018, 77: 13753-13771.

[71] Bezerra H, Eisemann E, DeCarlo D, et al. Diffusion constraints for vector graphics[C]// Proceedings of the 8th International Symposium on Non-photorealistic Animation and Rendering. 2010: 35-42.

[72] Jeschke S, Cline D, Wonka P. Estimating color and texture parameters for vector graphics[J]. Computer Graphics Forum (CGF), 2011, 30(2): 523-532.

[73] Jeschke S, Cline D, Wonka P. Rendering surface details with diffusion curves[J]. ACM Transactions on Graphics (TOG), 2009, 28(5): 1–8.

[74] Maini R, Aggarwal H. A comprehensive review of image enhancement techniques[J/OL]. CoRR, 2010, abs/1003.4053[2023-03-01]. https://arxiv.org/abs/1003.4053.

[75] Guo X, Li Y, Ling H. LIME: Low-light image enhancement via illumination map estimation [J]. IEEE Transactions on Image Processing, 2016, 26(2): 982-993.

[76] Lore K G, Akintayo A, Sarkar S. LLNet: A deep autoencoder approach to natural low-light image enhancement[J]. Pattern Recognition, 2017, 61: 650-662.

[77] Zhang Y, Zhang J, Guo X. Kindling the darkness: A practical low-light image enhancer[C]// Proceedings of the 27th ACM International Conference on Multimedia. 2019: 1632-1640.

[78] Jiang Y, Gong X, Liu D, et al. Enlightengan: Deep light enhancement without paired supervision [J]. IEEE Transactions on Image Processing, 2021, 30: 2340-2349.

[79] An X, Pellacini F. AppProp: All-pairs appearance-space edit propagation[J]. ACM Transactions on Graphics (TOG), 2008, 27(3): 1–9.

[80] Xu K, Li Y, Ju T, et al. Efficient affinity-based edit propagation using kd tree[J]. ACM Transactions on Graphics (TOG), 2009, 28(5): 1-6.

[81] Chen X, Zou D, Zhao Q, et al. Manifold preserving edit propagation[J]. ACM Transactions on Graphics (TOG), 2012, 31(6): 1-7.

[82] Endo Y, Iizuka S, Kanamori Y, et al. Deepprop: Extracting deep features from a single image for edit propagation[J]. Computer Graphics Forum (CGF), 2016, 35(2): 189-201.

[83] Gui Y, Zeng G. Joint learning of visual and spatial features for edit propagation from a single image[J]. The Visual Computer, 2020, 36(3): 469-482.

[84] Gatys L A, Ecker A S, Bethge M. A neural algorithm of artistic style[J/OL]. CoRR, 2015, abs/1508.06576[2023-03-01]. https://arxiv.org/abs/1508.06576.

[85] Gatys L A, Ecker A S, Bethge M. Image style transfer using convolutional neural networks[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 2414-2423.

[86] Li Y, Liu M Y, Li X, et al. A closed-form solution to photorealistic image stylization[C]// Proceedings of the European Conference on Computer Vision on Computer Vision (ECCV). 2018: 453-468.

[87] Jing Y, Yang Y, Feng Z, et al. Neural style transfer: A review[J]. IEEE Transactions on Visualization and Computer Graphics, 2019, 26(11): 3365-3385.

[88] Aksoy Y, Aydin T O, Smolić A, et al. Unmixing-based soft color segmentation for image manipulation[J]. ACM Transactions on Graphics (TOG), 2017, 36(2): 1-19.

[89] Lin S, Fisher M, Dai A, et al. Layerbuilder: Layer decomposition for interactive image and video color editing[J/OL]. CoRR, 2017, abs/1701.03754[2023-03-01]. https://arxiv.org/abs/1701.03754.

[90] Lin S, Hanrahan P. Modeling how people extract color themes from images[C]//Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2013: 3101-3110.

[91] Zhang Q, Nie Y, Zhu L, et al. A blind color separation model for faithful palette-based image recoloring[J]. IEEE Transactions on Multimedia, 2021, 24: 1545-1557.

[92] Cho J, Yun S, Mu Lee K, et al. Palettenet: Image recolorization with given color palette[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR). 2017: 62-70.

[93] Akimoto N, Zhu H, Jin Y, et al. Fast soft color segmentation[C]//2020 IEEE Conference on Computer Vision and Pattern Recognition. 2020: 8277-8286.

[94] Zhao N, Zheng Q, Liao J, et al. Selective region-based photo color adjustment for graphic designs[J]. ACM Transactions on Graphics (TOG), 2021, 40(2): 1-16.

[95] Wagemans J, Elder J H, Kubovy M, et al. A century of gestalt psychology in visual perception i. perceptual grouping and figure-ground organization[J]. American Psychological Association, 2012, 138(6): 1-81.

[96] Metelli F. The perception of transparency[J]. Scientific American, 1974, 230(4): 90-99.

[97] Metelli F. Stimulation and perception of transparency[J]. Psychological Research, 1985, 47(4): 185-202.

[98] Sayim B, Cavanagh P. The art of transparency[J]. i-Perception, 2011, 2(7): 679-696.

[99] Gabow H N, Myers E W. Finding all spanning trees of directed and undirected graphs[J]. SIAM Journal on Computing, 1978, 7(3): 280-287.

[100] Johnson S G. The nlopt nonlinear-optimization package[M/OL]. 2011[2011-01-01]. http://ab-initio.mit.edu/nlopt.

[101] Nocedal J. Updating quasi-newton matrices with limited storage[J]. Mathematics of Computation, 1980, 35(151): 773-782.

[102] Liu D C, Nocedal J. On the limited memory bfgs method for large scale optimization[J]. Mathematical Programming, 1989, 45(1): 503-528.

[103] Leal A M M, et al. autodiff, a modern, fast and expressive C++ library for automatic differentiation[EB/OL]. 2018[2018-01-01]. https://autodiff.github.io.

[104] Selinger P. Potrace: a polygon-based tracing algorithm[EB/OL]. 2009[2009-07-01]. http://potrace.sourceforge.net.

[105] Bradski G. The OpenCV library.[J]. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 2000, 25(11): 120-123.

[106] Kirillov A, Mintun E, Ravi N, et al. Segment anything[J/OL]. CoRR, 2023, abs/2304.02643 [2023-05-01]. https://arxiv.org/abs/2304.02643.

[107] Wang X, Zhang X, Cao Y, et al. SegGPT: Segmenting everything in context[J/OL]. CoRR, 2023, abs/2304.03284[2023-05-01]. https://arxiv.org/abs/2304.03284.

[108] Shi J, Malik J. Normalized cuts and image segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 888-905.

[109] Felzenszwalb P F, Huttenlocher D P. Efficient graph-based image segmentation[J]. International Journal of Computer Vision, 2004, 59: 167-181.

[110] Perbet F, Maki A. Homogeneous superpixels from random walks.[C]//MVA. 2011: 26-30.

[111] Tang D, Fu H, Cao X. Topology preserved regular superpixel[C]//2012 IEEE International Conference on Multimedia and Expo. IEEE, 2012: 765-768.

[112] Shen J, Du Y, Wang W, et al. Lazy random walks for superpixel segmentation[J]. IEEE Transactions on Image Processing, 2014, 23(4): 1451-1462.

[113] Veksler O, Boykov Y, Mehrani P. Superpixels and supervoxels in an energy optimization framework[C]//Proceedings of the 11th European Conference on Computer Vision on Computer Vision (ECCV). 2010: 211-224.

[114] Achanta R, Shaji A, Smith K, et al. SLIC superpixels compared to state-of-the-art superpixel methods[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(11): 2274-2282.

[115] Fu H, Cao X, Tang D, et al. Regularity preserved superpixels and supervoxels[J]. IEEE Transactions on Multimedia, 2014, 16(4): 1165-1175.

[116] Li Z, Chen J. Superpixel segmentation using linear spectral clustering[C]//2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015: 1356-1363.

[117] Van den Bergh M, Boix X, Roig G, et al. Seeds: Superpixels extracted via energy-driven sampling[J]. International Journal of Computer Vision, 2015, 111: 298-314.

[118] Achanta R, Shaji A, Smith K, et al. SLIC superpixels compared to state-of-the-art superpixel methods[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(11): 2274-2282.

[119] He K, Sun J, Tang X. Guided image filtering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 35(6): 1397-1409.

[120] Maćkiewicz A, Ratajczak W. Principal components analysis (PCA)[J]. Computers & Geosciences, 1993, 19(3): 303-342.

[121] Karamizadeh S, Abdullah S M, Manaf A A, et al. An overview of principal component analysis [J]. Journal of Signal and Information Processing, 2013, 4(3B): 173.

[122] Minka T. Automatic choice of dimensionality for PCA[J]. Advances in Neural Information Processing Systems, 2000, 13(1): 577-583.

[123] Aksoy Y, Oh T H, Paris S, et al. Semantic soft segmentation[J]. ACM Transactions on Graphics (TOG), 2018, 37(4): 1-13.

[124] Chen L C, Papandreou G, Kokkinos I, et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 40(4): 834-848.

[125] Hoffer E, Ailon N. Deep metric learning using triplet network[C]//Similarity-Based Pattern Recognition: Third International Workshop. Springer, 2015: 84-92.

[126] Klein G, Murray D. Parallel tracking and mapping for small AR workspaces[C]//2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. 2007: 225-234.

[127] Mur-Artal R, Tardós J D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras[J]. IEEE Transactions on Robotics, 2017, 33(5): 1255-1262.

[128] Newcombe R A, Lovegrove S J, Davison A J. DTAM: Dense tracking and mapping in real-time [C]//2011 International Conference on Computer Vision (ICCV). 2011: 2320-2327.

[129] Kähler O, Prisacariu V A, Ren C Y, et al. Very high frame rate volumetric integration of depth images on mobile devices[J]. IEEE Transactions on Visualization and Computer Graphics, 2015, 21(11): 1241-1250.

[130] Forster C, Pizzoli M, Scaramuzza D. SVO: Fast semi-direct monocular visual odometry[C]// 2014 IEEE International Conference on Robotics and Automation (ICRA). 2014: 15-22.

[131] Kerl C, Sturm J, Cremers D. Robust odometry estimation for RGB-D cameras[C]//2013 IEEE International Conference on Robotics and Automation (ICRA). 2013: 3748-3754.

[132] Newcombe R A, Fox D, Seitz S M. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time[C]//2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015: 343-352.

[133] Runz M, Buffier M, Agapito L. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects[C]//2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). 2018: 10-20.

[134] Scona R, Jaimez M, Petillot Y R, et al. Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). 2018: 3849-3856.

[135] Barath D, Matas J. Graph-Cut RANSAC[C]//2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018: 6733-6741.

[136] Sutton C, McCallum A, et al. An introduction to conditional random fields[J]. Foundations and Trends® in Machine Learning, 2012, 4(4): 267-373.

[137] Krähenbühl P, Koltun V. Efficient inference in fully connected CRFs with gaussian edge potentials[C]//Proceedings of the 24th International Conference on Neural Information Processing Systems. 2011: 109–117.

[138] Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems [C]//2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2012: 573-580.

[139] Palazzolo E, Behley J, Lottes P, et al. Refusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals[C]//2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2019: 7855-7862.

[140] Wikipedia. Regular skew polyhedron[EB/OL]. 2023[2023-03-01]. https://en.wikipedia.org/wiki/Regular_skew_polyhedron.

[141] Joshi P, Meyer M, DeRose T, et al. Harmonic coordinates for character articulation[J]. ACM Transactions on Graphics (TOG), 2007, 26(3): 71-es.

[142] Lipman Y, Levin D, Cohen-Or D. Green coordinates[J]. ACM Transactions on Graphics (TOG), 2008, 27(3): 1-10.

[143] Zhang J, Deng B, Liu Z, et al. Local barycentric coordinates[J]. ACM Transactions on Graphics (TOG), 2014, 33(6): 1-12.

[144] Ju T, Schaefer S, Warren J. Mean value coordinates for closed triangular meshes[J]. ACM Transactions on Graphics (TOG), 2005, 24(3): 561-566.

[145] Shugrina M, Kar A, Fidler S, et al. Nonlinear color triads for approximation, learning and direct manipulation of color distributions[J]. ACM Transactions on Graphics (TOG), 2020, 39 (4): 1-13.

致 谢

转眼间博士生涯即将结束，回顾过去的四年感慨良多。攻读博士是一段难忘的旅程，虽历经痛苦与孤独但也取得了成长和进步。值此论文付梓之际，我要对长期以来给予我莫大帮助的师长、同学以及亲友们表示最诚挚的谢意。

首先，我要真诚地感谢恩师徐昆副教授的辛苦栽培。徐老师学术水平一流，治学严谨，为人友善，是他一步步带领我走进科研的大门。在整个博士阶段，徐老师对我的影响是无与伦比的。小到平时的组内交流与展示，大到科研项目的立项、实施、论文撰写的各个阶段，他都倾注了大量心血。在他的指导下，我有幸能在图形学领域顶级的期刊和会议上发表论文，让我领略了科研之美，增强了自信心同时提升了科研水平。跟徐老师工作的四年，让我取得了长足的进步，这段经历是我一生的宝贵财富。师恩如海深，在此也祝愿徐老师桃李满天下, 梓楠遍五洲！

其次，感谢图形学实验室的老师和同学。感谢胡事民教授多年来对我的关心和帮助。他牵头在我工作的青海大学计算机系成立可视媒体研究所，并多次前往青海大学开展学术交流活动。此外，他还指导我们申请科研项目，并鼓励我和多名同事到清华大学攻读博士学位。我对胡老师这种无私奉献、助人为乐的精神感到由衷的敬佩。感谢张松海老师，王瑀屏老师，穆太江老师在科研和生活上给予我的帮助。感谢实验室的黄石生学长在早期的科研项目中对于我的详尽指导。感谢实验室的黄家晖、雷凯翔、康谅夫、折栋宇、吴启凌、王怡力、高端、郑少锟、邢建开、安頔等同学的并肩同行，跟这些高水平的同学交流与合作使我受益良多。感谢杨培、任洋甫等一起求学的同事，我们建立了深厚的友谊并一同追逐梦想。

再者，感谢美国乔治梅森大学的 Yotam Gingold 副教授和快手科技的谈建超研究员，感谢他们在两个合作项目中对我的精心指导，他们给出的很多建议对项目的实施起到了至关重要的作用。同时我也从他们身上学到了很多实用的科研方法。感谢英国卡迪夫大学的 Ralph Martin 教授在科研论文写作中提供的帮助。

此外，感谢教育部实施的对口支援计划，感谢清华大学对青海大学二十多年的倾力相助。受益于该计划，使我能在清华大学软件学院和计算机系攻读硕士和博士学位。在此也感谢青海大学计算机系的同事们对我长期的支持和帮助。

最后，我要特别感谢我的家人。感谢父母的养育之恩，感谢他们对我长期的关心和照顾，他们时时刻刻让我感受到家庭的温暖。感谢我的爱人李晓红对我工作、学习、生活上的全力支持，她温柔贤淑、善解人意，将家中事务安排得井井有条，使我能够心无旁骛安心求学。感谢所有亲朋好友的关心与鼓励，理解和包容。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

个人简历、在学期间完成的相关学术成果

个人简历

1989 年 10 月 17 日出生于四川省通江县。

2008 年 9 月考入青海大学计算机技术与应用系计算机科学与技术专业，2012 年 7 月本科毕业并获得工学学士学位。

2012 年 9 月免试进入清华大学软件学院软件工程专业学习，2015 年 7 月硕士毕业并获得工学硕士学位。

2015 年 9 月至今任教于青海大学计算机技术与应用系。

2019 年 9 月考入清华大学计算机科学与技术系攻读博士学位至今。

在学期间完成的相关学术成果

学术论文：

[1] **Du Z J**, Kang L F, Tan J C, et al. Image vectorization and editing via linear gradient layer decomposition. (已经被 ACM SIGGRAPH 2023 (Journal Track) 录用，将发表于 ACM Transactions on Graphics，CCF A 类期刊)

[2] **Du Z J**, Lei K X, Xu K, et al. Video recoloring via spatial-temporal geometric palettes[J]. ACM Transactions on Graphics, 2021, 40(4): 1-16. (CCF A 类期刊)

[3] **Du Z J**, Huang S S, Mu T J, et al. Accurate dynamic SLAM using CRF-based long-term consistency[J]. IEEE Transactions on Visualization and Computer Graphics, 2020, 28(4): 1745-1757. (CCF A 类期刊)

[4] 夏子勋, **杜正君 ***, 刘晓静. 代表性调色板提取及图像重着色. (已被计算机辅助设计与图形学学报录用, CCF A 类中文期刊)

专利：

[1] 杜正君, 夏子勋. 基于调色板的图像重着色方法和系统. 中国, CN202210468112.4. 2022-04-29.

科研项目：

[1] 杜正君, 李阳贵, 等. 图像量化技术在藏毯设计中的应用研究: 青海省自然科学基金青年基金项目, 主持. 2023.01-2025.12.

[2] 杜正君, 崔亚超, 等. 面向智能机器人感知的三维场景重建与分析: 国家自然科学基金项目, 主持. 2019.01-2022.12.

指导教师评语

论文由杜正君同学独立完成。

图像视频编辑是计算机图形学、媒体计算等领域的主要研究方向，相关技术在政治、经济、教育、军事、国防等诸多领域发挥了重要作用。近年来，随着图像视频分享平台的快速发展，很多应用场景都对图像视频编辑的便捷性和高效性提出了更高的要求。该论文围绕面向图像视频编辑的图层分解技术开展研究，旨在通过图层分解实现简单高效的图像视频编辑。论文选题新颖，提出的算法具有重要的理论意义和应用价值。

论文的创新性成果包括：

1. 提出了一种面向图像矢量化的半透明图层分解方法。首先，根据图像分割区域的支持关系构建区域邻接图；然后，通过搜索生成树确定所有可能的图层配置；最后，通过优化的方法确定图层参数。相对于已有方法，该方法在输入给定的情况下实现了全自动的图层分解，此外，通过引入感知规则约束，使得分解的图层具有更好的完整性，能够更好地反映图像本身的形状和层次结构。

2. 提出了一种面向图像内容感知重着色的加性图层分解方法。现有的面向图像重着色的加性图层分解方法难以实现语义级别的物体编辑，无法有效区分颜色相似的不同物体并进行不同的颜色编辑。该方法将图像的颜色特征同高级语义特征结合起来，在高维空间提取输入图像的调色板并将图像分解为一组包含语义信息的加性图层，实现了语义级别、内容感知的图像重着色。

3. 提出了一种面向视频重着色的时变图层分解方法。当前基于图层分解的方法在图像重着色上取得了不错的进展，然而针对视频重着色仍然没有较好的解决方案。为了实现直观高效的视频颜色编辑，该方法将面向图像重着色的加性图层分解算法拓展到视频场景，通过在 RGBT 4 维空间抽取视频的几何调色板，然后将视频分解为一组随时间光滑渐变的图层，实现了直观自然的视频颜色编辑。

论文章节安排合理，写作规范，实验详实，文献调研充分，符合博士学位论文要求。

该论文是一篇优秀的博士学位论文。

答辩委员会决议书

论文研究面向图像视频编辑的图层分解技术，选题具有重要的理论意义和实用价值。论文取得的创新性成果包括：

1. 提出了一种半透明图层分解算法，通过在区域邻接图上遍历搜索所有生成树，并引入感知规则约束有效减小搜索空间，实现了全自动和高质量的图层分解。

2. 提出了一种加性图层分解算法，通过结合图像颜色特征和高级语义特征，并在高维空间进行图像调色板提取和图层分解，实现了内容感知的图像重着色。

3. 提出了一种时变图层分解算法，通过提取时空四维几何调色板，实现了直观、自然的视频重着色。

论文工作表明，杜正君在计算机科学与技术领域具有坚实宽广的基础理论和系统深入的专门知识，独立从事科研工作能力强。论文结构合理、写作规范，研究成果创新性强，是一篇优秀的博士学位论文。

答辩过程表述清楚，回答问题正确。答辩委员会经无记名投票表决，一致同意通过杜正君的博士学位论文答辩，并建议授予杜正君工学博士学位。