Data-driven Approaches for Interactive Appearance Editing

Chuong H. Nguyen

Saarbrücken, Germany

Dissertation zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften der
Naturwissenschaftlich-Technischen Fakultäten der
Universität des Saarlandes

December 2014

Betreuender Hochschullehrer – Supervisor
Prof. Dr. Hans-Peter Seidel

Gutachter – Reviewer
Dr. Tobias Ritschel
Prof. Dr. Hans-Peter Seidel
Prof. Dr. Holly Rushmeier

Dekan – Dean
Prof. Dr. Markus Bläser, Universität des Saarlandes, Saarbrücken, Germany

Kolloquium – Examination

Datum – Date:
22. June. 2015

Vorsitzender – Chair:
Prof. Dr. Philipp Slusallek, Universität des Saarlandes, Saarbrücken, Germany

Prüfer – Examiners:
Dr. Tobias Ritschel, MPI Informatik, Saarbrücken, Germany
Prof. Dr. Hans-Peter Seidel, MPI Informatik, Saarbrücken, Germany
Prof. Dr. Holly Rushmeier, Yale University, USA

Protokoll – Reporter:
Dr. Christian Richardt, MPI Informatik, Saarbrücken, Germany

Abstract

This thesis proposes several techniques for interactive editing of digital content and fast rendering of virtual 3D scenes. Editing of digital content - such as images or 3D scenes - is difficult, requires artistic talent and technical expertise. To alleviate these difficulties, we exploit data-driven approaches that use the easily accessible Internet data (e. g., images, videos, materials) to develop new tools for digital content manipulation. Our proposed techniques allow casual users to achieve high-quality editing by interactively exploring the manipulations without the need to understand the underlying physical models of appearance.

First, the thesis presents a fast algorithm for realistic image synthesis of virtual 3D scenes. This serves as the core framework for a new method that allows artists to fine tune the appearance of a rendered 3D scene. Here, artists directly paint the final appearance and the system automatically solves for the material parameters that best match the desired look. Along this line, an example-based material assignment approach is proposed, where the 3D models of a virtual scene can be "materialized" simply by giving a guidance source (image/video). Next, the thesis proposes shape and color subspaces of an object that are learned from a collection of exemplar images. These subspaces can be used to constrain image manipulations to valid shapes and colors, or provide suggestions for manipulations. Finally, data-driven color manifolds which contain colors of a specific context are proposed. Such color manifolds can be used to improve color picking performance, color stylization, compression or white balancing.

Kurzzusammenfassung

Diese Dissertation stellt Techniken zum interaktiven Editieren von digitalen Inhalten und zum schnellen Rendering von virtuellen 3D Szenen vor. Digitales Editieren - seien es Bilder oder dreidimensionale Szenen - ist kompliziert, benötigt künstlerisches Talent und technische Expertise. Um diese Schwierigkeiten zu relativieren, nutzen wir datengesteuerte Ansätze, die einfach zugängliche Internetdaten, wie Bilder, Videos und Materialeigenschaften, nutzen um neue Werkzeuge zur Manipulation von digitalen Inhalten zu entwickeln. Die von uns vorgestellten Techniken erlauben Gelegenheitsnutzern das Editieren in hoher Qualität, indem Manipulationsmöglichkeiten interaktiv exploriert werden können ohne die zugrundeliegenden physikalischen Modelle der Bildentstehung verstehen zu müssen.

Zunächst stellen wir einen effizienten Algorithmus zur realistischen Bildsynthese von virtuellen 3D Szenen vor. Dieser dient als Kerngerüst einer Methode, die Nutzern die Feinabstimmung des finalen Aussehens einer gerenderten dreidimensionalen Szene erlaubt. Hierbei malt der Künstler direkt das beabsichtigte Aussehen und das System errechnet automatisch die zugrundeliegenden Materialeigenschaften, die den beabsichtigten Eigenschaften am nahesten kommen. Zu diesem Zweck wird ein auf Beispielen basierender Materialzuordnungsansatz vorgestellt, für den das 3D Model einer virtuellen Szene durch das simple Anführen einer Leitquelle (Bild, Video) in Materialien aufgeteilt werden kann. Als Nächstes schlagen wir Form- und Farbunterräume von Objektklassen vor, die aus einer Sammlung von Beispielbildern gelernt werden. Diese Unterräume können genutzt werden um Bildmanipulationen auf valide Formen und Farben einzuschränken oder Manipulationsvorschläge zu liefern. Schließlich werden datenbasierte Farbmannigfaltigkeiten vorgestellt, die Farben eines spezifischen Kontexts enthalten. Diese Mannigfaltigkeiten ermöglichen eine Leistungssteigerung bei Farbauswahl, Farbstilisierung, Komprimierung und Weißabgleich.

Summary

This thesis proposes several techniques for interactive editing of digital content and fast rendering of virtual 3D scenes. Editing of digital content - such as images or 3D scenes - is difficult, requires artistic talent and technical expertise. To alleviate these difficulties, we exploit data-driven approaches that use the easily accessible Internet data (e. g., images, videos, materials) to develop new tools for digital content manipulation. Our proposed techniques allow casual users to achieve high-quality editing by interactively exploring the manipulations without the need to understand the underlying physical models of appearance.

First, the thesis presents a fast algorithm for realistic image synthesis of virtual 3D scenes. This serves as the core framework for a new method that allows artists to fine tune the appearance of a rendered 3D scene. Here, artists directly paint the final appearance and the system automatically solves for the material parameters that best match the desired look. Along this line, an example-based material assignment approach is proposed, where the 3D models of a virtual scene can be "materialized" simply by giving a guidance source (image/video). Next, the thesis proposes shape and color subspaces of an object that are learned from a collection of exemplar images. These subspaces can be used to constrain image manipulations to valid shapes and colors, or provide suggestions for manipulations. Finally, data-driven color manifolds which contain colors of a specific context are proposed. Such color manifolds can be used to improve color picking performance, color stylization, compression or white balancing.

This work starts with an introduction in Chapter 1 that motivates the subjects, list the contributions made and gives an outline of the thesis. Chapter 2 discusses some background and reviews previous work, which are relevant for the thesis. The techniques proposed in this thesis contribute to different building blocks of a common design pipeline, where artists perform appearance editing using intuitive user interfaces and observe the result of either a real-world photograph or synthesis image of a virtual 3D scene. Chapter 3 proposes a fast rendering method that can be used for interactive visualization of different intuitive appearance editing techniques proposed in Chapter 4–6. Furthermore, Chapter 7 proposes a new user interface to improve both editing performance and quality. Chapter 8 concludes the thesis and discusses some directions for future work.

Preconvolved Radiance Caching In computer graphics, digital reproduction of real-world objects is done by means of virtual 3D scenes that contain geometries, materials, lightings and textures. Rendering them can be considered as the simulations of light transport to reproduce real-world appearance. Physically-based rendering, however, is computationally expensive, therefore, improving rendering performance is neccesary to

make it suitable for interactive applications. For realistic rendering of a virtual 3D scene, the incident indirect light over a range of image pixels is often coherent. Two common approaches to exploit this inter-pixel coherence to improve rendering performance are Irradiance Caching and Radiance Caching. Both compute incident indirect light only for a small subset of pixels (the cache), and later interpolate between pixels. Irradiance Caching uses scalar values that can be interpolated efficiently, but cannot account for shading variations caused by normal and reflectance variation between cache items , e. g., fine shading details due to bump mapping will be lost in Irradiance Caching. Radiance Caching maintains directional information, e. g., to allow highlights between cache items, but at the cost of storing and evaluating a Spherical Harmonics (SH) function per pixel. The arithmetic and bandwidth cost for this evaluation is linear in the number of coefficients and can be substantial. Chapter 3 proposes a method to replace it by an efficient per-cache item pre-filtering based on MIP maps — such as previously done for environment maps — leading to a single constant-time lookup per pixel. Additionally, per-cache item geometry statistics stored in distance-MIP maps are used to improve the quality of each pixel's lookup. Cache items can be computed independently and in parallel on a Graphics Processing Unit (GPU). The proposed technique is an order of magnitude faster than Radiance Caching with Phong BRDFs and can be combined with Monte Carlo-raytracing, Point-based Global Illumination or Instant Radiosity.

Surface Light Field Manipulation Chapter 4 addresses the challenge of intuitive appearance editing in scenes with complex geometric layout and complex, spatially-varying indirect lighting. In contrast to previous work, that aimed to edit surface reflectance, the new approach allows a user to freely manipulate the surface light field. It then finds the best surface reflectance that "explains" the surface light field manipulation. Instead of classic L_2 fitting of reflectance to a combination of incoming and exitant illumination, a sparse L_0 change of shading parameters is inferred. Consequently, no "diffuse" or "glossiness" brushes or any such understanding of the underlying reflectance parametrization from the users is required. Instead, the system infers reflectance changes from scribbles made by a single simple color brush tool alone: Drawing a highlight will increase Phong specular; blurring a mirror reflection will decrease glossiness; etc. A sparse-solver framework operating on a novel point-based, pre-convolved lighting representation proposed in Chapter 3 in combination with screen-space edit upsampling allows to perform editing interactively on a GPU.

3D Material Style Transfer Not all 3D scenes come with assigned materials, e. g., some scenes downloaded directly from the Internet were simply crafted without materials. When assigned materials are not available or not appropriate, a manual assignment is tedious, especially with complex scenes that contain a large number of objects. Chapter 5 proposes a technique to transfer the material style or mood from a guide source such as an image or video onto a target 3D scene. It formulates the problem as a combinatorial optimization of assigning discrete materials extracted from the guide source to discrete objects in the target 3D scene. The assignment is optimized to fulfill multiple goals: overall image mood based on several image statistics; spatial material organization and grouping as well as geometric similarity between objects that were assigned to similar materials. To be able to use common uncalibrated images and videos with unknown geometry and lighting as

guides, a material estimation derives plausible reflectance, i. e., diffuse color, specularity, glossiness, and texture. Finally, results produced by the technique are compared to manual material assignments in a perceptual study.

Shape and Color Subspaces While the option to change shape and color of an image into any possible other shape or color sounds like a good idea at first, in practice too many possible options actually decrease the human ability to make the right decision. Therefore, the right balance between generality and reduction of choices has to be found. Chapter 6 proposes a system to restrict the manipulation of shape and color in an image to a valid subspace which we learn from a collection of exemplar images. To this end, we automatically align a collection of images and learn a subspace model of shape and color using principal components. As finding perfect image correspondences for general images is not feasible, we build an approximate partial alignment and improve bad alignments leveraging other, more successful alignments. Our system allows the user to change color and shape in real-time and the result is "projected" onto the subspace of meaningful changes. The change in color and shape can either be locked or performed independently. Additional applications include suggestion of alternative shapes or color.

Data-driven Color Manifolds Color selection is required in many computer graphics applications, but can be tedious, as 1D or 2D user interfaces are employed to navigate in a 3D color space. Until now the problem was considered a question of designing general color spaces with meaningful, e. g., perceptual, parameters. Chapter 7 shows how color selection usability can be improved by applying 1D or 2D color manifolds which predict the most likely change of color in a specific context. A typical use case is manipulating the color of a banana: instead of presenting a 2D+1D RGB, CIE Lab or HSV widget, a simple 1D slider that captures the most likely change for this context is presented. Technically, for each context, a lower-dimensional manifold with varying density from labeled Internet examples is extracted. Finally, the increase in task performance of color selection is validated by several user studies.

Zusammenfassung

Diese Dissertation stellt Techniken zum interaktiven Editieren von digitalen Inhalten und zum schnellen Rendering von virtuellen 3D Szenen vor. Digitales Editieren - seien es Bilder oder dreidimensionale Szenen - ist kompliziert, benötigt künstlerisches Talent und technische Expertise. Um diese Schwierigkeiten zu relativieren, nutzen wir datengesteuerte Ansätze, die einfach zugängliche Internetdaten, wie Bilder, Videos und Materialeigenschaften, nutzen um neue Werkzeuge zur Manipulation von digitalen Inhalten zu entwickeln. Die von uns vorgestellten Techniken erlauben Gelegenheitsnutzern das Editieren in hoher Qualität, indem Manipulationsmöglichkeiten interaktiv exploriert werden können ohne die zugrundeliegenden physikalischen Modelle der Bildentstehung verstehen zu müssen.

Zunächst stellen wir einen effizienten Algorithmus zur realistischen Bildsynthese von virtuellen 3D Szenen vor. Dieser dient als Kerngerüst einer Methode, die Nutzern die Feinabstimmung des finalen Aussehens einer gerenderten dreidimensionalen Szene erlaubt. Hierbei malt der Künstler direkt das beabsichtigte Aussehen und das System errechnet automatisch die zugrundeliegenden Materialeigenschaften, die den beabsichtigten Eigenschaften am nahesten kommen. Zu diesem Zweck wird ein auf Beispielen basierender Materialzuordnungsansatz vorgestellt, für den das 3D Model einer virtuellen Szene durch das simple Anführen einer Leitquelle (Bild, Video) in Materialien aufgeteilt werden kann. Als Nächstes schlagen wir Form- und Farbunterräume von Objektklassen vor, die aus einer Sammlung von Beispielbildern gelernt werden. Diese Unterräume können genutzt werden um Bildmanipulationen auf valide Formen und Farben einzuschränken oder Manipulationsvorschläge zu liefern. Schließlich werden datenbasierte Farbmannigfaltigkeiten vorgestellt, die Farben eines spezifischen Kontexts enthalten. Diese Mannigfaltigkeiten ermöglichen eine Leistungssteigerung bei Farbauswahl, Farbstilisierung, Komprimierung und Weißabgleich.

Diese Arbeit beginnt mit einer Einführung in Kapitel 1, die die behandelte Thematik motiviert, die Beiträge auflistet und eine Gliederung der Arbeit aufführt. In Kapitel 2 wird der Hintergrund der Arbeit diskutiert und relevante Arbeiten rezensiert.

Die in dieser Arbeit vorgestellten Techniken beschreiben verschiedenen Basiskomponenten einer generischen Designpipeline, in der Künstler Erscheinungsänderungen mit Hilfe von intuitiven Nutzerschnittstellen vornehmen und das Ergebnis in Form einer Fotografie oder eines synthetischen Bildes einer virtuellen 3D Szene dargestellt wird. Kapitel 3 stellt eine effiziente Renderingmethode vor, die die interaktiven Visualisierung verschiedener intuitiver Erscheinungsänderungen, vorgestellt in Kapiteln 4–6, ermöglicht. Weiterhin wird in Kapitel 7 eine neuartige Nutzerschnittstelle vorgestellt, die die Qualität der Bildmanipulation verbessert. Kapitel 8 beschließt die Arbeit und bespricht Möglichkeiten für zukünftige Erweiterungen.

Preconvolved Radiance Caching Die Computergrafik behandelt die digitale Reproduktion von Objekten der realen Welt durch virtuelle 3D Szenen, die Geometrie, Materialien, Beleuchtung und Texturen enthalten. Rendering kann als Simulation des Lichttransports zum Reproduzieren der Erscheinung der realen Welt angesehen werden. Allerdings ist physikalisch-basiertes Rendering rechenintensiv und zum Ermöglichen interaktiver Anwendungen ist daher eine Steigerung der Renderingperformance notwendig. Das einfallende indirekte Licht von benachbarten Bildpixeln einer realistischen Synthese virtueller dreidimensionaler Szenen ist häufig kohärent. Zwei bekannte Ansätze, um diese Kohärenz zwischen den Pixeln zur Verbesserung der Renderingleistung zu nutzen, sind Irradiance Caching und Radiance Caching. Beide berechnen das einfallende indirekte Licht nur für eine kleine Untermenge an Pixeln (dem Cache) und interpolieren anschließend zwischen diesen. Irradiance Caching nutzt skalare Cacheeinträge, die effizient interpoliert werden können, kann aber zwischen Cacheeinträgen keine durch Normalen- und Reflexionsfaktoränderungen erzeugten Schattierungsvariationen darstellen. So gehen beispielsweise feine Schattierungsdetails aus der Verwendung von Bumpmapping beim Gebrauch von Irradiance Caching verloren. Radiance Caching erhält gerichtete Informationen, z.B. um Glanzpunkte zwischen Cacheeinträgen zu ermöglichen, erfordert allerdings die Speicherung und Auswertung einer Spherical Harmonics-Funktion pro Pixel. Die arithmetischen Kosten und die Bandbreitenkosten dieser Evaluation sind linear in der Anzahl an Koeffizienten und können erheblich sein. Kapitel 3 stellt eine Methode zum Ersetzen dieser Funktion durch eine Vorfilterung pro Cacheeintrag unter Zuhilfenahme von MIP-Maps vor. Dies erlaubt die Berechnung in konstanter Zeit mit einem einmaligen Lookup pro Pixel. Zudem werden pro Cacheeintrag in Distanz-MIP-Maps gespeicherte Geometriestatistiken genutzt, um die Qualität der Lookups pro Pixel zu verbessern. Die Cacheeinträge können unabhängig voneinander und parallellaufend auf einer Graphics Processing Unit (GPU) berechnet werden. Die vorgestellte Technik ist um eine Größenordnung schneller als Phong BRDFs und kann mit Monte Carlo Raytracing, Point-based Global Illumination oder Instant Radiosity kombiniert werden.

Surface Light Field Manipulation Kapitel 4 bespricht Möglichkeiten intuitiver Erscheinungsänderungen in Szenen mit komplexem geometrischem Layout und komplexer, räumlicher-variierender indirekter Beleuchtung. Im Gegensatz zu bisherigen Arbeiten, in denen der Oberflächenreflexionfaktor editiert wird, erlaubt der neue Ansatz dem Nutzer das Oberflächenlichtfeld frei zu manipulieren. Die Technik errechnet anschließend den Oberflächenreflexionfaktor, der das Oberflächenlichtfeld am besten erklärt. Anstatt einer klassischen L_2-Reflexionsfaktorangleichung an eine Kombination von ein- und ausgehender Beleuchtung, wird eine dünn besetzte L_0-Änderung der Beleuchtungsparameter erstellt. Folglich erfordert das System vom Nutzer kein tieferes Verständnis von Werkzeugen zum Malen von diffusen oder glänzenden Bildbereichen oder vergleichbaren Konzepten des zugrundeliegenden Reflexionsfaktors. Änderungen des Reflexionsfaktors werden stattdessen anhand von Annotationen mit einem einzelnen simplen Farbwerkzeug abgeleitet: Beispielsweise erhöht das Malen eines Glanzpunktes den Phong Spiegelfaktor, das Verwischen einer Spiegelreflexion verringert den Glanz, usw. Die Technik erlaubt interaktive Änderungen durch Verwendung einer GPU, auf der ein dünn besetztes Gleichungssystem gelöst wird, das auf einer neuartige, punktbasierte, vorgefaltete Beleuchtungsrepräsentation basiert(Kapitel 3).

3D Material Style Transfer Nicht alle dreidimensionalen Szenen enthalten vordefinierte Materialeigenschaften. So besitzen einige aus dem Internet geladene Szenen beispielsweise keine oder nur unpassende Materialen. Ist dies der Fall, gestaltet sich eine manuelle Zuordnung, besonders bei komplexen Szenen mit vielen unterschiedlichen Objekten, meist mühsam. Kapitel 5 stellt hierfür eine Technik zum Transfer von Materialstil und -stimmung anhand einer Bezugsquelle (Bild/Video) auf eine dreidimensionale Zielszene vor. Die Technik formuliert das Problem als eine kombinatorische Optimierung der Zuordnung von Materialien einer Bezugsquelle zu Objekten einer 3D Zielszene. Diese Zuweisung wird für folgende Zielsetzungen simultan optimiert: Ähnlichkeit der gesamten Bildstimmung basierend auf unterschiedlichen Bildstatistiken, räumliche Materialorganisation und -gruppierung sowie geometrische Ähnlichkeit zwischen Objekten, denen ähnliche Materialien zugeordnet wurden. Um gewöhnliche unkalibrierte Bilder und Videos mit unbekannter Geometrie und Beleuchtung als Bezugsquelle zu nutzen, werden die Materialparameter wie Reflexionsfaktor, Reflexionsgrad, Glanz und Textur perzeptuell plausibel geschätzt. Schließlich vergleicht eine perzeptuelle Studie die Ergebnisse der vorgestellten Technik mit Ergebnissen einer manuellen Materialzuweisung.

Shape and Color Subspaces Obwohl es auf den ersten Blick erstrebenswert scheint, Form und Farbe eines Bildes in jede mögliche andere Form und Farbe zu verändern, verringern in der Praxis zu viele Möglichkeiten die menschliche Entscheidungsfähigkeit. Folglich muss die richtige Balance zwischen Generalität und Einschränkung der Auswahl gefunden werden. Kapitel 6 stellt ein System zum Einschränken der Manipulationsmöglichkeiten von Form und Farbe eines Bildes auf einen validen Unterraum vor, der aus einer Sammlung an Beispielbildern gelernt wird. Zu diesem Zweck richten wir eine Sammlung von Bildern automatisch aneinander aus und lernen einen Unterraum aus Form und Farbe unter Verwendung der Hauptkomponentenanalyse. Da das Finden perfekter Bildkorrespondenzen für allgemeine Bilder nicht umsetzbar ist, erstellen wir eine approximierende partielle Ausrichtung und verbessern unbrauchbare Ausrichtungen durch Zuhilfenahme von anderen, erfolgreicheren Ausrichtungen. Unser System erlaubt dem Nutzer die Form und Farbe in Echtzeit zu verändern und das Ergebnis wird auf den Unterraum der plausiblen Änderungen projiziert. Änderungen in Form und Farbe können gekoppelt oder unabhängig voneinander durchgeführt werden. Weitere Anwendungsbeispiele sind automatisierte Vorschläge von alternativen Formen oder Farben.

Data-driven Color Manifolds Eine Farbauswahl wird in vielen Computergrafikapplikationen benötigt, kann aber mühsam sein, da ein- oder zweidimensionale Nutzerschnittstellen verwendet werden um in einem dreidimensionalen Farbraum zu navigieren. Bisher wurde dieses Problem als Frage des Entwurfes eines allgemeinen Farbraumes mit sinnvollen, perzeptuellen Parametern angesehen. Kapitel 7 zeigt, wie die Nutzerfreundlichkeit einer Farbauswahl verbessert werden kann, indem ein- oder zweidimensionale Farbmannigfaltigkeiten verwendet werden, um die wahrscheinlichsten Farbänderungen in einem spezifischen Kontext vorherzusagen. Ein typischer Anwendungsfall ist die Manipulation der Farbe eines Gegenstandes, wie z.B. einer Banane: Anstatt ein 2D+1D RGB, CIE Lab oder HSV Widget anzuzeigen, wird ein einfacher 1D Slider angezeigt, der die wahrscheinlichsten Farbänderungen für diesen Kontext erfasst. Aus technischer Sicht wird für jeden Kontext eine

niedrig-dimensionale Mannigfaltigkeit mit variierender Dichte aus annotierten Internetbeispielen extrahiert. Schließlich validiert eine Nutzerstudie die Verbesserung der Farbauswahl.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Contributions . . . 4
- 1.3 Outline . . . 5

2 Background and Previous Work **7**
- 2.1 Color . . . 7
  - 2.1.1 Terminology . . . 7
  - 2.1.2 Color Model . . . 8
  - 2.1.3 Color Space . . . 9
- 2.2 Material . . . 10
  - 2.2.1 Material Model . . . 10
  - 2.2.2 Material Perception . . . 11
- 2.3 Rendering . . . 11
  - 2.3.1 Rendering Equation . . . 11
  - 2.3.2 Surface Light Fields (SLFs) . . . 12
  - 2.3.3 Approximate Global Illumination . . . 13
- 2.4 Appearance Editing . . . 16
  - 2.4.1 Color Editing . . . 17
  - 2.4.2 Light, Shadow and Material Editing . . . 18
  - 2.4.3 Edit Propagation . . . 19
  - 2.4.4 Subspaces-aware Editing . . . 19
  - 2.4.5 Style Transfer . . . 20
  - 2.4.6 Appearance Manifolds . . . 22
- 2.5 Statistical Hypothesis Testing . . . 22
  - 2.5.1 Hypothesis . . . 23
  - 2.5.2 The p-value . . . 23
  - 2.5.3 Statistical Test . . . 24
  - 2.5.4 Post-hoc Test . . . 25
  - 2.5.5 Effect Size . . . 26

3 Preconvolved Radiance Caching **27**
- 3.1 Introduction . . . 28
- 3.2 Our Approach . . . 28
  - 3.2.1 Pre-convolution . . . 28
  - 3.2.2 Per-pixel Computation . . . 30

3.2.3 Recursive Lookups . . . 31
3.2.4 Implementation . . . 31
3.3 Results . . . 32
3.4 Discussion . . . 32

4 Surface Light Field Manipulation in 3D Scenes 39
4.1 Introduction . . . 39
4.2 Problem Statement . . . 41
4.3 Surface Light Field Manipulations . . . 41
4.3.1 Tools . . . 41
4.3.2 Direct and Indirect Mode . . . 43
4.3.3 Edit Propagation . . . 44
4.4 Discretization . . . 45
4.4.1 Discrete Domain . . . 45
4.4.2 Discrete Operators . . . 46
4.4.3 Discrete Minimization . . . 46
4.5 GPU Implementation . . . 49
4.5.1 Pre-computed Visibility (G) . . . 50
4.5.2 Pre-convolved Radiance (K) . . . 51
4.5.3 Solver . . . 51
4.5.4 Rendering . . . 51
4.5.5 Upsampling . . . 51
4.6 Results . . . 53
4.7 Dicussion . . . 58

5 3D Material Style Transfer 61
5.1 Introduction . . . 61
5.2 Our Approach . . . 62
5.2.1 Definitions . . . 63
5.2.2 Material Extraction . . . 63
5.2.3 Material Assignment . . . 66
5.2.4 Optimization . . . 67
5.2.5 Implementation Details . . . 68
5.3 Results . . . 69
5.4 Discussion . . . 69

6 Shape and Color Subspaces 75
6.1 Introduction . . . 75
6.2 Our Approach . . . 76
6.2.1 Alignment . . . 79
6.2.2 Subspace Construction . . . 85
6.3 Applications and Results . . . 87
6.3.1 Shape and Color Manipulation . . . 87
6.3.2 Shape and Color Suggestions . . . 93
6.3.3 Manipulation of Complex Images . . . 94
6.4 Limitations . . . 95

7 Data-driven Color Manifolds **99**
7.1 Introduction . . . 99
7.2 Our Approach . . . 100
7.2.1 Acquisition . . . 101
7.2.2 Density Estimation . . . 103
7.2.3 Dimensionality Reduction . . . 104
7.3 Algorithm Evaluations . . . 106
7.3.1 Algorithm Comparison . . . 106
7.3.2 Algorithm Analysis . . . 106
7.3.3 User Study . . . 113
7.4 Results . . . 118
7.4.1 Manifolds . . . 118
7.4.2 Applications . . . 121
7.5 Discussion and Limitations . . . 125

8 Conclusion **129**
8.1 Closing Remarks . . . 129
8.2 Future Works . . . 131
8.2.1 Individuals . . . 131
8.2.2 Combinations . . . 132
8.2.3 General Outlook . . . 133
8.3 Message . . . 134

1 Introduction

This thesis proposes several new techniques for interactive appearance editing of images or three-dimensional virtual scenes. In this first chapter, we motivate our research (Section 1.1), present our main contributions (Section 1.2) and outline the whole thesis (Section 1.3).

1.1 Background

Digital appearance models are necessary to simulate real-world objects, ranging from materials (e. g., human skin, trees, the sky), digital shapes of objects (e. g., 3D models of building, trees, human) to digital captures of the real-world scenes (e. g., photographs, digital paintings). With the advancement of the Internet technology, in recent years, more and more content is created and shared everyday, e. g., images on photo-sharing websites (e. g., Flickr) and online social networks (e. g., Facebook), or 3D models on online open-source repositories (e. g., 3D Warehouse). The increase in creative content demands for intuitive editing tools for both artists and casual users. Digital content editing is difficult as it requires a lot of manual work, experience, techical expertise and artistic talent. As the editing process requires a lot of efforts, interactivity of editing tools is important during the trial-and-error process. Still, it is difficult for casual users without training to produce high-quality results. One possible solution to alleviate such difficulty is to use knowledge from existing data to assist the editing process, as in data-driven approaches.

Appearance Modeling Modeling appearance dates back to pre-historic times when humans reproduced the appearance of the real world by paintings. Over time, artists developed techniques seeking for visual impact through the use of colors in their paintings. Originating in the Renaissance era, "chiaroscuro" is a technique that mimics the three-dimensional volume by shading objects using light and shadow effects. Increased understanding of colors at the time also allowed artists to create very complex paintings via realistic depiction of real-world material appearance, e. g., by mixing six color palettes: red, yellow, blue, green, black and white, combining with chiaroscuro and other techniques, Leonardo da Vinci painted the Mona Lisa, which is considered as a masterwork of Renaissance painting

[Feisner and Reed 2013]. Even though artists' understanding of light and color at the time might not align well with modern study, it set an important keystone in the history of real-world appearance modeling.

Digital modeling of material appearance aims for realistic synthesis of computer generated images, either by empirical models inspired by physical observation [Phong 1975; Blinn 1977] or by complex theoretical models [Cook and Torrance 1982]. These models, either following the basic laws of physics or not, define a set of mathematical functions that can be controlled by a set of parameters to simulate the appearance of materials. Recent research accounts for physical correctness of light transport: The rendering equation [Kajiya 1986] which is based on the principles of geometric optics describes physical behavior of light in a vacuum filled with solid objects. Radiative transfer theory [Chandrasekhar 1960] describes the spatial variation of radiance due to emission, in-scattering, absorption and out-scattering in participating media.

Appearance models are used in engineering as a reliable source for predictive appearance. RGB color images, having three channels: red, green, and blue that follow the color receptors in the human eyes, are used as a standard in computer displays. Light and materials models in virtual settings allow full three-dimensional simulations of real-world appearance and can be used to assist designers, such as in an opera scene [Dorsey, Sillion and Greenberg 1991]. In general, appearance models set up the fundamental concepts for digital content editing as to find the right set of parameters that achieves a specific target appearance.

Appearance Editing Appearance editing of digital content contains a wide variety of tasks, ranging from modifying the colors of a photograph [Reinhard et al. 2001; Lischinski et al. 2006; Pellacini et. al. 2007; An and Pellacini 2008], editing materials of objects in images [Khan et al. 2006] or virtual 3D scenes [Ben-Artzi, Overbeck and Ramamoorthi 2006] to performing image [Schaefer, McPhail and Warren 2006] or geometry deformation [Müller et al. 2005].

There are commercial software packages that assist the content editing process such as Adobe Photoshop [Adobe Systems Incorporated 2014b] for image editing; AutoDesk Maya [Autodesk Inc 2014], SketchUp [Trimble Navigation 2014] or Blender [Blender Foundation 2014] for virtual 3D scenes, etc, yet, creating and editing of creative content are still very challenging processes. Trained artists or casual users need to get inspiration from real-world experiences such as cultural background, nature, historical events, etc., during the editing process. One example case is to edit the colors of a bedroom image to make it feel calm and serene. The psychological properties of colors and their mutual impact require knowledge of interior design that is not easy to grasp for novice users.

While it is not easy for casual users to understand the underlying algorithms that modify digital content appearance, existing programs provide intuitive interfaces to perform the edits by tweaking the set of parameters. Understanding the set of parameters requires experience and certain knowledge about the software. To alleviate this difficulty, perceptual-based parameter models were proposed, such as for glossy materials [Pellacini et. al. 2000] or artist-friendly hair shading model [Sadeghi et al. 2010]. To further complicate the process, the parameters lie in a high-dimensional space and their mutual impact on each other is usually not intuitive. Nevertheless, users demand for intuitive editing techniques.

Interactivity If the computation is efficient enough to provide immediate feedback to a user that adjusts the edits, interactivity is achieved. As content editing requires a lot of trial and error with different parameter settings, digital content editors normally utilize interactive manipulation and editing packages to assist their work flow. Interactivity allows users to converge to their goals faster, to explore the design space and to discover new effects. Interactive content editing is an active research area where new tools for different editing tasks are being proposed over the years, e. g., relighting [Pellacini et. al. 2005], tone adjustment [Lischinski et al. 2006] or shape deformation [Schaefer, McPhail and Warren 2006]. While some techniques require pre-computation [Sloan et. al. 2002; Ng, Ramamoorthi and Hanrahan 2003; Ben-Artzi, Overbeck and Ramamoorthi 2006], interactivity during editing is crucial as immediate visual feedback is required to confine the effect of manipulations [Kerr and Pellacini 2010].

Data-driven Approaches Data-driven approaches model an activity by using data rather than by intuition or personal experience. The major advantage of data-driven approaches is that the modeling process is effectively guided by exploiting the mutual relationships inside the data. On the other hand, data have to be available in order to extract a model.

Data-driven approaches have been used since the dawn of modern science. In the 18th century, medical statistics have dealt with applications of statistics to medicine and the health sciences. During the 1840s, statistician William Farr plotted cycles of temperature and cholera deaths, believing that the illness was spread by "miasma" or bad air. While it was actually spread by water-borne bacteria, Farr set up the first national system for collecting statistics and pushed for a more data-driven approach to public health. The weather patterns of the past from log books can be used to test the climate models [Wilkinson et al. 2011].

In computer vision, early work on data-driven approaches have been used in the construction of a space of human faces, called Eigenfaces [Turk and Pentland 1991], which is then used for recognition tasks. Blanz and Vetter [1999] later extended the idea by constructing a morphable model for human faces, derived from a dataset of prototypical 3D scans of faces. In computer graphics, data-driven approaches have been used to tackle different interesting problems from cloth simulation [Wang, O'Brien and Ramamoorthi 2011; Miguel et al. 2012], geometry modeling [Funkhouser et al. 2004], automatic generation of realistic indoor scenes [Merrell, Schkufza and Koltun 2010; Yu et al. 2011] to shape manipulation [Zhou et al. 2010] or appearance transfer [Reinhard et al. 2001; Wang et. al. 2010]. These kind of works are particularly desired for subjective, artistic creation as they can enhance the ease of these tasks while still providing users the freedom to control the creative process.

Preparing the data is a crucial step for data-driven approaches. Depending on the applications, data can be acquired in simple form such as a single photograph [Reinhard et al. 2001], specific guidelines [Merrell et al. 2011] or by specialized measurements [Wang, O'Brien and Ramamoorthi 2011; Blanz and Vetter 1999; Wang et. al. 2011]. In recent years, as the amount of Internet data increases and becomes easier to access, they become an appealing source for data-driven approaches. However, as these data might come from different unsecured sources, their reliability is limited and either fully or semi-supervised post processing is required to refine them.

Conclusion These observations suggest several important properties while developing new appearance editing techniques that this thesis pursues:

- *Interactivity:* Interactivity is crucial for real-time feedback in a design session.
- *Intuitiveness:* Intuitiveness can be made possible by adopting interfaces that require less knowledge of the underlying physical appearance models from the users.
- *Practicability:* In order to increase the practicability of our data-driven approaches, Internet data (such as images, videos or materials) will be used in the scope of this thesis.

1.2 Contributions

This thesis addresses important observations outlined above and makes five contributions, based on the work published in [Scherzer et al. 2012; Nguyen et. al. 2012; Nguyen et. al. 2013; Nguyen et. al. 2015b; Nguyen et. al. 2015].

The main contributions of Chapter 3 (based on [Scherzer et al. 2012]) are

- A novel, scalable GPU-based preconvolved radiance caching technique to efficiently gather incident radiance in large and dynamic scenes.
- A shading scheme that re-uses the nearby pre-convolved radiance from a sparse set of caches to shade all pixels

To this end, a new point-based, fast global illumination algorithm is proposed. This framework allows interactivity in a design session and has served as the core of a new intuitive appearance editing technique proposed in Chapter 4 (based on [Nguyen et. al. 2013]), of which the main contributions are:

- A new user interface to manipulate surface light fields
- An approach to infer a sparse changes of reflectance from the manipulated surface light fields
- A pre-convolved, point-based representation of a family of potential surface light fields, that can be used for efficient manipulation, optimization and rendering

Compared to previous appearance editing approaches, the novel perspective of this work is to permit direct manipulation of the target appearance. The system does not expose the shading model and its parameters to the user and uses scribbles to infer the changes in reflectance.

The main contributions of Chapter 5 (based on [Nguyen et. al. 2012]) are:

- A heuristic algorithm to automatically extract materials from an image/video
- An optimization framework to optimize material appearance in a 3D scene

The proposed system advances as a new tool for automatic material assignment using casual exemplars, e. g., Internet images.

The main contributions of Chapter 6 (based on [Nguyen et. al. 2015b]) are:

- Efficient partial alignment of images in casual image collections with varying appearance
- Completion of partial alignment to a global alignment for all images using an alignment graph

- A novel interactive user interface for shape and color subspace manipulation that preserves detail

Built from casual image collections (such as images from the Internet), our shape and color subspaces encode a valid space for an object class that can be used as a guidance for interactive manipulations and suggestions.

Orthogonal to new methods for intuitive content editing, new user interfaces can greatly improve content editing tasks, one such instance is the color picker which is one of the most common interfaces used for image and video editing. The main contributions of Chapter 7 (based on [Nguyen et. al. 2015]) are:

- A class-specific color manifold
- An analysis of manifold construction using different non-linear and linear dimensionality reduction methods
- User studies that confirm the usefulness of the proposed manifolds

Data-driven color manifolds are constructed from easily accessible data, e. g., Internet images, thus, highly practical, and can be used as an alternative to classic color pickers in order to improve performance and quality of color editing tasks.

1.3 Outline

This thesis is structured as follows. After this introduction, we discuss some background and review previous work in Chapter 2. From Chapter 3 to Chapter 7, five novel techniques are presented in detail. More specifically, Chapter 3 proposes a new interactive global illumination rendering algorithm that is further used to develop a new material editing scheme based on manipulation of surface light fields in Chapter 4. Next, we propose a system to automatically materialize a virtual three-dimensional scene in Chapter 5. In Chapter 6, shape and color subspace for an object class are proposed to improve image manipulations. Finally, to improve color editing tasks, a data-driven approach to extract color manifolds from a specific context is proposed in Chapter 7. The thesis is completed by a conclusion in Chapter 8 which also contains a discussion of future work.

2

Background and Previous Work

In this chapter, we recall some background and previous work in color and material, global illumination rendering, appearance editing, as well as statistical hypothesis testing. First, we review some background on color in Section 2.1 and material models in Section 2.2. Our discussion of global illumination rendering will start in Section 2.3. Next, we review some appearance editing techniques that are most related to our work in Section 2.4. Finally, we briefly discuss some background on statistical hypothesis testing in Section 2.5.

2.1 Color

Perception of colors is a subjective process where the human visual system responds to the stimuli that are produced when incoming light reacts with three types of cone photo receptors in the eye: L, M and S cones. As the physical light consists of a continuous spectrum of wavelengths, these names refer to the long-wavelength (L), middle-wavelength (M), and short-wavelength (S) sensitive cones, respectively. Before discussions about color models and color spaces, we first review some color terminology.

2.1.1 Terminology

Here we describe some common terminology used in color appearance modeling, as defined in [Fairchild 2005], and will be used in this thesis. Detailed exemplars are given in Figure 2.1.

Hue is the degree to which an area appears to be similar to or different from one of the perceived colors: red, yellow, green, and blue, (the unique hues) or to a combination of two of them.

Brightness (luminance) is an attribute of a visual sensation according to which an area appears to emit more or less light.

Lightness is the brightness of an area judged relative to the brightness of a similarly illu-

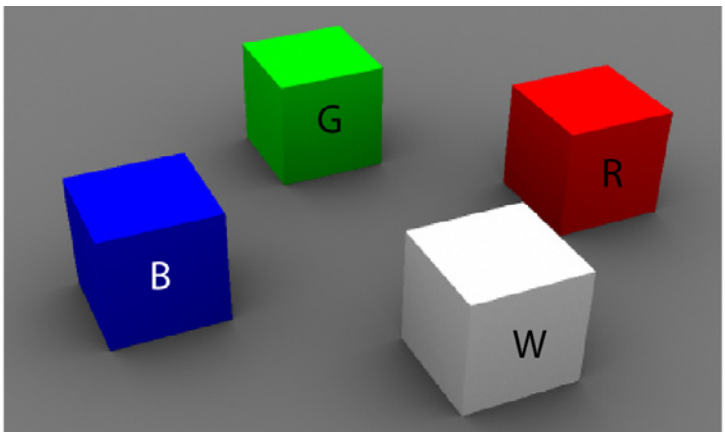

Figure 2.1: A scene consists of four boxes illuminated by an area light source is used to explain different color terminology. The three cubes B, G, and R are of three different *hues*: blue, green, and red. The W cube is white and thus achromatic, possessing no hue. For every cube, each face is illuminated differently and has different *brightness*. However, all visible faces of each cube have the same *lightness* as their brightness relative to the brightness of a similar illuminated white object are identical (Equation 2.1). The faces of the cubes with stronger illumination exhibit greater *colorfulness*, but the *chroma* is roughly constant within each cube (Equation 2.2). Finally, for each cube, the *saturations* of all faces are approximately constant (Equation 2.3).

minated area that appears to be white or highly transmitting.

$$Lightness = \frac{Brightness}{Brightness(White)} \tag{2.1}$$

Colorfulness is an attribute of a visual sensation according to which the perceived color of an area appears to be more or less chromatic.

Chroma is the colorfulness of an area judged as a proportion of the brightness of a similarly illuminated area that appears white or highly transmitting.

$$Chroma = \frac{Colorfulness}{Brightness(White)} \tag{2.2}$$

Saturation is the colorfulness of a color in proportion to its brightness.

$$Saturation = \frac{Colorfulness}{Brightness} = \frac{Chroma}{Lightness} \tag{2.3}$$

2.1.2 Color Model

A color model is a mathematical model which describes colors as tuples of numbers, typically as 3 (e. g., RGB, HSV, LAB) or 4 values (e. g., CMYK). Color model can be derived based on the physics of light, color perception of the eyes or the color reproduction by inks. Here we will discuss color models (Figure 2.2) that are normally used for color pickers and image editing software (See Section 2.4.1).

RGB is a color model that uses the three primary (**r**ed, **g**reen, **b**lue) additive colors and their mixtures to compose all other colors. The mixture of all three colors produces white. The **c**yan, **m**agenta, **y**ellow and **k**ey (black) inks absorb colored light. CMYK is a substractive color model, used in most commercial color printing (books, magazines, etc.). In the CMYK model, white is the natural color of the paper or background in the absence of inks while black is the full combination of colored inks. HSV describes colors in term of **h**ue, **s**aturation and **v**alue (brightness). HSV uses the basic color concepts as its components and is quite similar to the way human perceive colors. The CIEXYZ color model, created by the International Commission on Illumination in 1931 [CIE 1931], is a mapping system that uses tristimulus values to reproduce any color that a human eye can perceive. The CIEXYZ model takes into account the chromatic response of different types of cones (in the retina of the eyes) to different color and light. It is widely considered as the most accurate color model. CIELAB is a perceptual-based color model designed to approximate human vision. CIELAB use three components: **l**ightness, and two opposing color channels red-green (a) and yellow-blue (b) to represent the theoretical range of human vision.

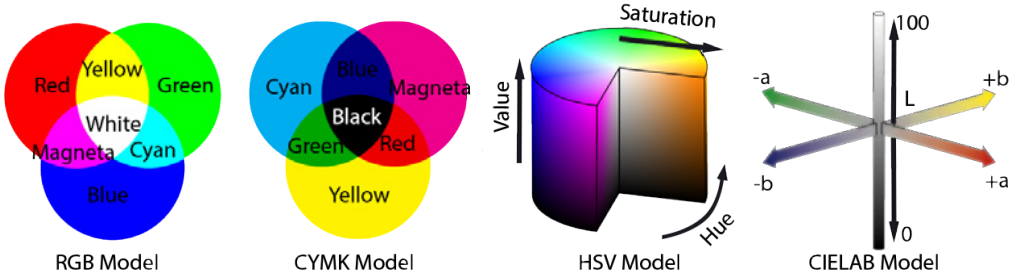

Figure 2.2: RGB *(1st col.)*, CYMK *(2nd col.)*, HSV *(3rd col.)* and CIELAB *(4th col.)* color model.

2.1.3 Color Space

A color space is a specific implementation of a color model by adding a specific mapping function between the color model and a reference color space (such as CIELAB or CIEXYZ color spaces) to define the color gamut (certain complete subset of color) within the reference color space. Note that, CIELAB and CIEXYZ are both a color model and a color space.

A vast choice of different physical color spaces such as sRGB, AdobeRGB based on the RGB model, and perceptual color spaces such as CIE Lab and CIE XYZ or CIECAM02 [Moroney et al. 2002] were proposed over the years. The discussion of what space or what model is best for which purpose is extensive; some are discussed in [Tkalcic and Tasic 2003].

Specialized Color Spaces In computer vision, some specialized color spaces have been proposed by extracting statistics of colors. An example is the work of Hsu, Abdel-Mottaleb and Jain [2002], who proposed a color space for human skin to be used for face detection. They use the principal component analysis (PCA), which implies that the best manipulation happens along a particular linear direction in RGB with equally-sized steps. Image-dependent PCA has also been used to improve compression of color images [Clausen and Wechsler 2000]. For a general survey of dimension-reduction techniques in color

science, where they are mostly applied to reducing high-dimensional spectral signals to low-dimensional spaces, see Tzeng and Berns [2005]. The color science community has addressed the deformation of space to fit to certain data for problems of linearization in agreement to some measurement, e. g., hue [Lissner and Urban 2009]. Omer and Werman [2004] use a set of 1D-subsets of a color space (lines) to detect and reduce distortions of colors in acquisition and reproduction of images. They extract multiple disconnected 1D lines and do not account for varying (perceptual) density of color distribution. While multiple disconnected lines can serve as a regularization to restrict the set of colors to plausible ones, they do not allow for an intuitive user interface as there is no obvious way how to embed a set of disconnected lines into a single slider. Finally, they do not capture two-dimensional relationships.

2.2 Material

In computer graphics, creating realistic images requires simulating and modeling of real world materials. While the appearance of an object from a certain view in specific illumination settings can be represented by colors alone, it is not obvious to predict object appearance under different views or different illuminations. In this case, material models can be used as a predictive model for object appearance under arbitrary views and illumination settings. Material models contain a higher number of dimensions compared to color models.

2.2.1 Material Model

Material models can (potentially) account for all effects of light scattering through surfaces such as sub-surface scattering in translucent materials (e. g., milk, skin), wavelength-dependent effects (e. g., fluorescence). Various material models with different levels of complexity have been proposed over the years.

In this thesis, we consider a subclass of materials that only accounts for the reflection of light within the upper hemisphere, modeled by the Bi-directional Reflectance Distribution Function (BRDF). The BRDF is a four-dimensional function that defines how light is reflected at an opaque surface.

In general, BRDF models can be classified into *empirical* models, often not physically-correct, e. g., Phong [1975], Blinn [1977], and *physically-based* models, e. g., Cook and Torrance [1982], Ward and Heckbert [1992], or *intermediate* models that stay between empirical and physically-correct, e. g., Schlick [1994] or Ashikmin and Shirley [2000]. Empirical models are computationally efficient but lack the physical validity. Therefore, they are used in applications where interactivity is important (e. g., interactive global illumination). Physically-correct models involve higher computational costs are adapted to applications that need physically-based rendering. In the scope of this thesis, we use Phong model as it is widely used by artists, easier for artistic control and better suited to our proposed interactive applications.

Recently, there are approaches that tried to extract material properties from captured data. Materials shared by several surfaces can be acquired from a single image [Tominaga and Tanaka 2000] or a clustered of images captured under controlled environment such as in

the work of Lensch et al. [2003] and Matusik et al. [2003], or via user interaction [Dong et. al. 2011].

2.2.2 Material Perception

While material models are widely used in practice, "thinking" in terms of parameters such as "gloss" does not map well to human perception. Even though some perceptual-motivated material models were proposed [Pellacini et. al. 2000; Wills et al. 2009; Sadeghi et al. 2010], dealing with material models in a perceptually meaningful way stays challenging. The perceptual disambiguation of light and materials under direct [Land and McCann 1971] or indirect [Langer 1999] illuminants further complicates the issue. Nishida and Shinya [1998] report difficulties in matching gloss in the Phong model for height fields of different spatial frequency and amplitude. The relation between measured specular gloss values and the perceived gloss is highly non-linear, where the sensitivity for changes is higher at extreme (low and high) scale values than in the middle [Obein, Knoblauch and Viéot 2004]. Doerschner, Maloney and Boyaci [2010] observe that background affects perceived gloss and albedo markedly: they are higher for objects placed in front of a dark background than a bright one. Fleming, Dror and Adelson [2003] showed that glossiness constancy is not perfect in illumination conditions close to the real world, when captured HDR environment maps are used to illuminate rendered scenes. Vangorp, Laurijssen and Dutré [2007] performed experiments for even more complex, realistically rendered scenes, and observed that identical materials may have different appearance for differently shaped objects. Also, the detectability of perceived differences between materials with manipulated reflectance parameters depends on the type of lighting and object shape. All those observations suggest a weak material constancy under varying illumination, object shapes, surface structure, different object layout, and viewing conditions. This may indicate that judging material properties independently without taking all these factors into account might be difficult.

2.3 Rendering

Realistic synthesis of virtual 3D scenes can be considered as the simulations of light transport. In this section, our discussion of global illumination rendering will start with some theoretical background and basic notation in rendering (Section 2.3.1) that are later used in Chapter 3 and Chapter 4. We then discuss computationally efficient rendering techniques (Section 2.3.3) that are used in rendering and editing of physically-based illumination.

2.3.1 Rendering Equation

The rendering equation (RE) [Kajiya 1986] describes the radiance L_o leaving at location $\mathbf{x}$ on a surface $\mathcal{M} \subseteq \mathbb{R}^3$ in direction ω_o as an integration over all incoming directions. In the following, we will ignore the dependency on wavelength and assume all operations are performed on all color channels.

$$L_\mathrm{o}(\mathbf{x}, \omega_\mathrm{o}) = L_\mathrm{e}(\mathbf{x}, \omega_\mathrm{o}) + \int_{\mathbb{S}^2} L_\mathrm{i}(\mathbf{x}, \omega_\mathrm{i})\, R(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o})\, \langle n(\mathbf{x}), \omega_\mathrm{i} \rangle^{+} \mathrm{d}\omega_\mathrm{i},$$

where L_e is the emitted radiance, L_i is the radiance coming toward $\mathbf{x}$ from direction ω_i, $n(\mathbf{x})$ is the normal at $\mathbf{x}$, $R(\mathbf{x}, \omega_i, \omega_o) \in \mathcal{M} \times \mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{R}^+$ is the BRDF function from the incoming direction ω_i to the outgoing direction ω_o.

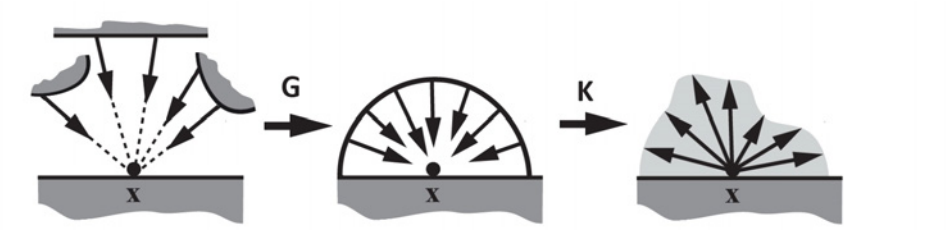

Figure 2.3: The actions of $\mathbf{G}$ and $\mathbf{K}$ at a single point $\mathbf{x}$. The operator $\mathbf{G}$ converts exitant surface radiance *(left)* directed toward $\mathbf{x}$ into incoming radiance *(middle)*, where it is again mapped into exitant surface radiance by the reflection operator $\mathbf{K}$ *(right)*.

Light reflection can be understood as a convolution of incoming radiance L_i with the BRDF R, we can rewrite the rendering equation in operator form [Arvo, Torrance and Smits 1994], using the *reflection operator* $\mathbf{K} \in (\mathcal{M} \times \mathbb{S}^2 \times \mathbb{S}^2) \times (\mathcal{M} \times \mathbb{S}^2) \to \mathcal{M} \times \mathbb{S}^2$ (Figure 2.3),

$$\mathbf{K}(R)L_i(\mathbf{x}, \omega_o) := \int_{\mathbb{S}^2} L_i(\mathbf{x}, \omega_i) R(\mathbf{x}, \omega_i, \omega_o) \langle n(\mathbf{x}), \omega_i \rangle^+ \mathrm{d}\,\omega_i$$

Next, we define a *geometry operator* $\mathbf{G} \in \mathcal{M} \times \mathbb{S}^2 \to \mathcal{M} \times \mathbb{S}^2$ produces the field of incident radiance from a field of exitant radiance (Figure 2.3):

$$\mathbf{G}\,L_o(\mathbf{x}, \omega) := L_o(v(\mathbf{x}, \omega), \omega),$$

where the *raycasting* function $v(\mathbf{x}, \omega)$ returns the position that is closest to $\mathbf{x}$ along a ray from $\mathbf{x}$ in direction ω. This operator includes the visibility and turns distance surface radiance into local incident radiance. This allows to rewrite the RE as

$$L_o = L_e + \mathbf{K}(R)\,\mathbf{G}\,L_o,$$

Arvo, Torrance and Smits [1994] show that such equation can be solved using an infinite Neumann series and the solution to the RE is

$$L_o = L_e + \mathbf{K}(R)\,\mathbf{G}\,L_e + \mathbf{K}(R)\,\mathbf{G}\,\mathbf{K}(R)\,\mathbf{G}\,L_e + \ldots.$$

Or shorter, using the i-bounce *transport operator* $\mathbf{T}$

$$\mathbf{T}^i(R) = \sum_{j=1}^{i} (\mathbf{K}(R)\mathbf{G})^{j-1} \quad \text{and} \quad \mathbf{T}^0(R) = \mathbf{I}.$$

2.3.2 Surface Light Fields (SLFs)

Light field represents the radiance at a point $\mathbf{x}$ in space in a given direction ω. Light fields can be used to reconstruct a faithful image-based rendering by densely sampling the plenoptic function using a camera array [Levoy and Hanrahan 1996; Gortler et al. 1996].

Similar to light fields, SLFs [Miller, Rubin and Ponceleon 1998] map every location $\mathbf{x}$ on a surface $\mathcal{M}$ and direction ω to the outgoing radiance $L_o(\mathbf{x},\omega) \in \mathcal{M} \times \mathbb{S}^2 \rightarrow \mathbb{R}^+$. We simply define $L_o(\mathbf{x})$ as the SLF at location $\mathbf{x}$. In this thesis, we restrict the viewing directions to the upper hemisphere. Intuitively, a SLF describes how a surface looks from different viewing directions. Diffuse surfaces are invariant under changing of view directions. Little spheres will be used in this thesis to visualize the SLF at a certain location (Figure 2.4). In Chapter 4, we propose a new material editing approach that is based on the manipulation of SLFs.

More general than BRDF, SLFs can be combined, edited [Wood et al. 2000] and displayed interactively [Horn and Chen 2007]. Regrettably, SLFs resulting from this approach are not always physically meaningful or valid and manipulation is restricted to basic compositing.

2.3.3 Approximate Global Illumination

Solving the rendering equation for a given scene is the main target of realistic rendering. Unbiased techniques (e. g., path tracing, bidirectional path tracing [Lafortune and Willems 1993], Metropolis light transport [Veach and Guibas 1997]) do not introduce any systematic error into solving the RE. While these techniques generate physically correct images, they are computationally expensive and not applicable for interactivity. In this section, we review techniques that approximate global illumination, producing plausible rendered images in a computational efficient manner.

Irradiance and Radiance Caching To approximate global illumination, the idea of re-using illumination computation results between pixels dates back to work by Ward, Rubinstein and Clear [1988], where scalar irradiance incident on diffuse surfaces is computed using raytracing for a subset of pixels and interpolated for the others. Several different approaches were proposed to place [Greger et al. 1998] and interpolate cache items, including the idea of using gradients [Ward and Heckbert 1992].

Irradiance caching [Ward, Rubinstein and Clear 1988] turns reflection computation into a mixture of irradiance computed for a number of discrete cache items

$$L_o(\mathbf{x},\omega_o) = \rho_d(\mathbf{x}) \sum_{j=1}^{n_c} w(\mathbf{x},\mathbf{x}_j)\, E(\mathbf{x}_j),$$

where $w(\mathbf{x},\mathbf{x}_j) \in \mathbb{R}^3 \rightarrow [0,1]$ is a weighting function such as a Gaussian kernel where $\sum_{j=1}^{n_c} w(\mathbf{x},\mathbf{x}_j) = 1$ for a fixed $\mathbf{x} \in \mathbb{R}^3$, E is the irradiance at cache location $\mathbf{x}_j$ and $\rho_d(\mathbf{x})$ the diffuse albedo. Doing so, the costly irradiance computation is only required at a fixed, low number n_c of discrete cache locations $\mathbf{x}_j$ which allows to pre-compute and store it. In practice, the sum is iterated only over a low number of nearby cache items, where w is non-zero. However, to shade at a pixel, several cache items have to be *evaluated*, i. e., the reflectance operator must be applied. Although the method handles diffuse color bleeding nicely, details on highly specular surfaces are often missing.

Radiance Caching [Křivánek et al. 2005] extends Irradiance Caching, in that it interpolates

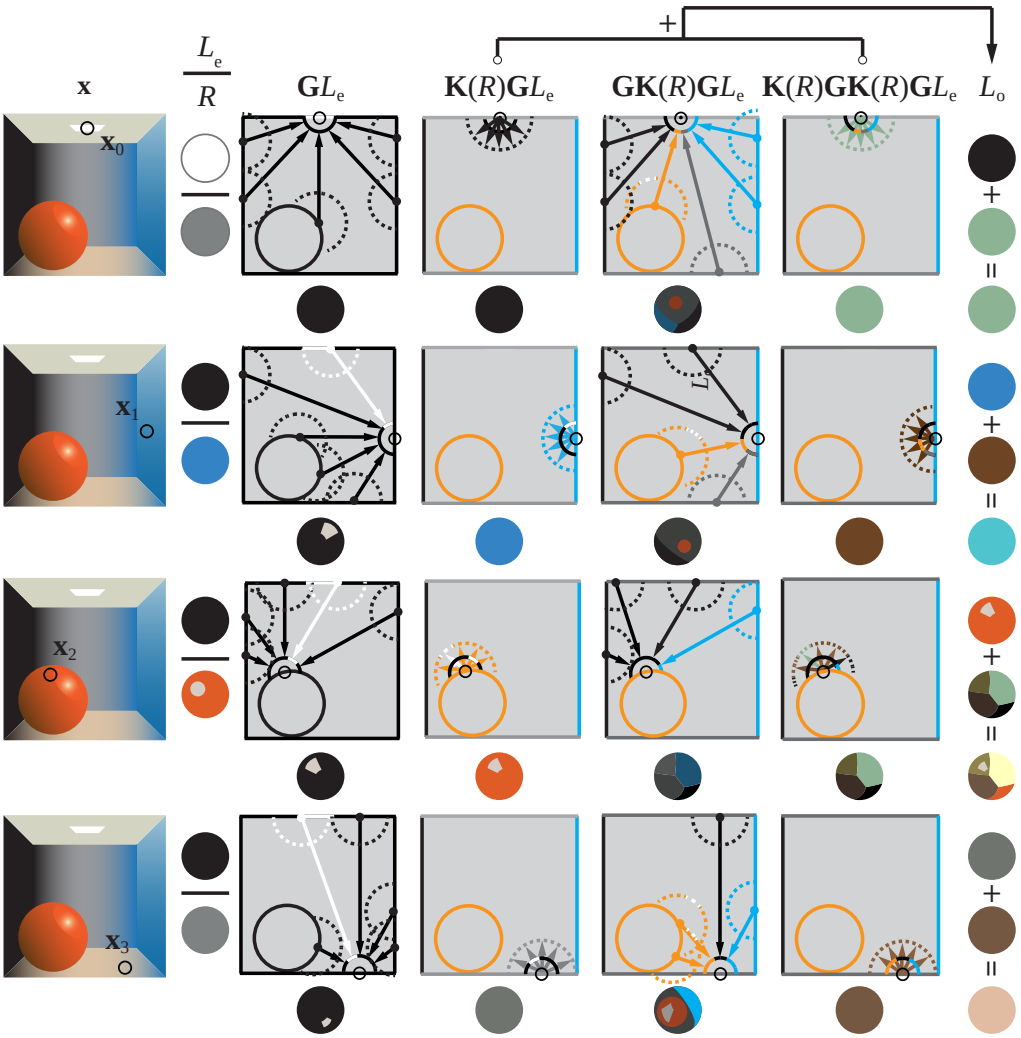

Figure 2.4: Steps of different operators at four different locations $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ *(rows)*. The *first column* shows the location $(\mathbf{x})$ in 3D. The *second column* shows the emitted radiance (L_e) *(top)* and a 2D slice of the BRDF (R) *(bottom)*. Note that only $\mathbf{x}_0$ lies on a self-emitting area light source, it is colored white (area light) and the others are colored black. Furthermore, $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_3$ lie on diffuse surfaces, their BRDF slices are colored homogeneously, while the slice of $R(\mathbf{x}_2)$ shows a specular white highlight. In the *third column*, operator $\mathbf{G}$ turns distance surface radiance *(colored arrows)* into local field *(colored semicircle that gathers all the arrowheads)*. Scene frames are colored according to L_e. In the *fourth column*, scene frames are colored according to the diffuse color of BRDF; at every row, operator $\mathbf{K}$ convolves local field radiance *(colored semicircle from the third column)* and BRDF to surface radiance *(dashed colored curves)*. Further bounces are traced by applying operator $\mathbf{G}$ *(5st col.)*, and $\mathbf{K}$ *(6st col.)* alternatively, again. The final column shows the surface light field as the outgoing radiance $L_o = \mathbf{K}(R)\mathbf{G}L_e + \mathbf{K}(R)\mathbf{GK}(R)\mathbf{G}L_e$ at different outgoing direction ω_o after 2 bounces.

the incoming light and performs the reflection every time a cache item is queried:

$$L_{\mathrm{o}}(\mathbf{x},\omega_{\mathrm{o}}) = \sum_{j=1}^{n_{\mathrm{c}}} w(\mathbf{x},\mathbf{x}_j) \int_{\mathbb{S}^2} L_{\mathrm{i}}(\mathbf{x}_j,\omega_{\mathrm{i}})\, R(\mathbf{x},\omega_{\mathrm{i}},\omega_{\mathrm{o}})\, \langle n(\mathbf{x}),\omega_{\mathrm{i}}\rangle^{+} \mathrm{d}\omega_{\mathrm{i}}.$$

If the incoming light field has fine details, the reflection can be a costly operation .

Radiance Caching represents incoming radiance $L_{\mathrm{i}}(\mathbf{x},\omega_{\mathrm{i}})$ using the spherical harmonics basis:

$$L_{\mathrm{i}}(\mathbf{x},\omega_{\mathrm{i}}) \approx \sum_{k=1}^{n_{\mathrm{d}}} c_k(\mathbf{x}) B_k(\omega_{\mathrm{i}}),$$

where $c_1,\ldots,c_{n_{\mathrm{d}}}$ are n_{d} coefficients and $B_k(\omega_{\mathrm{i}})$ is the k-th SH basis function [Sloan et. al. 2002]. Similarly, the BRDF can be approximated with

$$R(\mathbf{x},\omega_{\mathrm{i}},\omega_{\mathrm{o}}) \approx \sum_{k=1}^{n_{\mathrm{d}}} f_k(\mathbf{x},\omega_{\mathrm{o}}) B_k(\omega_{\mathrm{i}}),$$

using n_{d} coefficients $f_1,\ldots,f_{n_{\mathrm{d}}}$. To reflect the incoming lighting using the BRDF R at position $\mathbf{x}$, only a dot product is required:

$$L_{\mathrm{o}}(\mathbf{x},\omega_{\mathrm{o}}) = (L_{\mathrm{i}} * R)(\mathbf{x},\omega_{\mathrm{o}}) \approx \sum_{k=1}^{n_{\mathrm{d}}} c_k(\mathbf{x}) f_k(\mathbf{x},\omega_{\mathrm{o}})$$

Computationally, for every pixel the evaluation requires interpolating the SH coefficients of the incoming radiance, arithmetic operations for spherical harmonics rotation into the local frame, and a loop over n_{d} coefficients, that in each step performs a (texture-) read per operation for c_k and a multiplication with the BRDF coefficient f_k. In short, the per-pixel work again is *linear* in the number of coefficients, i. e., in the amount of directional detail that can be represented.

Radiance Caching [Křivánek et al. 2005] overcomes the limitation to diffuse reflectance by storing incoming radiance as a directional function, interpolating it between pixels and convolving with the BRDF for every pixel. For highly specular surfaces, however, a high number of SH coefficients has to be stored and evaluated per pixel i. e., a simple dot-product, but on a high-dimensional vector. For alternative representations of the radiance function [Gautron et. al. 2004] the quality can be improved, but the storage and computational cost remain the same. A GPU friendly version of radiance caching is based on splatting [Gautron et. al. 2005]: instead of finding the cache items that map to a pixel, cache items are traversed and mapped to all pixels they affect. Vector irradiance allows to approximate the lighting directionality of RC using a number of discrete directions but at a cost similar to IC [Tabellion and Lamorlette 2004]. This works well if surfaces are moderately glossy and lighting is dominated by a low number of dominant light directions.

Enviromentmap Pre-filtering To avoid computing the BRDF-lighting product in the case of distant lighting, pre-convolved irradiance maps were used [Greene 1986; Heidrich and Seidel 1999]. The idea to use MIP maps for this purpose is as old as MIP mapping itself [Williams 1983]. Heidrich and Seidel [1999] generalize the original diffuse pre-convolution

to glossy reflections. Multiple reads from a MIP map can be used to approximate the convolution with BRDF more faithfully [Kautz et al. 2000]. Alternatively, environment maps can be stored in the frequency domain and convolved with BRDFs using dot products [Ramamoorthi and Hanrahan 2001].

Pre-computed Radiance Transfer Pre-computed radiance transfer (PRT) methods [Sloan et. al. 2002] address the issue of computing the convolution of BRDF and lighting, or in the case of distant lighting the triple product of light, visibility and reflectance. Using wavelets, the complexity of products can be reduced drastically [Ng, Ramamoorthi and Hanrahan 2003] when exploiting sparseness. However, non-linear wavelet compression requires irregular and dynamic data structures that do not map well to GPUs.

Point-based Global Illumination For complex local lighting and global illumination, most PRT and environment map pre-filtering ideas do not apply. Instead, Instant Radiosity [Keller 1997] (IR) or Point-based Global Illumination [Christensen 2008] (PBGI) are used, in particular in interactive GPU-based solutions [Ritschel et. al. 2008; Ritschel et. al. 2009a]. In such methods, upsampling based on regular structures is pre-dominant [Sloan et. al. 2007], e. g., using joint bilateral upsampling or edge-aware G-buffer blurs [Laine et al. 2007].

Voxel-based Global Illumination Earlier, Malgouyres [2002] approximated global illumination by using a voxel-based representation for discrete radiosity solver; Haumont and Warzée [2002] proposed a fast automatic method to convert polygonal scenes into volumetric representation, however, it is still computationally heavy and not suitable for realtime applications. Later, voxel scene representations have gained attention due to the advance of powerful GPUs; it was shown how to efficiently exploit the rasterization pipeline of graphics hardware to generate a voxelized grid of a polygonal scene in real-time [Dong et. al. 2004; Eisemann and Décoret 2008] and even giga-voxel grids can be processed at interactive rates [Crassin et. al. 2009]. Recently, Crassin et. al. [2011] used hierarchical voxel octree representation of a scene, coupled with an approximate voxel cone tracing for fast estimation of visibility and incoming energy; as a result, global illumination can be computed interactively.

2.4 Appearance Editing

Appearance editing is a process where an artist seeks for a specific visual outlook of images or virtual 3D scenes that matches a certain vision. Many methods were proposed over the years, and in this section, we will only discuss some that are most related to our work. Fundamental editing operations such as color editing in images is discussed in Section 2.4.1 and light, shadow, material editing in 3D scenes are discussed in Section 2.4.2. Next, two instances of low-level editing techniques: edit propagation and subspace-aware editing techniques, are discussed in Section 2.4.3 and Section 2.4.4, respectively. We recall some work on style transfer, a high-level editing technique, in Section 2.4.5 and finally, appearance manifold in Section 2.4.6.

2.4.1 Color Editing

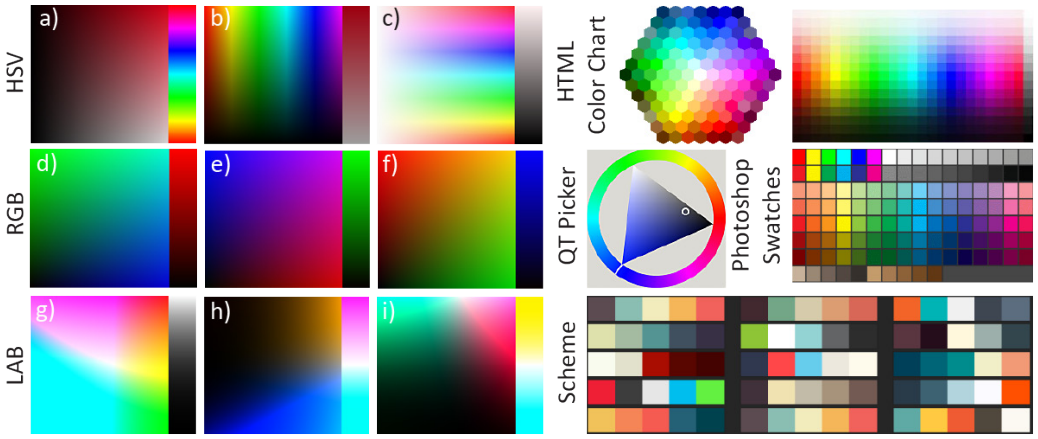

Figure 2.5: Common color pickers. On the left are the HSV *(a-c)*, RGB *(d-f)*, LAB *(g-i)* color pickers. On the right, from top to bottom are the HTML color chart, the color picker based on hue wheel of QT and Photoshop's color swatches. Finally, on the bottom right are several color schemes from Adobe Color [Adobe Systems Incorporated 2014a].

Editing color in photographs is one of the most common digital content editing task. Color editing is the process of replacing the original colors in a photograph by newly selected colors from a color picker to alter image appearance, e. g., to re-color human skin in images.

Color Picker A color picker is an utility, usually found within graphics software, used to choose colors or create color schemes. A color picker provides an interface to traverse the higher dimensional color space in 2D. Color pickers can vary in interface, e. g., an HSV color picker that uses the HSV color model (Section 2.1.2) has two rectangles to create a range of colors, the thin rectangle on the right for one color component, such as hue, and the square on the left for the remaining color components, such as saturation and value (Figure 2.5 (a)). Adobe Photoshop provides RGB, HSV and LAB color pickers and many color pickers exist on the World Wide Web that include features such as color harmonization, or a collection of predefined color schemes (Figure 2.5).

Color Selection As a human-computer interaction, color selection has received only little attention with the exception of work by Schwarz, Cowan and Beatty [1987] as well as by Douglas and Kirkpatrick [1999]. They found that the choice of color space has only little impact on performance when comparing different color spaces and visual feedback is the most important usability factor. Performance of specialized color spaces for color selection tasks [Hsu, Abdel-Mottaleb and Jain 2002; Omer and Werman 2004] are not well studied. Shapira, Shamir and Cohen-Or [2009] present an exploratory interface to edit image appearance interactively. Their approach is also concerned with modeling the distribution of colors. Color distribution is modeled as mixture of Gaussians and used for manipulation. A user can visually navigate the high-dimensional space of possible color manipulations by transforming the Gaussian mixtures. Each pixel's color follows its distribution accordingly.

Color Templates and Themes The relation of images to colors, called "Image Themes" is extracted, transferred and enhanced in the works of [Wang et. al. 2010; Wang et. al. 2011]. Color templates are a pre-defined discrete selection of colors. Popular Internet sites, such as Adobe Color [Adobe Systems Incorporated 2014a], provide a large collection of such color templates. Templates are well-suited for picking a combination of colors, but less suited to fine adjustment of colors. Hue templates were studied by Matsuda [1995] and later used for color harmonization [Cohen-Or et al. 2006]. A perceptual study of such color templates was conducted by ODonovan et. al. [2011].

2.4.2 Light, Shadow and Material Editing

Light, shadow and material editing approaches are basically classified into two main categories: forward and inverse. Forward methods allow the user to perform manipulation by changing the parameters of light, shadow or material directly and provide interactive feedback. On the other hand, inverse approaches allow the user to specify a desired appearance and the matching parameters of reflectance, light or geometry is found automatically.

Forward Approaches Interactive relighting techniques assume fixed geometry and materials. The user can change the light parameters and the relighting system interactively re-lights the scenes. Pellacini et. al. [2005] used deferred shading to allow interactive relighting for computer generated scenes. For complex environment maps, PRT-based techniques were proposed for relighting [Ng, Ramamoorthi and Hanrahan 2004; Sloan et. al. 2002] and all-frequency shadowing [Ng, Ramamoorthi and Hanrahan 2003]. While early work only supported the direct illumination model, the support of indirect illumination was proposed in later work [Hašan, Pellacini and Bala 2006; Cheslack-Postava et al. 2008].

Material editing frameworks focus on the direct manipulation of BRDF parameters. Intuitive editing of BRDFs under complex environment lighting, limited to direct illumination is presented by Ben-Artzi, Overbeck and Ramamoorthi [2006]. Later work extended to editing with global illumination [Ben-Artzi et al. 2008; Cheslack-Postava et al. 2008]. These methods are based on PRT which takes long time (minutes to hours) for precomputation and impose a fixation of the view point [Nguyen et. al. 2010] or the number of editable BRDFs [Ben-Artzi et al. 2008].

Inverse Approaches Paint with light [Schoeneman et al. 1993] and Radio-optimization [Kawai, Painter and Cohen 1993] are classic solutions to infer light settings from user constraints. Pellacini et. al. [2002] propose an interactive system to design shadows and lights, but decoupled from materials. Gingold and Zorin [2008] use painted changes in light to define a change in shape. Intuitively repositioning shadows, caustics, and indirect illumination using simple click-and-drag on surfaces is proposed by Ritschel et. al. [2010]. Mattausch, Igarashi and Wimmer [2013]'s system allows artist to directly edit the shadow boundaries in the scene in an intuitive fashion similar to free-form curve editing.

Poulin and Fournier [1995] were the first to propose a system that infers material parameters for a surface directly lit by a point light seen from a single view point. Anjyo and

Hiramitsu [2003] proposed the interactive control of highlight shapes by painting, but this technique is limited to the area of cartoon animations. User is provided with image manipulation tools, to change shading, that is used in an optimization approach to fit a surface. However, their work only addresses diffuse surfaces. Ritschel et. al. [2009b] introduce a system for interactive reflections editing by directly specifying the reflection constraints. Kerr and Pellacini [2010] evaluate user interfaces for material design, and identify color editing as a main issue. A full survey of artistic lighting and material editing is discussed in Schmidt et al. [2014].

2.4.3 Edit Propagation

In this section, we discuss some edit propagation techniques that build on propagation of sparse manipulations of color, material or shape to the full image. To manipulate color, sparse strokes, defining a certain color for some locations in an image, are propagated to the rest of the image, either using edge-aware (local) [Levin, Lischinski and Weiss 2004; Lischinski et al. 2006; Gastal and Oliveira 2011] or all-pairs propagation (global) approaches [Pellacini et. al. 2007; An and Pellacini 2008] or a unification of both global and local methods [Xu, Yan and Jia 2013]. For example, a red ball is covered by a green stroke, and other red and spatially close balls are changed to become green. The goal of the propagation often is to fill smooth regions and stop propagation at edges. Chapter 4 generalizes these propagation approaches by taking illuminant and geometry into account when propagating edits.

Appearance editing includes but is not limited to color and material editing. Other instances of appearance editing are shape manipulation in 2D images or geometry modeling in 3D. For manipulating shape, instead of making strokes, sparse control points are moved, and a plausible deformation is propagated to the rest of the 2D image [Bookstein 1989; Alexa, Cohen-Or and Levin 2000; Igarashi, Moscovich and Hughes 2005; Schaefer, McPhail and Warren 2006] or 3D geometry [Cuno et al. 2007]. As an example, a user clicks two control points such as the feet of a character in the image to remain fixed and places a third one on the head. Moving the control point on the head, the latter follows the displacement, but the feet remain in place. A key challenge is to produce an intuitive deformation response and avoid shape distortion and overlap. Deformations that locally preserve distances are of particular interest here [Alexa, Cohen-Or and Levin 2000; Igarashi, Moscovich and Hughes 2005; Schaefer, McPhail and Warren 2006; Cuno et al. 2007].

2.4.4 Subspaces-aware Editing

For edit propagation approaches discussed in Section 2.4.3, the user either knows where or how to change the image and the propagation is performed based solely on the image and the user's edits. In case we want to exploit more information from other sources (such as data available on the Internet) to improve the propagation, subspace-aware editing is an interesting alternative choice.

Subspaces Subspaces have a long history of use in computer graphics where they come in form of Eigenfaces [Turk and Pentland 1991], morphable models [Blanz and Vetter 1999]

or active appearance models [Cootes, Edwards and Taylor 2001]. Their usages range from faces [Blanz and Vetter 1999] over human body poses [Allen, Curless and Popović 2003] to other objects such as demonstrated for sea animals by Cashman and Fitzgibbon [2013]. The difficulty is how to acquire such a space. Chapter 6 proposes shape and color subspaces of an object, built from a collection of exemplar images available on the Internet, that can be used to restrict manipulations.

Alignment The underlying difficulty in constructing subspaces is the requirement of correspondences between training exemplars. Early work used either controlled conditions to acquire the exemplars [Blanz and Vetter 1999] or manual alignment. Modern semi-automatic approaches to align image pairs e. g., for morphing [Liao et al. 2014] often combine a data and a smoothness term with user-defined constraints. Automatic alignment has been done for 3D shapes [Sumner et al. 2005; Feng, Kim and Yu 2008] and deformations limited to rigid parts [Kokkinos and Yuille 2007]. Optical flow algorithms [Lucas and Kanade 1981; Tao et al. 2012; Lang et al. 2012] are designed to align video frames with their temporally adjacent frames. However, they are not suitable for image pairs containing large deformations, significant structural differences or changes in appearance. Recently, several image alignment techniques such as SIFT flow [Liu, Yuen and Torralba 2011] or Patch Match [Barnes et al. 2009] were proposed that are able to deal with drastically different scales and would allow to align training images. While they are successful in producing plausible images by shuffling image patches, they do not yield meaningful deformation fields. Non-rigid dense correspondence (NRDC) [HaCohen et al. 2011] is based on Patch Match and works best for pairs of images depicting similar regions acquired by different cameras or under different lighting conditions undergoing non-rigid deformations. The idea of co-aligning an image collection is demonstrated in ImageWeb [Heath et al. 2010] which assumes a partial, per-region affine transformation. That is great for browsing image collections with repeating objects but does not capture smoothly varying deformations such as between two faces. Furthermore, no considerations are made for the spatial placement of the regions themselves.

Cheng et al. [2010] use a boundary-band map to find repetitions of similar objects in single images. Goldberg et al. [2012] present a semi-automatic system that leverages a collection of aligned images to improve image manipulations. Manual intervention is used and the alignment is limited to align the outer boundary using shape contexts [Belongie, Malik and Puzicha 2000]. Furthermore, their approach does not create a space and cannot be used to understand the variation of e. g., horses, as a whole. Outside of computer graphics, the most successful alignment methods proposed in computer vision use involved learning and graph matching machinery for alignments [Caetano et al. 2009]. Chapter 6 proposes an automatic alignment approach that uses all structures that can be matched reliably, including internal structures, such as the eyes of animals. Aligning these properly is essential for creating an expressive space.

2.4.5 Style Transfer

Low-level appearance editing operations, such as edit propagation (Section 2.4.3) and subspace-aware editing (Section 2.4.4), allow direct control from the user to fine tune the

final appearance. On the other hand, high-level editing methods, such as color transfer [Reinhard et al. 2001] or scene attributes adjustment [Laffont et al. 2014], aim for more global edit effects. In this section, we will discuss high-level style transfer within a single or amongst different media, which has been a longstanding challenge in computer graphics.

Mood Perception Style transfer methods are concerned with style perception (i. e., its mood), and how style can be extracted and transfered. Humans perform remarkably when quickly categorizing natural images into a classes (its *gist* [Oliva and Torralba 2001]). Possible cues include spatial organization of textures [Oliva and Torralba 2001; Walker and Malik 2002], or color [Castelhano and Henderson 2010], but also the organization of shapes, colors, spatial proximity [Berg 1948], congruence [Stroop 1935] and grouping (e. g., the scene's Gestalt) can influence the scene's mood. Human observers are extremely efficient in material categorization, which is a rapid effortless process; robust performance was reported even for a 160 ms image display [Lavanya Sharan 2008], and only slightly more for grey-leveled, blurred, or inverted-contrast stimuli.

Color Transfer Color transfer alters the appearance of an image to look natural by adjusting their content using the characteristic of another image [Reinhard et al. 2001; Irony, Cohen-Or and Lischinski 2005; Luan et al. 2007; Liu et al. 2008; Wang et. al. 2010]. Chapter 5 proposes a system for material style transfer from image to a target 3D scene. Transferring material style differs from of the classic color transfer problem as materials include a much higher number of parameters (such as glossiness and textures) and the 3D scenes contain geometry that can be used to make the transfer more reliable than it could be with images only.

Texture Transfer Another important part of style are surface details often in form of textures. Texture synthesis [Heeger and Bergen 1995] takes a statistical approach and produces new instances from a training example or manual statistical settings. The Image-Analogies framework of Hertzmann et al. [2001] can produce new exemplars with details that fit specified content. In the context of 3D models, Mertens et al. [2007] model the statistical relationship between local geometric properties, such as curvature and local statistics of reflectance. Using this relation, a texture synthesis on a new object produces shape-dependent textures that capture, e. g., weathering. In a similar fashion, Chajdas, Lefebvre and Stamminger [2010] consider local geometric structure to assist the user in assigning textures. Their transfer is from 3D to 3D and does not consider the resulting perceived appearance, but solely statistical physical qualities. Extraction of a single texture to be then assigned to a target 3D object is considered by Lagae et al. [2010]. The CG2Real system [Johnson et al. 2010] uses large image collections to decorate a synthetic image with details. Their results are image compositions and cannot be used easily for 3D scenes.

Color Compatibility Lalonde and Efros [2007] have analyzed color distributions of typical classes of images. A distance measure between distributions can be used to assess color compatibility of one image to a certain class or the compatibility of a foreground and a

background. Oskam et al. [2012] address the problem of global color balancing between images using a sparse set of desired color correspondences by deforming the color space.

2.4.6 Appearance Manifolds

Multi-dimensional-scaling (MDS), as used in the Design Galleries framework [Marks et. al. 1997], allows for embedding higher-dimensional qualia into lower-dimensional layouts. Matusik et al. [2003] sample the even higher-dimensional space of BRDFs to create a neighborhood graph that allows to move to nearby plausible BRDFs. Appearance manifolds have been widely used in computer vision and computer graphics community. Wang et. al. [2006] build appearance manifolds to capture time-variant appearance of materials from data captured at a single instant in time. Xue et al. [2008] model the reflectance of weathered surfaces from a single input image as a manifold and use it for interactive editing of the weathering effects in an image. Recently, construction of font manifolds was proposed [Campbell and Kautz 2014]. Chapter 7 proposes a data-driven color manifold, the comparatively low dimensionality of colors allows us to fit a parametric model with an explicit dimensionality to the color distribution resulting in a smooth and continuous manipulation and layout of the color structure. Furthermore, we have a density measure defined on the high dimensional color data that is preserved as local area changes (Jacobian) of the embedding.

2.5 Statistical Hypothesis Testing

This section provides some basic background on statistical hypothesis testing for scientific study. As many appearance editing techniques were proposed, a scientific validation is required to clarify which method produces better results than others in a scientifc study, e. g., between subspace-aware color editing and edit propagation in a color manipulation task.

Inferential statistics are used because it allows to measure behavior in limited samples and extrapolate to make a general conclusion about the behavior in groups that are often too large or inaccessible [Gravetter and Wallnau 2013]. A statistical hypothesis test is a method of inferential statistics using data from a scientific study. Important findings can be obscured by biological variability and experimental imprecision, which makes it difficult to distinguish real differences from random variation. In statistics, a result is called statistically significant if it has been predicted as unlikely to have occurred by chance alone, according to a pre-determined threshold probability: the significance level.

To illustrate, suppose we want to analyze the effects of different color pickers on the accuracy of color adjustment task (more details are described in Section 7.3.3 of Chapter 7). To perform statistical hypothesis testing, we first need to define the hypothesis (Section 2.5.1). Next, the criteria for a decision is set (Section 2.5.2) and an appropriate statistical test to compute the p-value is performed (Section 2.5.3). The conclusion is made about the statistical significance of the hypothesis and in case of many observations, a post-hoc test is performed to extract the pairwise relationship (Section 2.5.4). To further strengthen the study, a quantitative measure of the strength of the observation - effect size - should be provided to complement the statistical hypothesis testing (Section 2.5.5).

2.5.1 Hypothesis

Null Hypothesis The *null hypothesis* ($\mathbf{H}_0$) is a statement of no difference and contains the "equal to" (=) sign. If $\mathbf{H}_0$ is retained, the sample statistic and group parameter are not significantly different from each other. This means that it is likely that the sample came from the group described in $\mathbf{H}_0$.

Alternative Hypothesis The alternative hypothesis ($\mathbf{H}_a$) is a statement of difference and contains symbols implying direction or simply difference. If ($\mathbf{H}_0$) is rejected, the sample statistic is significantly different from (or greater than, or less than) the group parameter. This means that it is likely that the sample came from a group that is distinctly different from the one described in $\mathbf{H}_0$.

For our example, the null hypothesis is that accuracies using different color pickers have the same mean. When analyzing an experiment, the null hypothesis is usually the opposite of the experimental hypothesis. The experimental hypothesis, the reason of the experiment, that not all color pickers give the same accuracy, is the alternative hypothesis.

2.5.2 The p-value

	Decisions	
Truth	Retain $\mathbf{H}_0$	Reject $\mathbf{H}_0$
$\mathbf{H}_0$ True	Correct $1-\alpha$	Type-I Error α
$\mathbf{H}_0$ False	Type-II Error β	Correct $1-\beta$

Table 2.1: Type-I and Type-II Errors

The Level of Significance α Type-I error is the probability of rejecting a null hypothesis that is actually true. On the other hand, Type-II error, or β error, is the probability of retaining a null hypothesis that is actually false (Table 2.1). Researchers directly control for the probability of committing Type-I error.

An α level is the level of significance or criterion for a hypothesis test. It is the largest probability of committing a Type-I error that we will allow and still decide to reject the null hypothesis.

The p-value A p-value is the probability of obtaining a sample outcome, given that the value stated in the null hypothesis is true. The p-value is compared to the level of significance α. In practice, α is almost always set to 0.05 (an arbitrary svalue that has been widely adopted).

If $p < 0.05$, the null hypothesis is rejected and the results are deemed to be statistically

significant. It means that the observation stated in the null hypothesis would happen less than 5 % of the time.

If $p \geq 0.05$, the null hypothesis is retained. One can conclude that the observed results are not inconsistent with the null hypothesis, and the difference is not statistically significant. Note that it is incorrect to conclude that the null hypothesis is true. It is quite possible that the null hypothesis is false, and that there really is a difference between the groups.

2.5.3 Statistical Test

Now, in order to perform the appropriate statistical test to compute the p-value, we briefly introduce several test methods that are most suitable for the challenges encountered in this thesis. More details of the methods can be found in statistics books [Siegel and Castellan 1988; Cohen 1988; Gravetter and Wallnau 2013; GraphPad 2014].

Binomial Test The binomial test is an exact test, used when there are two possible outcomes (called "success" and "failure"). For example, we want to test whether our method or a competitor method performed better in a user study (more details in shape and color manipulation study in Chapter 6). Subjects are asked to choose the preferred results (generated using the two methods) in a two-alternative forced choice task. The null hypothesis is that the results generated using the two methods are chosen equally.

Student t-test The student t-test is a statistical method that is used to decide if two groups of data differ significantly. The method assumes that the results follow the normal distribution (also called student's t-distribution) if the null hypothesis is true.

In case the test contains more then two groups of data, such as in our example case of many different color interfaces. Every time a t-test is conducted there is a chance, e. g., $\alpha = 5$ %, that Type-I error would occur. By running two t-tests on the same data, the chance of "making a mistake" is increased, e. g., to 10 %. This is called the multiple testing problem that occurs when a set of statistical inferences is considered simultaneously [Miller 1966]. Different methods for multiple testing correction were proposed, one possible solution is to use Analysis of Variance (ANOVA) for multiple testing scenarios.

Consider our example where we want to study the accuracy in color adjustment tasks of different color pickers. The variable of interest is therefore the accuracy, measured by a scale. The factor being studied is color picker. There is just one factor (color picker) and hence the situation is appropriate for the one-way ANOVA.

One-way ANOVA The one-way ANOVA is used to determine whether there are any significant differences between the means of three or more independent (unrelated) groups. One-way ANOVA has several important assumptions:

- *Normality:* The dependent variable is normally distributed in each group that is being compared. So, for our example, if we were comparing three color pickers (e. g., the

RGB, LAB and HSV color picker), the accuracy of the color adjustment task (dependent variable) would have to be normally distributed for these three color pickers.
- *Homogeneity:* The variances in each group are equal.
- *Independence:* It is to be assured, that the observations (in our example, the accuracy of the color adjustment task) from the study are independent of the design of the study. These may includes many factors, e. g., randomness in selection of subjects for the study, the selection of a participant should not be dependent upon the selection of another participant, or the treatment condition.

One-way ANOVA can tolerate data that are non-normal with only a small effect on the Type-I error rate. However, the effect can be profound if the sample counts are small. In this case, data can be transformed using various algorithms so that the shapes of the distributions become normally distributed or alternatively, the non-parametric Kruskal-Wallis test (which does not require the assumption of normality) is used.

Kruskal-Wallis Test The Kruskal-Wallis test is a non-parametric test that compares three or more unpaired or unmatched groups. Unlike parametric tests, non-parametric tests make no assumptions about the probability distributions of the variables being assessed. The parametric equivalent of the Kruskal-Wallis test is the one-way ANOVA.

By selecting a non-parametric test, the *normality assumption* can be avoided, but there are drawbacks to using a non-parametric test: if the groups are really normally distributed, the non-parametric tests are less likely to detect a true difference, especially with small sample sizes.

The Kruskal-Wallis test still assumes that the shapes of the distributions are identical. If two groups have very different distributions, data are transformed to make the distributions more similar.

In Chapter 7, we use Kruskal-Wallis for our hypothesis testing as our distributions are not Gaussian and our sample size are moderate.

2.5.4 Post-hoc Test

ANOVA or Kruskal-Wallis test the overall difference between the groups, but they do not test which specific groups differed. A post-hoc test is required to compute the pairwise difference between two groups.

A post-hoc test should only be run when an overall significant difference in group means is found (i.e., a significant one-way ANOVA or Kruskal-Wallis). Post-hoc tests attempt to control the experiment-wise error rate, e. g., $\alpha = 5\,\%$, in the same manner that the one-way ANOVA or Kruskal-Wallis is used, instead of multiple t-tests.

A great number of different post-hoc tests were proposed. For one-way ANOVA, the Tukey's honestly significant difference (HSD) or Scheffe's post-hoc can be used. Often, Tukey's HSD test is recommended by statisticians because it is not as conservative as the Scheffe test [Maxwell and Delaney 2004]. For Kruskal-Wallis, Dunn's multiple comparison test can be used [Daniel 1990].

2.5.5 Effect Size

Hypothesis testing of different groups discussed earlier identifies whether an effect exists. A decision to reject the null hypothesis means that an effect is important, however, hypothesis testing does not inform how big the effect is. To determine this, the *effect size* is computed. There are different measures of effect size, here we will only discuss the definition of Cohen [1988], which will be used later in the thesis.

Cohen's d Cohen [1988] defined effect size d as the degree to which the null hypothesis is false. The effect size is measured between two groups. The larger the absolute value of d, the larger the effect.

$$d = \frac{\mu_1 - \mu_2}{\sigma}$$

where

$$\sigma = \sqrt{\frac{(n_1 - 1)\sigma_1^2 + (n_2 - 1)\sigma_2^2}{n_1 + n_2}}$$

is the pooled standard deviation, n_1, n_2 are the sample counts, σ_1, σ_2 are the variance of the two groups. The value of Cohen's d is zero when there is no difference between two means and increases as the differences get larger. Cohen's effect size conventions are shown in Table 2.2.

Description of effect	Effect Size (d)
Small	$\lvert d\rvert < 0.2$
Medium	$0.2 \leq \lvert d\rvert \leq 0.8$
Large	$\lvert d\rvert > 0.8$

Table 2.2: Cohen's effect size convention.

3

Preconvolved Radiance Caching

Figure 3.1: A virtual 3D scene with highly specular global illumination rendered using our method (1920×1080, 4 k caches, 50 ms radiance caching, 27 ms pre-convolution, Nvidia GeForce 560 Ti) *(top)*. The *bottom right* image shows the sparse set of cache locations *(colored circle)*. At every location, one specular and one diffuse cache were constructed. The *bottom left* image shows the specular and diffuse pre-convolved caches at one sample location.

3.1 Introduction

To deliver high-quality rendering results as fast as possible, the right simplifications have to be made. One such simplification is, to not shade every pixel, but only a subset, and re-use the shading of nearby pixels. Re-using scalar shading results (Irradiance Caching) between pixels is computationally efficient, but cannot account for details in geometry and reflectance resulting in blurred surface features and missing highlights. Re-using the full incoming light (Radiance Caching) can result in much richer images, but is computationally expensive. For interactive applications such as material design or architectural visualization that require high-quality rendering of complex light, spatially varying material, specular reflectance and detailed geometry, a combination of speed and quality is highly desirable. In this chapter we propose a modification to Radiance Caching by shifting computations that were done per-pixel before to the per-cache item stage to deliver almost the same quality, but an order of magnitude faster for HD images (Figure 3.1).

3.2 Our Approach

Background on Irradiance and Radiance Caching are discussed in Section 2.3.3. Radiance Caching requires a linear look up for every pixel. We extend the idea to a per-cache item pre-convolution in Section 3.2.1 combined with constant time per-pixel pre-convolved lookups (Section 3.2.2). Further, we will describe how to improve the lookups using recursion on distance information (Section 3.2.3) and finally provide implementation details in Section 3.2.4.

Our method, identical to Radiance Caching, takes a set of cache items and a deferred shading buffer with position, normals and reflectance as input and produces a high-resolution image with shading (Figure 3.2). Our approach is orthogonal to the way the cache items are placed or filled; details for our choice of implementation are given in Section 3.2.4.

3.2.1 Pre-convolution

The above per-pixel reflection is fine if the number of coefficients n_d is low, e. g., 9 for diffuse reflectance [Ramamoorthi and Hanrahan 2001]. For glossy surfaces, a much higher number of coefficients, e. g., $n_\mathrm{d} = 200$ are required [Ng, Ramamoorthi and Hanrahan 2003; Křivánek et al. 2005].

In this case, for every pixel, more than 200 RGB-values have to be read from a texture to evaluate one 15-th order SH per pixel, and the SH has to be transformed into the local frame. This takes more than 7 s on a Nvidia Geforce 560 Ti at a resolution of 1920×1080. Note, that every pixel is drawn many times, as the splats are overlapping. Consequently, the number of SH evaluations and rotations is much larger than the number of pixels. Our approach can perform a different computation leading to almost the same result at the time-cost of 50 ms + 12 ms = 62 ms and the additional cost of only one third more memory cost. This is achieved by converting per-pixel work to per-cache item work; 12 ms are spent on per-cache item pre-filtering resulting in only 50 ms spend on per-pixel evaluation. We will describe both the per-cache item and the per-pixel procedure in the following.

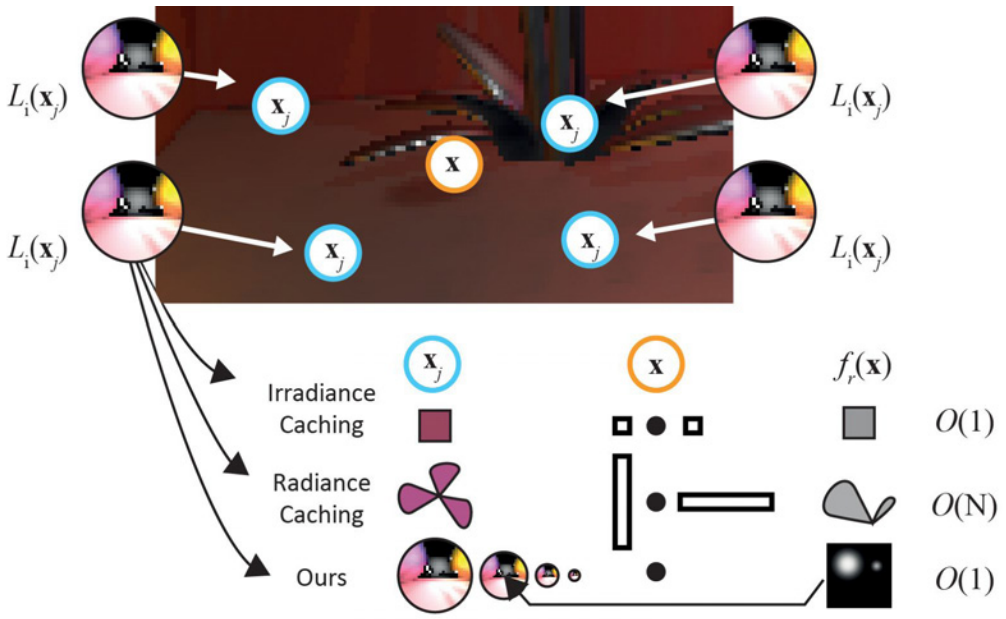

Figure 3.2: Irradiance Caching, Radiance Caching and our approach all start *(top)* from a set of cache items *(blue circles)* that are interpolated for an image pixel *(orange circle)*. Differences lie in the way a single cache item is evaluated at a location *(bottom)*. Irradiance Caching *(bottom, 1st row)* turns the incoming radiance into a scalar and interpolates only this value. Radiance Caching *(bottom, 2nd row)* projects incoming radiance into the Spherical Harmonic (SH) basis, for every pixel converts the BRDF into SH and performs a dot-product between two dense vectors per pixel. Our approach *(bottom, 3rd row)* performs a pre-convolution per cache item using MIP mapping which is queried using a BRDF- and normal-dependent lookup.

Per-cache Item Computation

Definition First, recall the definition of pre-convolution [Heidrich and Seidel 1999]: Assuming physically-plausible Phong [1975], the outgoing radiance can be decomposed into diffuse and specular outgoing radiance

$$L_\mathrm{o}(\mathbf{x},\omega_\mathrm{o}) = \rho_\mathrm{d}(\mathbf{x})D(\mathbf{x},n(\mathbf{x})) + \rho_\mathrm{s}(\mathbf{x})S(\mathbf{x},\omega_\mathrm{r},g(\mathbf{x})),$$

where $\rho_\mathrm{d}(\mathbf{x})$ is the diffuse albedo, $\rho_\mathrm{s}(\mathbf{x})$ the specular albedo, $g(\mathbf{x})$ the glossiness and $\omega_\mathrm{r} = 2\langle n(\mathbf{x}),\omega_\mathrm{o}\rangle n(\mathbf{x}) - \omega_\mathrm{o}$ is the reflection direction. The diffuse incident illumination (irradiance)

$$D(\mathbf{x},n) = \int L_\mathrm{i}(\mathbf{x},\omega_\mathrm{i})\langle n,\omega_\mathrm{i}\rangle^{+} d\omega_\mathrm{i},$$

for a fixed position $\mathbf{x}$ only depends on the normal n and can be pre-convolved. The specular incident illumination

$$S(\mathbf{x},\omega_\mathrm{r},g) = \int L_\mathrm{i}(\mathbf{x},\omega_\mathrm{i})\langle \omega_\mathrm{r},\omega_\mathrm{i}\rangle^{g} d\omega_\mathrm{i},$$

for a fixed location $\mathbf{x}$, depends both on the outgoing direction ω_o and the glossiness g. Similar to diffuse, it can be pre-convolved for every outgoing direction ω_o and every glossiness g.

Projection and Discretization In practice the pre-convolution has to be performed onto discrete two-dimensional images that represent incoming radiance, irradiance and specular incident illumination for every cache item.

First, while $L_\mathrm{i}(\mathbf{x}_\mathrm{j}, \omega_\mathrm{i})$ was oriented in world space it is transformed into a local space aligned with the normal $n(\mathbf{x}_\mathrm{j})$ using a transformation matrix $\mathsf{R} \in \mathbb{R}^{3\times3}$. To map image locations to directions, we choose the bijective paraboloid [Heidrich and Seidel 1999] projection $p(y) : \mathbb{R}^2 \leftrightarrow \mathbb{S}^2_+$ between the unit circle and the hemisphere. The resulting solid angle for every pixel is similar and our implementation ignores the remaining variation. Second, the continuous directional function $L_\mathrm{i}(\mathbf{x}_\mathrm{j}, \omega_\mathrm{i})$ is discretized into an image $\bar{L}_\mathrm{i}(\mathbf{x}_\mathrm{j}, y) = L_\mathrm{i}(\mathbf{x}_\mathrm{j}, \mathsf{R} \cdot p(y))$ of resolution $\sqrt{n_\mathrm{d}} \times \sqrt{n_\mathrm{d}}$. Only dealing with solid surfaces, we restrict $\bar{L}_\mathrm{i}$ to the upper hemisphere. This avoids the difficulties for filtering near the equator that challenges omnidirectional maps. In the case of PBGI or IR, $\bar{L}_\mathrm{i}$ is already produced efficiently in this regular discretized form.

Pre-convolution The diffuse pre-convolution $\bar{D}(\mathbf{x}_\mathrm{j})$, of $\bar{L}_\mathrm{i}(\mathbf{x}_\mathrm{j})$ is stored in a 4×4 pixel image. A 4×4 resolution is similar to the nine SH coefficients which were found to be sufficient for diffuse irradiance [Ramamoorthi and Hanrahan 2001], i. e., using a higher resolution we did not observe an improvement in accuracy. The specular pre-convolution $\bar{S}(\mathbf{x}_\mathrm{j})$, of $\bar{L}_\mathrm{i}(\mathbf{x}_\mathrm{j})$ corresponds to a MIP map of $\bar{L}_\mathrm{i}(\mathbf{x}_\mathrm{j})$. Every MIP level maps to a different glossiness: a highly glossy reflection (less blurred; more detailed) will look up a lower MIP level; a lower gloss reflection will look up the average of a large number of pixels [Kautz et al. 2000]. Note that the time cost per cache item for both pre-convolutions is *linear* (the same as the one of a single SH projection) and parallel over all directions and cache items. We used diffuse pre-convolution because its implementation is a single line of shader code when already using specular pre-convolution. It does not give any significant computational gain, as the dot product with constant, low-order SHs is efficient.

3.2.2 Per-pixel Computation

Here, instead of evaluating the dot product of n_d SH basis functions as done in Radiance Caching, a constant number i. e., two lookups in $\bar{D}(\mathbf{x}_\mathrm{j})$ and $\bar{S}(\mathbf{x}_\mathrm{j})$ are performed. First, the diffuse reflection is fetched from $\bar{D}(\mathbf{x}_\mathrm{j})$ using the per-pixel (i. e., bump-mapped) normal and multiplied with the per-pixel diffuse albedo. Second, the specular reflection is fetched from $\bar{S}(\mathbf{x}_\mathrm{j})$ using the reflected (i. e., again, bump-mapped) view direction from a MIP level corresponding to the per-pixel glossiness and multiplied with the specular albedo. Lafortune BRDFs, that are a sum of n_l simple lobes, can be supported using n_l lookups in $\bar{S}(\mathbf{x}_\mathrm{j})$ [Kautz et al. 2000].

Our method also scales well with spatially varying BRDFs with different glossiness per pixel. The per-pixel operation is a single MIP map-lookup which is highly efficient on all GPUs and scales well to high resolutions.

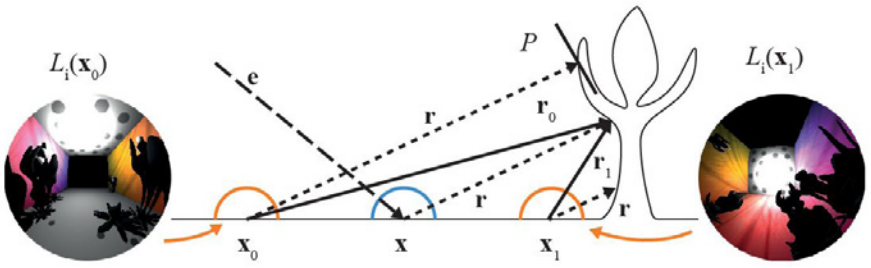

Figure 3.3: Recursive lookups based on distance impostors for Radiance Caching: To query direction $\mathbf{r}$, e. g., the reflected view direction $\mathbf{e}$ at location $\mathbf{x}$ when a cache item was created at $\mathbf{x}_0$ it is better to look up direction r_0. Similarly, to look up $\mathbf{r}$ in $\mathbf{x}_1$, an improved direction would be $\mathbf{r}_1$. The plane P is used to approximately improve the lookup directions. Please see Szirmay-Kalos et al. [2005]'s work for an in-depth explanation in their case of local reflection maps.

3.2.3 Recursive Lookups

The approach of Szirmay-Kalos et al. [Szirmay-Kalos et al. 2005] allows to perform more accurate queries of pre-convolved radiance cache items, as explained in Figure 3.3.

Different from their approach, we store distributions of positions over solid angle in a MIP map, that is used to approximate P. Recursive lookups achieve a goal similar to irradiance gradients [Ward and Heckbert 1992]: accounting for the change of incoming light over space between cache locations. Please see Section 3.3 for a comparison between recursion and no-recursion and the performance characteristics.

3.2.4 Implementation

Cache Generation We tested several implementations to generate caches: Instant Radiosity [Keller 1997], PBGI [Christensen 2008; Ritschel et. al. 2009a] and direct rasterization of all triangles without any approximation. PBGI directly produces $\bar{L}_\mathrm{i}$. For IR, each VPLs's contribution is splat additively into an initially black $\bar{L}_\mathrm{i}$, including shadow maps for visibility.s As we are not concerned with the quality or performance of the underlying cache generation approach, all images in this chapter use the simple direct rasterization approach. Note, that direct rasterization produces diffuse and specular shading when filling each cache item, i. e., caustics (Figure 3.5). A combination with Monte Carlo-raytracing to produce caches containing all light paths would certainly be possible in future work.

Pre-filtering All cache items are stored in a tiled layout in a large float-valued RGB-texture read using bilinear filtering. A typical resolution for this texture is 2048×2048 for a $4k$ cache with cache items of size 32×32 . MIP maps of these textures are created in hardware in 21.9 ms and require 32 MB of memory. Note, that the diffuse cache items only need a size of 4×4 requiring only 0.5 MB.

For specular pre-convolution, we perform MIP map down-sampling, in parallel for all cache items and in parallel over 4-tuples of directions in $\log_2(n_\mathrm{d})$ passes. For diffuse pre-convolution, we use a suitable higher MIP map-level from the specular step to avoid comput-

ing the product of all directions with the geometric term for each direction, e. g., only use 8×8 instead. The diffuse convolution is performed in parallel for all cache items and all directions.

While every direction ω on the upper hemisphere maps to an image location $y = p^{-1}(\omega)$, not every image location y maps to a direction $\omega = p(y)$. To avoid undefined pixels when reading low-resolution textures with bilinear filtering, we map undefined locations, i. e., outside the unit circle, where $|y| > 1$ to the closest defined location.

Cache Placement We place cache items according to a blue-noise distribution on the surface in world space. Distributions in image space, — regular ones found in most interactive GI systems, or adaptive ones found in offline rendering — are orthogonal to our approach.

Querying Caches using Splatting Our approach uses splatting of caches [Gautron et. al. 2004] to a deferred shading buffer that stores per-pixel position, normals, and reflectance (diffuse color, specular color, and Phong exponent). Our cache items affect pixels based on their spatial distance. We define a maximal spatial distance α and a maximal angle β. First, a point is drawn for every item. A geometry shader creates a quad that covers all pixels which are closer to the i-th cache location than α. For every pixel in this quad, the cache item has to be evaluated.

3.3 Results

Our main result is a comparison to Irradiance and Radiance Caching. In summary, our approach performs at a quality very similar to Radiance Caching, at a cost that is more similar to Irradiance Caching, i. e., simple interpolation of scalar light. Radiance caching is implemented by a per-pixel loop over all SH coeffcients stored in textures. We experimented with combining diffuse and specular Irradiance caching, but finally decided to remove it completely for the comparison, as excessive speckles appear when a cache happens to fall on a specular highlight. In Figure 3.6, Figure 3.7 and Figure 3.8, we show equal time-equal quality comparisons.

The plots in Figure 3.4 show, how our approach scales for the scenes in Figure 3.1 when the amount of directional detail is changed. In Figure 3.5, we show an example of glossy bounces and analyze the effect of recursive queries in Figure 3.10. Finally, we investigate the difference of our approach and a reference, where 1000 directions were sampled for every pixel in Figure 3.9.

3.4 Discussion

Drawbacks of this work include the typical one third-increase in required memory for a MIP map [Williams 1983] and the slightly suboptimal signal reproduction compared to the Fourier basis [Ramamoorthi and Hanrahan 2001]. Note, that both SH and our approach need to store the cache coefficients, which can be a problem when scaling to scenes that

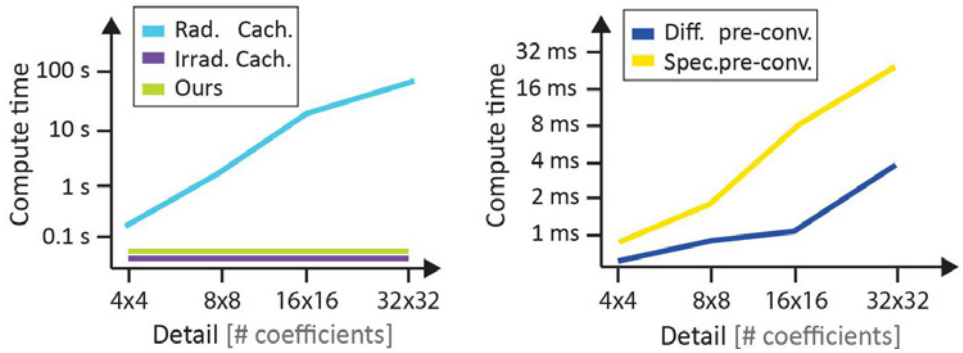

Figure 3.4: Computation time relative to lighting detail (number of SH coefficients, resp. number of pixels). *Left*: Per-pixel cost of Radiance Caching scales linearly with detail, whereas our approach remains almost constant, similar to Irradiance Caching. *Right*: The cost of the pre-convolution is small, and scales linearly with the cache resolution.

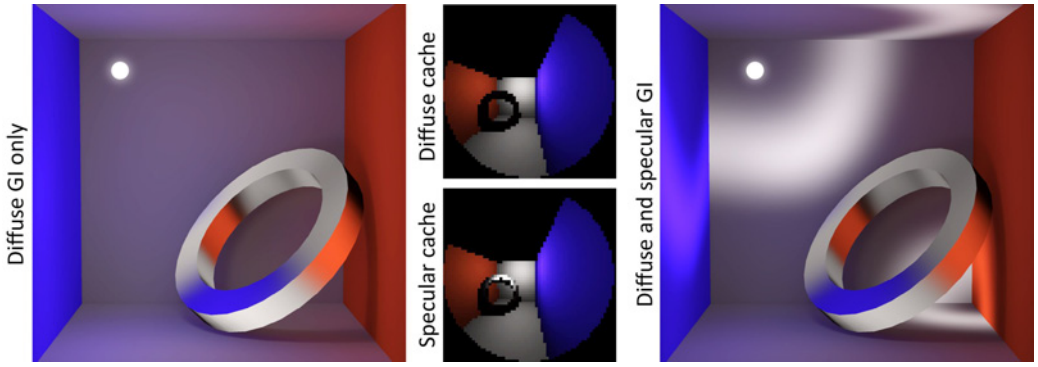

Figure 3.5: A caustic LSDE path (different from the LDSE paths shown in the rest of the chapter) using our approach (58 ms, 2048×2048, 4 k caches, each 64×64). Diffuse caches would ignore highlights *(left)*, highlights are present in specular caches (*right*). Please note, that the high contrast of caustics is difficult for Radiance Caching where it causes SH ringing.

require a large number of caches. Nevertheless, nothing prevents RC or our approach from creating cache items incrementally if the full cache does not fit into memory.

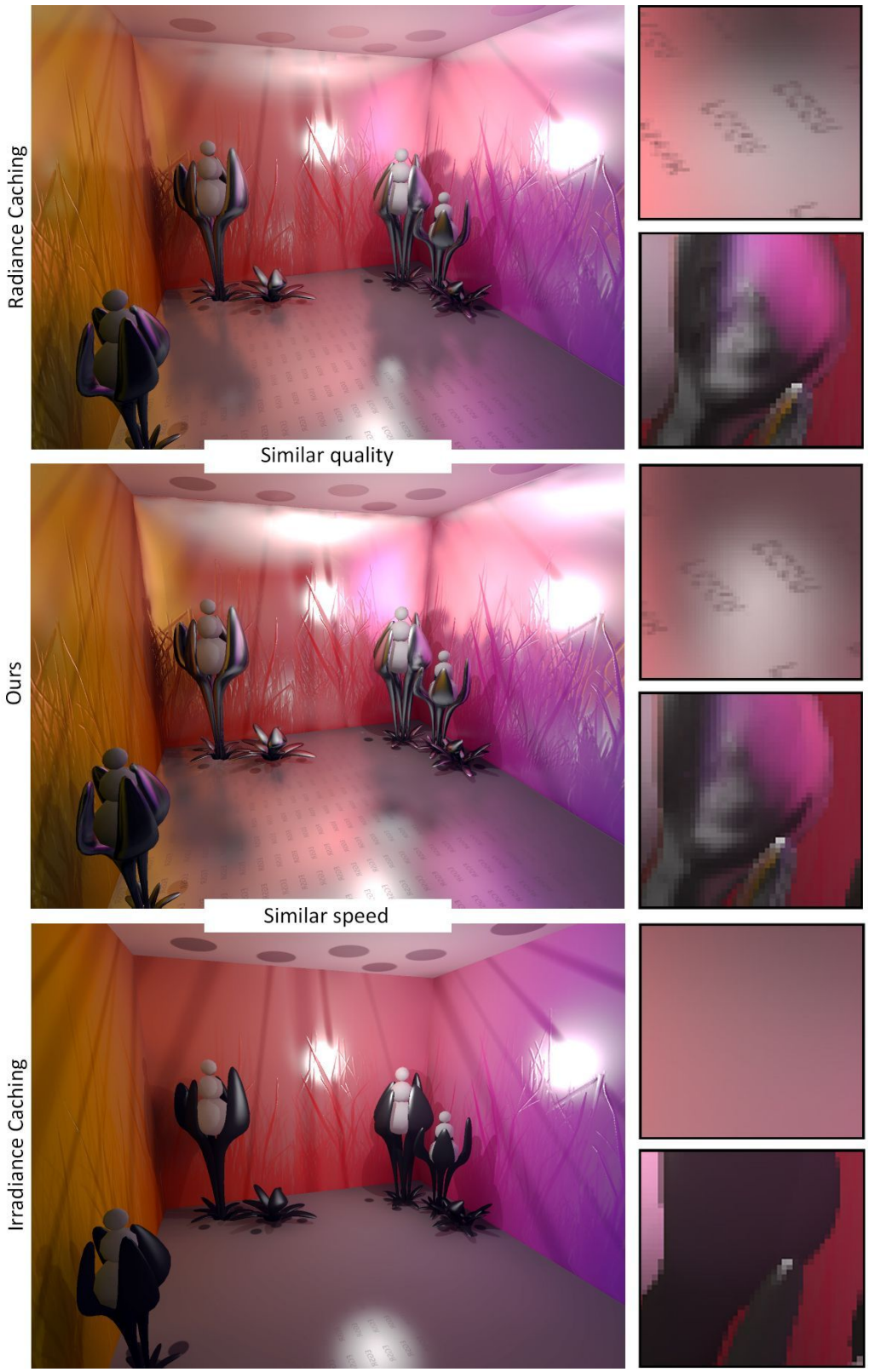

Figure 3.6: Comparison between Radiance Caching (RC) *(top)*, Our Approach *(middle)* and Irradiance Caching (IC) *(bottom)*. At speed similar to IC (≈33 ms), our approach (≈50 ms) achieves quality very similar to RC (>1 s). RC and our approach both produce fine shading details due to bump mapping (side walls) or spatially-varying gloss (logo on the ground) that are lacking in Irradiance Caching.

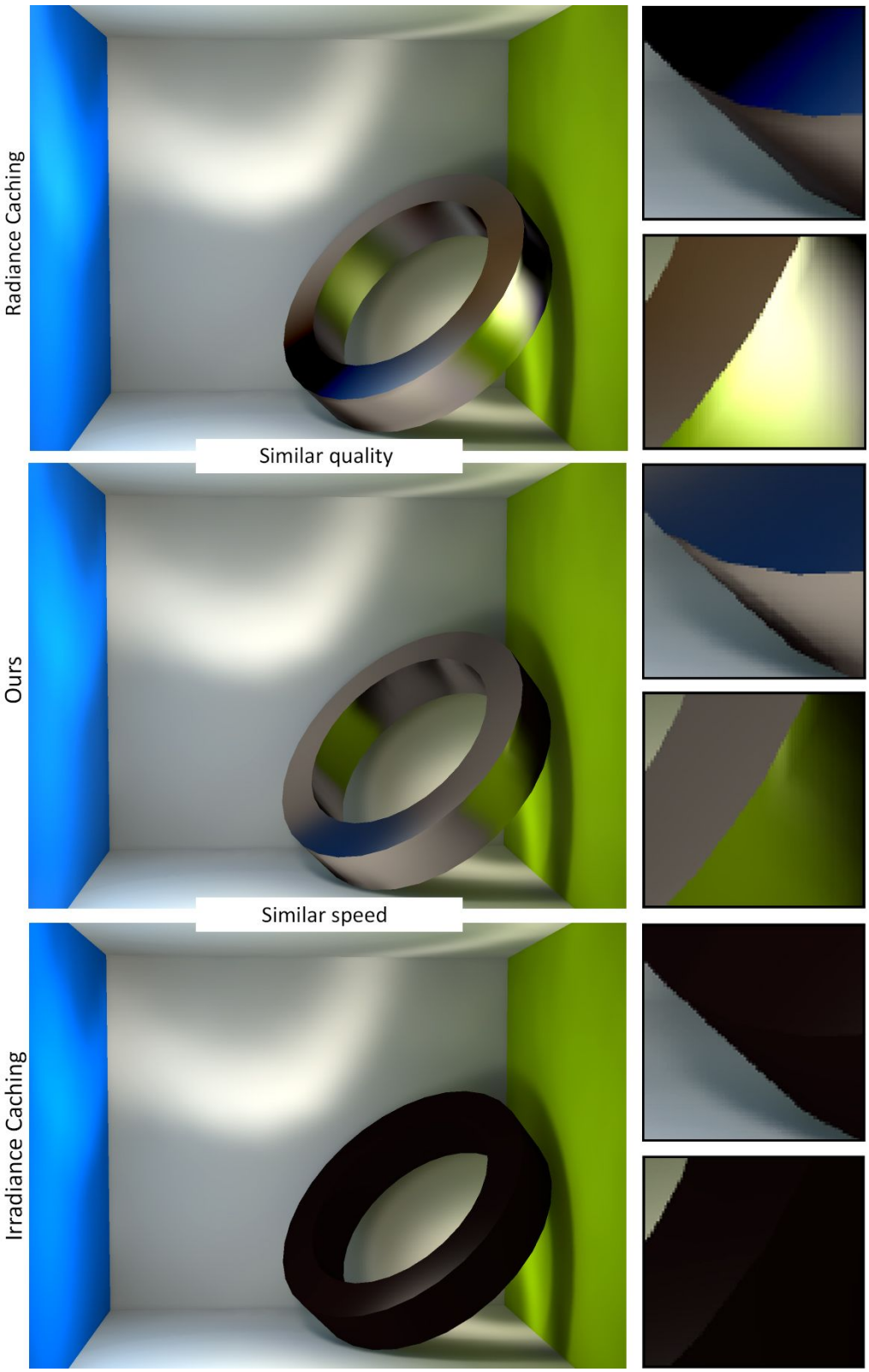

Figure 3.7: Comparison between Radiance Caching (RC) *(top)*, Our Approach *(middle)* and Irradiance Caching (IC) *(bottom)*. At speed similar to IC (≈33 ms), our approach (≈50 ms) achieves quality very similar to RC (>1 s). RC produces negative *(1st inset, right)* and overshot *(2nd inset, right)* light due to SH ringing caused by the high contrast of the caustic. While IC does not capture the specular bounce, our approach reproduces all effects.

Figure 3.8: Comparison between Radiance Caching (RC) *(top)*, Our Approach *(middle)* and Irradiance Caching (IC) *(bottom)*. At speed similar to IC (≈33 ms), our approach (≈50 ms) achieves quality very similar to RC (>1 s). RC and our approach show a similar reprodution of specular shading details for a complex scene, such as the pear *(1st inset, right)* or the metallic skull *(2nd inset, right)* which are missing for IC.

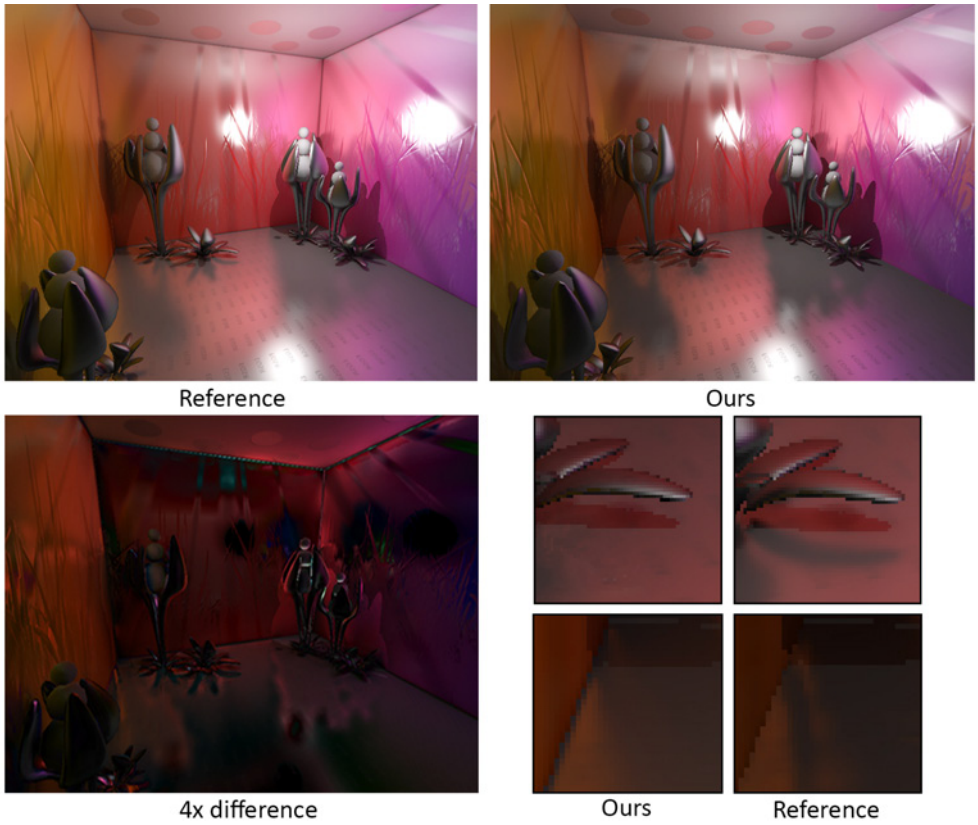

Figure 3.9: Comparison between a reference *(left)* and our solution *(middle)*. Next are the $\times 4$ difference between the images *(right, bottom)* and the insets *(right, top)*.

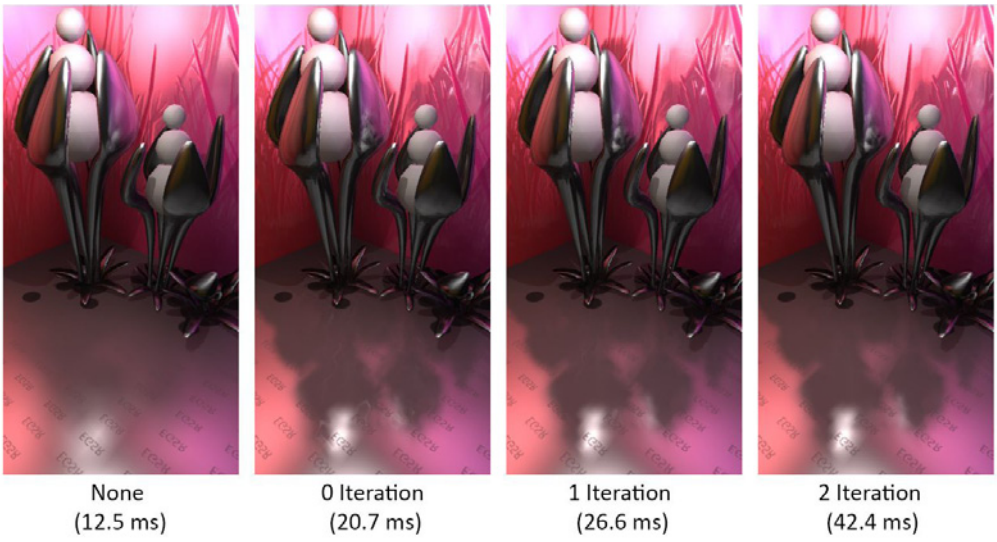

Figure 3.10: None, as well as different numbers of recursion, *(left to right)* for an glossy ground plane. It can be seen, that after no or one iterations, the lookups for the purpose of glossy reflections in Radiance Caching are sufficiently accurate. Note, that the overlap between splats smoothes out errors. Consequently, the error appears as blur, instead of noise or banding. A successful lookup, such as ours, will result in sharp reflections.

4

Surface Light Field Manipulation in 3D Scenes

4.1 Introduction

The appearance of materials is an important cue for human understanding of its surrounding, beginning with distinguishing between edible and rotten food for early humans, and reaching up to today's intuition about the price of a car by looking at the lacquer. Therefore, material appearance is highly important in many computer graphics applications, ranging from product visualization, to feature films or computer games and its proper depiction. Acquisition and manipulation can be the key to achieving a desired goal. Recent studies about material design [Kerr and Pellacini 2010] point out the difficulty of material design, even under controlled and simple illumination and without spatial variation. However, in many use-cases, e. g., feature film production, it is important how spatially-varying materials appear in combination, in complex geometrical arrangements, with occlusions, and under complex (global) illumination.

We propose a system to design materials under such settings where an artist performs interactive appearance manipulation by painting strokes onto a "3D+2D canvas" (the surface light field *(SLF)* of the scene) and the system finds the best reflectance to produce the desired appearance (Figure 4.2).

Our work has two key motivations: Reflectance manipulation is too tedious to be effective, and direct SLF manipulation is too general to be intuitive. Questions like "How and where do I have to change glossiness of this ring over here to get a caustic that just is bright enough to be visible and blurry enough over there?" are avoided using our system: The user paints the bright spot and the system will find the required change. Second, direct manipulation of a SLF is too tedious, has too many degrees of freedom, in particular when the viewer moves and is not well supported by most rendering systems. Consider a user painting a white dot onto a red sphere, which could either mean a highlight, or a white, diffuse dot. If the intention was to draw a highlight, the user would need to move the camera, and draw the highlight in a new position. This would need to continue for many, if not all possible views, a prohibitively tedious process. Our system seeks to understand if the white dot is

Figure 4.1: An input 3D scene *(a)* is painted by view-dependent strokes manipulating its surface light field. *(b)* A red stroke made on the wagon indicating a change of diffuse color. *(c)* A white stroke changing highlight shapes on the plane. *(d)* Painting a white highlight on the wheel. Our system finds the smallest change of shading parameters to produce a light field matching the strokes *(e)*.

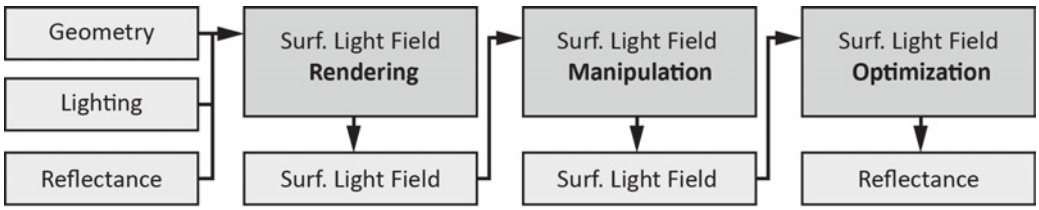

Figure 4.2: Overview of our approach *(Please see text)*.

meant to be a highlight, and if yes, to change the reflectance of the red sphere in a way that generalizes to different viewpoints. Different from previous $\mathcal{L}_2$-solutions in graphics and vision that match observations (i. e., a sampling of the SLF) with a reflectance model in a least-squares sense, our $\mathcal{L}_0$-approach will find sparse changes of reflectance, i. e., it will prefer changes in only one parameter that reflects the users intention.

4.2 Problem Statement

Section 2.3.1 and Section 2.3.2 discuss some background on rendering that will be used in this chapter. This section introduces the required notation and states the continuous problem that has to be solved. Table 4.2 and Table 4.1 ensemble the sets, constants and general symbols used in the chapter.

Valid Surface Light Fields Let $\mathcal{R}$ be a set of reflectance fields, e. g., all physically-plausible BRDFs $\mathcal{R}_{\text{Physical}}$, or all BRDFs a rendering system supports, e. g., Phong $\mathcal{R}_{\text{Phong}}$. We will call a SLF "valid" in respect to a surface $\mathcal{M}$, an initial emissive lighting L_e and a set of BRDFs $\mathcal{R}$ if it is the solution of the RE for *any* reflectance $R \in \mathcal{R}$. Our approach allows the user to manipulate (Section 4.3) a valid SLF L_o to become a new, potentially invalid, SLF L_o^m. Our system will seek to find a new reflectance $R^\text{m} \in \mathcal{R}$ resulting in a SLF after n_b bounces that is most similar to L_o^m. In a least squares sense, this is

$$R^\text{m} := \arg\min_{\hat{R}\in\mathcal{R}} ||\mathbf{T}^{n_\text{b}}(\hat{R})\, L_\text{e} - L_\text{o}^\text{m}||^2, \tag{4.1}$$

using an SLF norm

$$||L|| := \sqrt{\int_\mathcal{M}\int_{\mathbb{S}^2} L(\mathbf{x},\omega)^2 \mathrm{d}\omega\mathrm{d}\mathbf{x}}.$$

4.3 Surface Light Field Manipulations

Manipulation is performed by selecting a tool, and painting strokes into the scene. A *tool* (Section 4.3.1) changes the SLF (spatially under the stroke; directionally from the view it is currently seen) into a new SLF. Tools can be used either direct, or indirectly (Section 4.3.2). After the system has computed a change of reflectance it gets propagated to all surface locations that are similar to the locations under the stroke (Section 4.3.3).

4.3.1 Tools

A *tool* is a manipulation operator $\mathbf{M} \in \mathcal{M}\times\mathbb{S}^2 \to \mathcal{M}\times\mathbb{S}^2$ that maps a SLF into an edited SLF. Common image manipulations like "replace", "clone", "brighten", "darken," "blur", "sharpen" etc. can be used here (Figure 4.3). A "fixate" disallows changes to the SLF for some regions.

Symbol	Meaning
$\mathbf{x}, \mathbf{y}$	Location
$\mathbf{n}$	Normal
$n(\mathbf{x})$	Normal at position $\mathbf{x}$
$\omega, \omega_\mathrm{i}, \omega_\mathrm{o}$	Direction; incoming, outgoing
$R(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o})$	Spatially varying BRDF (reflectance)
$L_\mathrm{i}(\mathbf{x}, \omega)/L_\mathrm{o}(\mathbf{x}, \omega)$	Incident/Outgoing radiance
$\mathbf{G}$	Geometry operator
$\mathbf{K}(R)$	Reflection operator
$\mathbf{T}^i(R)$	i-bounce transport operator
$L_\mathrm{e}/L_\mathrm{o}^\mathrm{m}$	Emissive/Manipulated SLF
R^m	New reflectance
$\mathbf{M}$	Manipulation operator
$m(\mathbf{x}, \omega)$	Manipulation stroke function
$l_\mathrm{o}(\mathbf{x}, \cdot)$	SLF for position $\mathbf{x}$
$s(\mathbf{x}, \mathbf{y})$	Similarity
$\Delta_\mathrm{R}(\mathbf{x}, \mathbf{y})$	BRDF difference function
$\Delta_\mathrm{r}(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o})$	Change of reflectance
$R_\mathrm{f}(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o})$	Final reflectance
$\mathbf{p}_i$	Element i
$\omega_{i,j}$	Direction j of element i
$\mathbf{l}_\mathrm{e}/\mathbf{l}_\mathrm{o}^\mathrm{m}$	Discrete emissive/manipulated SLF
K_d	Diffuse reflection matrix
$\mathsf{K}_\mathrm{s}(\mathbf{f}_\mathrm{g})$	Specular reflection matrix
$\mathbf{r}$	Shading model parameter vector
$\mathbf{r}^\mathrm{m}$	New discrete reflectance
$\mathbf{r}_\mathrm{d}/\mathbf{r}_\mathrm{s}/\mathbf{r}_\mathrm{g}$	Diffuse/Specular/Glossy components of the parameter vector
$\mathbf{r}_{\mathrm{d},i}/\mathbf{r}_{\mathrm{s},i}/\mathbf{r}_{\mathrm{g},i}$	Diffuse/Specular/Glossy components of element i
$\perp(\omega, \mathbf{n})$	Reflection function
G	Discrete geometry matrix
$\mathsf{K}(\mathbf{f}_\mathrm{r})$	Discrete reflection matrix
$\mathsf{T}^i(\mathbf{f}_\mathrm{r})$	Discrete i-bounce transport matrix
$\mathrm{rep}(\mathbf{v}, n)$	Creates a vector were each entry of $\mathbf{v}$ is repeated n times
$\mathrm{diag}(\mathbf{v})$	Creates a diagonal matrix out of $\mathbf{v}$
W	Diagonal weighting matrix
$\mathbf{m}$	stroke vector $\in \{0,1\}^{n_\mathrm{p} n_\mathrm{d}}$
$\hat{\mathbf{x}}$	Least-squares fit for diffuse and specular shading parameters
$\alpha(\mathfrak{X})$	Selects the element with smallest residual magnitude from a set
$\mathcal{L}_2$	Least-squared optimization
$\mathcal{L}_0$	Zero-norm optimization
$\hat{\mathbf{x}}_\mathrm{d}/\hat{\mathbf{x}}_\mathrm{s}$	Least-squares fit for diffuse/specular if specular/diffuse is fixed
$\mathbf{h}$	index vector $\in \mathbb{N}^{n_\mathrm{p} n_\mathrm{d}}$

Table 4.1: Other symbols table

Symbol	Meaning
$\mathcal{M}$	Surface domain
$\mathbb{S}^2$	Spherical surface domain
$\mathcal{R}$	Set of reflectance fields
P	Set of locations of elements
$\mathfrak{R}$	Set of discrete reflectance fields
$\mathbf{g}$	Set of glossiness values for optimization. $\lvert\mathbf{g}\rvert = n_\mathrm{c}$
$\mathfrak{X}$	Set of solutions for different glossiness values
n_p	Number of elements
n_d	Number of directional bins
n_s	Dimension of shading model parameter vector
$\sigma_\mathrm{s}, \sigma_\mathrm{n}, \sigma_\mathrm{r}$	weight for position, normal, reflectance
n_c	Number of glossiness values for optimization
n_b	Number of bounces
σ_m	Manipulation tool weight
g_i	Glossiness value i used for search

Table 4.2: Symbol table for sets and constants

Figure 4.3: Original SLFs L_o *(Top)* and the manipulated one $\mathbf{M}L_\mathrm{o}$ *(Bottom)*. The first three change the highlight size. The next two change its strength. The following five change the diffuse color at the same time. The last two show more complex examples, under non-point lighting.

A manipulation *stroke* is a function $m(\mathbf{x},\omega) \in \mathcal{M} \times \mathbb{S}^2 \to \{0,1\}$, which is 1 if the surface at $\mathbf{x}$ seen from direction ω is affected by the tool and 0 otherwise. The new SLF is then given by $L_\mathrm{o}^\mathrm{m} := L_\mathrm{o} + m \cdot (\mathbf{M}L_\mathrm{o} - L_\mathrm{o})$.

4.3.2 Direct and Indirect Mode

All tools can be used either in "direct" or "indirect" mode. In direct mode, they affect the stroke and all similar regions, as defined above. In indirect mode, they do not affect the area under the stroke, but those areas, that contribute radiance to the stroke, such that $L_\mathrm{o}^\mathrm{m} := L_\mathrm{o} + \mathbf{T}(R)\,(m \cdot (\mathbf{M}L_\mathrm{o} - L_\mathrm{o}))$ (Figure 4.4).

For the example of a mirror object, $\mathbf{T}$ "copies" into every outgoing direction the value incoming in its mirrored direction, resulting in a "draw-across-the-mirror" behavior. The same indirection works for glossy and diffuse appearance. The indirect mode, allows to keep reflectance constant at one location, but changes appearance by changing reflectance in another location of the scene. An example of indirect painting is to manipulate the color of indirect lighting. The system will change the reflectance, of the object producing the indirect light. Another example application is to paint a bright caustic-shaped stroke next to a diffuse ring and the system detects that the ring should be turned metallic to achieve a caustic. Note, that the reflectance at the location of the caustic is not changed, but at the

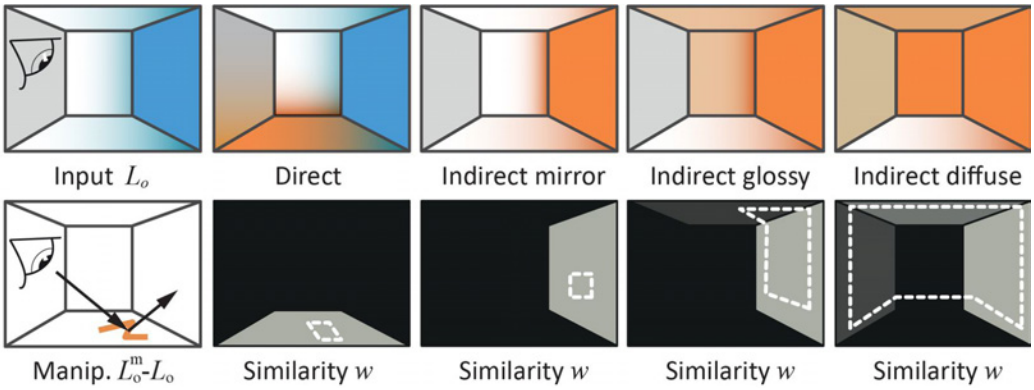

Figure 4.4: *(From left to right)*: A user viewing from the left paints an orange stroke onto the ground *(a)*. Direct mode *(b)*: The stroke and everything similar – in this case the entire ground – changes reflectance. Indirect mode: When the ground is a mirror *(c)*, the reflected location changes its reflectance to match the stroke. A glossy *(d)* or diffuse *(e)* ground causes the change to be distributed over many locations.

location of the object producing the caustic (Figure 4.5 and Figure 4.18).

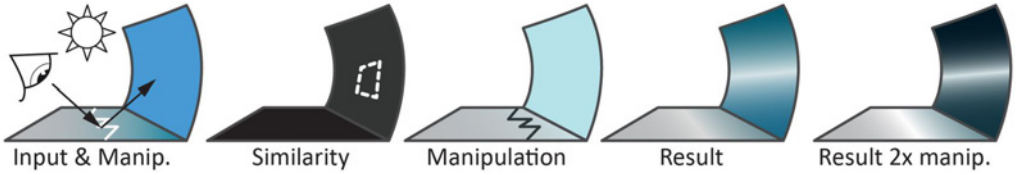

Figure 4.5: Caustic example *(left to right)*. *(a)* A user paints an indirect white stroke in the vicinity of a diffuse blue wall. *(b)* The resulting similarity. *(c)* A second stroke disambiguates the manipulation and the diffuse blue wall becomes specular. *(d)* Repeating the change makes the initially diffuse wall increasingly metallic and the caustic becomes more pronounced.

4.3.3 Edit Propagation

After the stroke has finished, the system computes a change of appearance that is propagated to all similar locations. The similarity $s(\mathbf{x},\mathbf{y}) \in \mathcal{M}^2 \to \mathbb{R}^+$ between a pair of points $\mathbf{x}$ and $\mathbf{y}$ on the manifold is the sum of the weighted distance in position, normal, or reflectance

$$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_\mathrm{s}}{||\mathbf{x}-\mathbf{y}||} + \frac{\sigma_\mathrm{n}\,\langle n(\mathbf{x}), n(\mathbf{y})\rangle}{1-\langle n(\mathbf{x}), n(\mathbf{y})\rangle} + \frac{\sigma_\mathrm{r}}{\Delta_\mathrm{R}(\mathbf{x},\mathbf{y})}$$

were the $\sigma_{\{\mathrm{s,n,r}\}}$ are the weights and Δ_R is a BRDF difference function (Figure 4.6).

The change of reflectance $\Delta_\mathrm{r} := R^\mathrm{m} - R \in \mathcal{M} \times \mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{R}^+$ is propagated from each point to all other points, proportional to similarity, resulting in the final reflectance

$$R_\mathrm{f}(\mathbf{x},\cdot,\cdot) := R(\mathbf{x},\cdot,\cdot) + \int_\mathcal{M} s(\mathbf{x},\mathbf{y})\Delta_\mathrm{r}(\mathbf{y},\cdot,\cdot)\mathrm{d}\mathbf{y} \,/\, \int_\mathcal{M} s(\mathbf{x},\mathbf{y})\mathrm{d}\mathbf{y}.$$

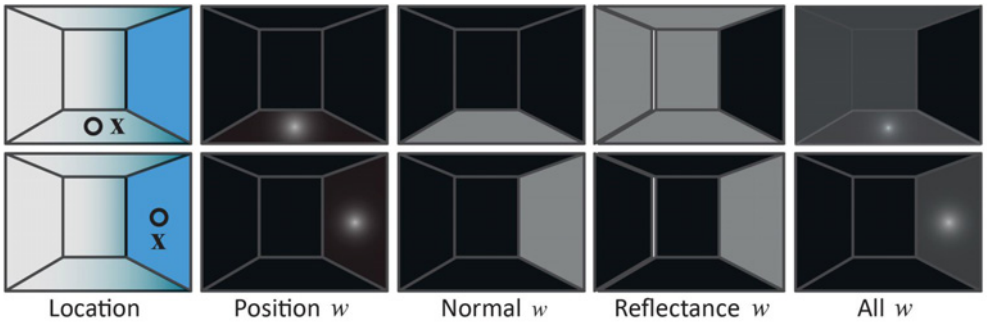

Figure 4.6: Similarity of two different locations $\mathbf{x}$ *(Rows)* and other locations $\mathbf{y}$. Different components *(Columns, left to right)*: Position, normal, reflectance and a combination of all.

4.4 Discretization

This section will describe the discretization of the domain used and how different forms of optimizations can be performed in practice. Surface fields will be discretized into finite spatio-directional elements stored in vectors (Section 4.4.1), and operators will take the form of matrices or matrix-valued functions (Section 4.4.2), similar to non-diffuse radiosity [Immel, Cohen and Greenberg 1986]. Using these entities, the discrete reflectance that best matches the users manipulation is found in a non-linear optimization procedure (Section 4.4.3)

4.4.1 Discrete Domain

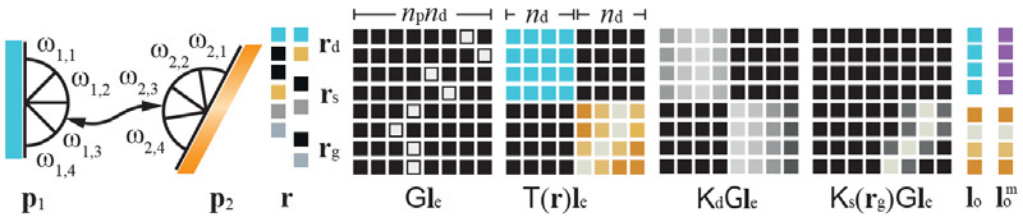

Figure 4.7: Discretization example *(Left to right):* Two elements $\mathbf{p}_1$ and $\mathbf{p}_2$ (blue diffuse; orange specular) and their directional bins $\omega_{1,1},\ldots,\omega_{2,4}$. The reflectance $\mathbf{r}$ and its Phong components $\mathbf{r}_d,\mathbf{r}_s,\mathbf{r}_g$. The operator G has one non-zero entry per row that links every element to another visible element and its bin *(arrow)*. The transport operator $\mathbf{T}(\mathbf{r})$ giving different weights to different incoming directions. Assuming Phong, it can be decomposed into $\mathbf{K}_d$ and $\mathbf{K}_s(\mathbf{r}_g)$. The SLF $\mathbf{l}_o$ which the user manipulates into $\mathbf{l}_o^m$.

The problem is discretized into point elements to obtain a solution numerically (Figure 4.7). The scene surface $\mathcal{M}$ is assumed to be static. It is spatially discretized into a set of n_p (typically thousands of) elements with locations $P := \{\mathbf{p}_1,\ldots,\mathbf{p}_{n_p}\} \in \mathbb{R}^{3\times n_p}$. Directionally, each element is discretized into n_d (typically $1024 = 32\times 32$) directional bins with directions $\omega_{i,1},\ldots,\omega_{i,n_d}$. Sampling into point elements decouples the problem from the particular geometrical representation of $\mathcal{M}$. The original SLF is represented as a vector $\mathbf{l}_o \in \mathbb{R}^{n_p n_d}$, the desired SLF as $\mathbf{l}_o^m \in \mathbb{R}^{n_p n_d}$ and the initial emissive light field as a vector $\mathbf{l}_e \in \mathbb{R}^{n_p n_d}$. Every elements's reflectance can be parametrized by a n_s-dimensional shading

model parameter vector. Over all elements this results in $\mathbf{r} \in \mathfrak{R}$, where $\mathfrak{R} \subseteq \mathbb{R}^{n_\mathrm{p} n_\mathrm{s}}$ is the set of potential BRDFs, a discrete version of $\mathcal{R}$. We will discuss the case of $n_\mathrm{s} = 3$ for Phong diffuse, specular and glossiness and write shorthand $\mathbf{r}_\mathrm{d} \in \mathbb{R}^{n_\mathrm{p}}$ for the diffuse, $\mathbf{r}_\mathrm{s} \in \mathbb{R}^{n_\mathrm{p}}$ for the specular and $\mathbf{r}_\mathrm{g} \in \mathbb{R}^{n_\mathrm{p}}$ for the glossy components of the parameter vector.

4.4.2 Discrete Operators

For more convenient notation, let $u_\mathrm{p} = u/n_\mathrm{d}$ the spatial index of row index u and $v_\mathrm{d} = v \bmod n_\mathrm{d}$ the directional index of column index v. Please, see Figure 4.7 for the interleaving of indices.

Geometry The discrete version of the geometry operator $\mathbf{G}$ is the matrix $\mathsf{G} \in \mathbb{R}^{(n_\mathrm{p} n_\mathrm{d}) \times (n_\mathrm{p} n_\mathrm{d})}$. For solid surfaces, G is zero in every column of row u and 1 in the column with the index of the element visible from $\mathbf{p}_{u_\mathrm{p}}$ in direction $\omega_{u_\mathrm{p}, v_\mathrm{d}}$.

Reflection The discrete reflection operator is a matrix-valued function $\mathsf{K}(\mathbf{r}) \in \mathbb{R}^{(n_\mathrm{p} n_\mathrm{d}) \times (n_\mathrm{p} n_\mathrm{d})}$. Assuming a reflectance model like Phong, it can be written as

$$\mathsf{K}(\mathbf{r}) = \mathrm{diag}(\mathrm{rep}(\mathbf{r}_\mathrm{d}, n_\mathrm{d}))\mathsf{K}_\mathrm{d} + \mathrm{diag}(\mathrm{rep}(\mathbf{r}_\mathrm{s}, n_\mathrm{d}))\mathsf{K}_\mathrm{s}(\mathbf{r}_\mathrm{g}),$$

the sum of a diffuse reflection matrix, multiplied by diffuse reflection component $\mathbf{r}_\mathrm{d}$ and a specular reflection matrix, depending on glossiness $\mathbf{r}_\mathrm{g}$, multiplied by specular reflection component $\mathbf{r}_\mathrm{s}$. Both the diffuse and specular matrix are of the same size as K itself. Diffuse reflection maps incoming light into a directionally-invariant constant exitant value

$$\mathsf{K}_{\mathrm{d},u,v} := \langle n(\mathbf{p}_{u_\mathrm{p}}), \omega_{u_\mathrm{p}, v_\mathrm{d}} \rangle^+.$$

Specular reflection maps incoming light into directionally-dependent exitant values that are both mirrored and blurred proportional to glossiness

$$\mathsf{K}_{\mathrm{s},u,v}(\mathbf{r}_\mathrm{g}) := \langle \perp(\omega_{u_\mathrm{p}, u \bmod n_\mathrm{d}}, n(\mathbf{p}_{u_\mathrm{p}})), \omega_{u_\mathrm{p}, v_\mathrm{d}} \rangle^{\mathbf{r}_{\mathrm{g},u_\mathrm{p}}},$$

where $\perp(\omega, \mathbf{n}) = \omega - 2\,\mathbf{n}\langle \omega, \mathbf{n} \rangle$ reflects direction ω at the normal $\mathbf{n}$. Finally, the discrete i-bounce transport matrix is

$$\mathsf{T}^i(\mathbf{r}) = \sum_{j=1}^{i} (\mathsf{K}(\mathbf{r})\mathsf{G})^{j-1} \quad \text{with} \quad \mathsf{T}^0 = \mathsf{I}.$$

4.4.3 Discrete Minimization

The minimization seeks to find a discrete solution for the problem stated continuously in Equation 4.1 by solving

$$\mathbf{r}^\mathrm{m} := \underset{\hat{\mathbf{r}} \in \mathfrak{R}}{\arg\min} ||\mathsf{W}(\mathsf{T}^{n_\mathrm{b}}(\hat{\mathbf{r}})\,\mathbf{l}_\mathrm{e} - \mathbf{l}_\mathrm{o}^\mathrm{m})||^2, \tag{4.2}$$

a reflectance, that after n_b bounces, given the initial emissive lighting $\mathbf{l}_e$, produces a SLF most similar to the desired $\mathbf{l}_o^m$. The diagonal matrix $\mathsf{W} = \mathsf{I} + \sigma_m \operatorname{diag}(\mathbf{m})$ is created from the stroke vector $\mathbf{m} \in \{0,1\}^{n_p n_d}$ (a discrete version of m) and a fixed and tool-dependent constant σ_m.

The problem is *non-linear* because the matrix T depends on the Phong glossines in $\mathbf{g}$ in a non-linear way. Furthermore, it is a *constrained* problem, as $\mathfrak{R}$ is only a subset of possible reflectance values that are valid, e. g., energy-preserving.

One-bounce Weighted Least-square Solution For simplicity, we will for now only consider one bounce. In this case, the reflectance for every element i can be found independently of all others. Still, the dependency of T on $\mathbf{r}_g$ poses a non-linear problem, but with only one remaining non-linear degree of freedom: glossiness. Therefore, our solution iterates over n_c (typically $n_c = 20$) fixed and perceptually uniform [Pellacini et. al. 2000] glossiness values $g_0, \ldots, g_{n_c}$ and solves the linear problem of finding the best specular and diffuse reflectance for each (Figure 4.8).

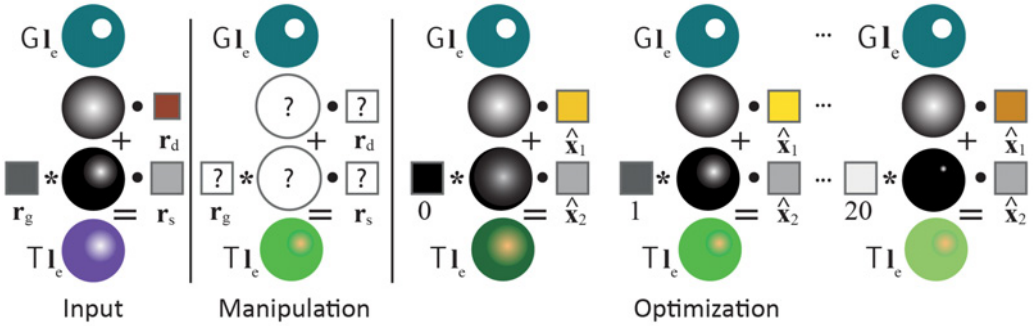

Figure 4.8: Our fitting. In every column, from top to bottom are the given lighting *(1st row)*, diffuse component *(2nd row)*, glossiness *(left)* and specular *(left)* component *(3rd row)*, and the SLF *(4th row)*. Starting from a given lighting *(1st col., 1st row)* and desired SLF *(2nd cols, 4th row)*, the optimal reflectance is found by enumerating different glossiness levels *(left, 3rd row, 3rd to 5th col.)*, using a linear solution for the diffuse and specular reflectance at each.

In the remainder of this section we will introduce "local" identifiers $(\mathbf{x}, \mathbf{b}, \mathsf{W}, \mathsf{A}, \hat{\mathbf{x}}, \mathfrak{X})$ with a dependency on i and j which is omitted in the notation for brevity. For element i and glossiness j, finding $\mathbf{r}_d^{(i,j)}$ and $\mathbf{r}_s^{(i,j)}$ is a linear problem $\mathsf{W}\mathsf{A}\mathbf{x} = \mathsf{W}\mathbf{b}$. Here $\mathbf{x}$ is the unknown diffuse and specular reflection coefficient of the current element and the current glossiness, $\mathbf{b} = (\mathbf{l}^m_{o,(i+0)\cdot n_d}, \ldots, \mathbf{l}^m_{o,(i+1)\cdot n_d})^\mathsf{T}$ is the desired SLF and

$$\mathsf{A} = \begin{pmatrix} (\mathsf{K}_d \mathsf{G} \mathbf{l}_e)_{i \cdot n_d} & (\mathsf{K}_s(g_j) \mathsf{G} \mathbf{l}_e)_{i \cdot n_d} \\ \vdots & \vdots \\ (\mathsf{K}_d \mathsf{G} \mathbf{l}_e)_{i \cdot n_d + n_d} & (\mathsf{K}_s(g_j) \mathsf{G} \mathbf{l}_e)_{i \cdot n_d + n_d} \end{pmatrix}.$$

The first column in A is the reflected diffuse light, the second column the reflected specular light for the current glossiness.

Let $\mathsf{C} \in \mathbb{R}^{2\times 2}$ be $\mathsf{A}^\mathsf{T}\mathsf{W}\mathsf{A}$ and $\mathbf{d} \in \mathbb{R}^2$ be $\mathsf{A}^\mathsf{T}\mathsf{W}\mathbf{b}$. The closed-form solution for a least-squares fit then is

$$\hat{\mathbf{x}} = \left(\frac{\mathsf{C}_{2,2}\,\mathbf{d}_1 - \mathsf{C}_{1,2}\,\mathbf{d}_2}{\mathsf{C}_{1,1}\,\mathsf{C}_{2,2} - \mathsf{C}_{1,2}\,\mathsf{C}_{2,1}}, \frac{\mathsf{C}_{1,1}\,\mathbf{d}_2 - \mathsf{C}_{2,1}\,\mathbf{d}_1}{\mathsf{C}_{1,1}\,\mathsf{C}_{2,2} - \mathsf{C}_{1,2}\,\mathsf{C}_{2,1}} \right)^\mathsf{T} \tag{4.3}$$

Let $\mathfrak{X} := \{(\hat{\mathbf{x}}^{(1)}, g_1), \cdots, (\hat{\mathbf{x}}^{(n_c)}, g_{n_c})\}$ be the set of different solutions for different glossiness values. Finally, we pick $\mathbf{r}_i := \alpha(\mathfrak{X})$, where

$$\alpha(\mathfrak{X}) := \mathfrak{X}_k \text{ with } k = \underset{1 \leq k \leq |\mathfrak{X}|}{\arg\min} ||\mathbf{l}_o^m - \mathsf{T}(\mathfrak{X}_k)\,\mathbf{l}_e|| \tag{4.4}$$

is an operator to select the element from a set that produces the smallest residual magnitude.

The solution is constraint to $\hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2 \leq 1$ for energy-preserving BRDFs. We do not solve a constraint least-squares problem, but re-scale the solution to be energy-preserving, or to 0 if it is negative in any component. This might result in suboptimal solutions, but guarantees energy preservation.

One-bounce Sparse Solution While the $\mathcal{L}_2$ solution explained before changes all shading parameters to match to the desired SLF, we will now extend it to a sparse solution, which will aim to change only one or a few parameters inspired by $\mathcal{L}_0$ minimization in compressed sensing [Donoho 2006]. Let $||\mathbf{x}||_0 := \mathbb{R}^n \to \mathbb{N}$ be the count of non-zero entries in the vector $\mathbf{x}$. We are looking for a new reflectance $\mathbf{r}^m$, such that $||\mathbf{r}^m - \mathbf{r}||_0$ is n_p (the number of elements), or in other words, such that only one shading parameter is changed per element. At the same time, we want the solution to match the desired SLF. Please note, that we are not looking for sparse shading parameters (which is hardly meaningful), but for a sparse *change* of shading parameters. While $\mathcal{L}_0$ minimization is computational intractable for large system, our problem is small enough to enumerate over the sensible solutions. To this end, the solver explained before is extended to hold all shading parameters fixed except one, and solve for the remaining parameter in the least squares sense. The closed-form least-squares fit for holding diffuse or specular fixed, is

$$\hat{\mathbf{x}}_d = \left(\frac{\mathsf{C}_{1,1}(\mathbf{d}_1 - \mathsf{C}_{1,2}\mathbf{r}_s) + \mathsf{C}_{2,1}(\mathbf{d}_2 - \mathsf{C}_{2,2}\mathbf{r}_s)}{\mathsf{C}_{1,1}^2 + \mathsf{C}_{2,1}^2}, \mathbf{r}_s\right)^\mathsf{T} \tag{4.5}$$

and

$$\hat{\mathbf{x}}_s = \left(\mathbf{r}_d, \frac{\mathsf{C}_{1,2}(\mathbf{d}_1 - \mathsf{C}_{1,1}\mathbf{r}_d) + \mathsf{C}_{2,2}(\mathbf{d}_2 - \mathsf{C}_{2,1}\mathbf{r}_d)}{\mathsf{C}_{1,2}^2 + \mathsf{C}_{2,2}^2},\right)^\mathsf{T}. \tag{4.6}$$

The first solution answers, what diffuse reflectance will result in the remaining outgoing radiance, when the specular part is fixed. The second solution is of similar nature: what specular reflectance will best produce the outgoing radiance when the diffuse part is fixed? In both, glossiness is fixed. To find the solutions for varying glossiness, we simply enumerate in n_c discrete steps, resulting in $n_c + 2$ possible solutions: A diffuse change, a specular change and many glossy changes:

$$\mathfrak{X} := \{(\hat{\mathbf{x}}_d^{(\mathbf{r}_{g,i})}, \mathbf{r}_{g,i}), (\hat{\mathbf{x}}_s^{(\mathbf{r}_{g,i})}, \mathbf{r}_{g,i}), (\mathbf{r}_{d,i}, \mathbf{r}_{s,i}, g_1), \ldots, (\mathbf{r}_{d,i}, \mathbf{r}_{s,i}, g_{n_c})\}$$

Again, the tentative solution with the smallest residual error is picked using α from Equation 4.4

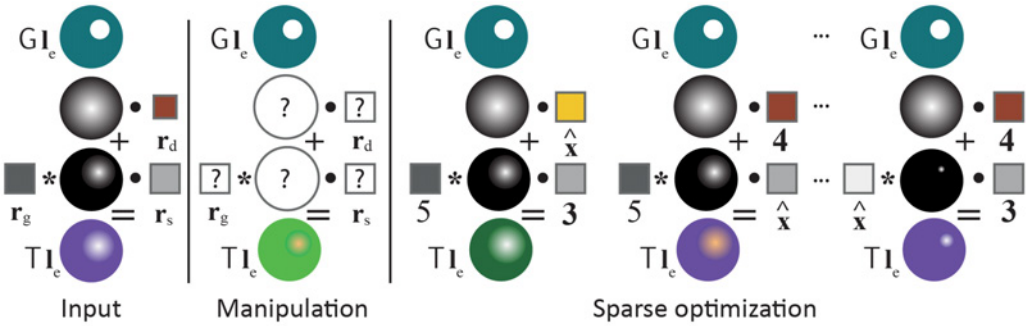

Figure 4.9: Sparse fitting. In every column, from top to bottom are the given lighting *(1st row)*, diffuse component *(2nd row)*, glossiness *(left)* and specular *(left)* component *(3rd row)*, and the SLF *(4th row)*. Starting from a given lighting *(1st col., 1st row)* and desired SLF *(2nd cols, 4th row)*, the optimal reflectance is found by solving "only" for either diffuse *(2nd row, 3rd col.)* or specular *(right, 3rd row, 4th col.)* or glossiness *(left, 3rd row, 5th col.)* while the remaining 2 components are fixed.

Mixed-Norm Solver A mixed solver computes changes in all ($\mathcal{L}_2$), in some (similar to $\mathcal{L}_0$, but with mixed degrees of freedom instead of only 1), and in only one parameter ($\mathcal{L}_0$) at the same time. We enumerate different glossiness in n_c discrete steps and current glossiness, in every glossiness iteration, we find the solution in case of fixing both, fixing diffuse only, fixing specular only or fixing none, resulting in $4(n_c+1)$ different solutions:

$$\begin{aligned}\mathfrak{X} := \{ \quad & (\mathbf{r}_{d,i}, \mathbf{r}_{s,i}, \mathbf{r}_{g,i}), (\hat{\mathbf{x}}^{(\mathbf{r}_{g,i})}, \mathbf{r}_{g,i}), (\hat{\mathbf{x}}_d^{(\mathbf{r}_{g,i})}, \mathbf{r}_{g,i}), (\hat{\mathbf{x}}_s^{(\mathbf{r}_{g,i})}, \mathbf{r}_{g,i}) \\ & (\mathbf{r}_{d,i}, \mathbf{r}_{s,i}, g_1), (\hat{\mathbf{x}}^{(1)}, g_1), (\hat{\mathbf{x}}_d^{(1)}, g_1), (\hat{\mathbf{x}}_s^{(1)}, g_1) \\ & \vdots \\ & (\mathbf{r}_{d,i}, \mathbf{r}_{s,i}, g_{n_c}), (\hat{\mathbf{x}}^{(n_c)}, g_{n_c}), (\hat{\mathbf{x}}_d^{(n_c)}, g_{n_c}), (\hat{\mathbf{x}}_s^{(n_c)}, g_{n_c}) \}.\end{aligned}$$

To this end, weights are given to the outcome of each solution, e. g., 1 to the sparse (fix 2 parameters out of 3), 5 to the mixed (fix 1 parameter out of 3) and 25 to the least squares solver (fix none). The best solution is again picked using α from Equation 4.4.

***n*-bounce** Because change in reflectance in the presence of multiple bounces also changes the incoming light for every element, the above process is iterated. This approach smoothes out the error, but in the presence of specular transport might not converge to the global optimal reflectance. In practice we did not observe any problems with convergence towards local minima.

4.5 GPU Implementation

We parallelize the solver over all elements. One thread is executed for every element and all possible solutions $\mathfrak{X}$ are enumerated and the best one returned by α. Still, computing the elements of A is computationally expensive: the first column contains exitant diffuse illumination for all directions, the second one stores exitant specular illumination for all directions. To compute exitant illumination, first the geometry operator needs to be applied,

and second the convolution with one diffuse and with many specular kernels (one for each glossiness) needs to be performed. Next we will introduce pre-computed visibility (Section 4.5.1) and pre-convolved radiance (Section 4.5.2) to accelerate both steps. After this, the final algorithm is explained, including pseudocode in Section 4.5.3 and an approach to up-sample (Section 4.5.5) and render (Section 4.5.4) the solution found.

4.5.1 Pre-computed Visibility (G)

When dealing with solid surfaces, the operator G is a matrix with only one non-zero entry per row. Let $\mathbf{h} \in \mathbb{N}^{n_p n_d}$ be an integer vector storing the index of this non-zero entry of row u in entry $\mathbf{h}_u$. A texture is used to store $\mathbf{h}$, allowing to apply G to a vector $\mathbf{l}_e$ stored in a texture using a single indirect texture read. The vector $\mathbf{h}$ is pre-computed in two passes (Figure 4.10).

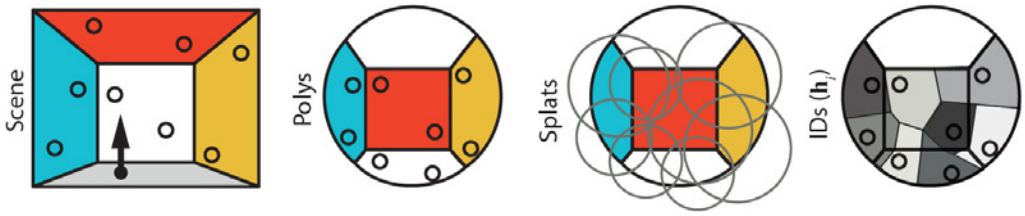

Figure 4.10: Pre-computed visibility. *(a)* The finite element points P *(circles)* of the surface $\mathcal{M}$ *(polygons)* and one particular point element p *arrow*. *(b)* View of the scene from p, including the other elements. *(c)* The same view, but with the splat around every element. *(d)* Grey-coding of the id j of the element p_j, that was closest to the surface location under q.

Visibility from every point in every direction is resolved in a first pass. To this end, the scene's polygons are rasterized n_p times using paraboloid projection and depth-buffering from each $\mathbf{p}_i$ into a texture of size $\lceil\sqrt{n_d}\rceil \times \lceil\sqrt{n_d}\rceil$ that contains the nearest surface positions.

Second, the index of the element closest to each surface position in this texture is found. This is achieved by drawing every element $\mathbf{p}_j$ in P as a splat with a size that guarantees the covering of $\mathcal{M}$. For every pixel covered by this splat with surface position p found in the first pass, the value j is written if, and only if, $d = ||\mathbf{p}_j - \mathbf{p}||$ is smaller than the one of all splats drawn before it. The conditional write is accomplished by using the distance d instead of the real depth value in a z-buffer. This 5D variant of the Voronoi construction proposed by Hoff III et al. [1999] takes less than a second for a typical scene such as Figure 4.1, and consequently allows for near-interactive editing of dynamic scenes.

Discussion The parameters n_p limits the spatial and n_d the angular resolution i. e., glossiness of rendering and editing; n_c limits the granularity of gloss control as shown in Figure 4.23 and Figure 4.24. Restricting G to be binary causes under-sampling, as multiple elements project into one bin and all but one are lost; a common problem for the hemi-cube in radiosity. If the emissive lighting is only from point lights, using $\mathbf{h}$ to apply G is replaced with shadow maps that use a more efficient discretization from the light's view in all our results.

4.5.2 Pre-convolved Radiance (K)

For a single element, applying K_s to $\mathbf{l}_\mathrm{i}$ in order to compute all A's would require as much as $n_\mathrm{c} n_\mathrm{d}^2$ operations: A loop that, for each glossiness level and each direction, visits every other direction. This cost can be reduced to constant time $4/3\, n_\mathrm{d}$ by making two observations: First, applying $\mathsf{K}_\mathrm{s}(g)$ behaves as a lowpass filter with a cutoff proportional to g. Second, we need to know the result of applying all filters $\mathsf{K}_\mathrm{s}(g_0), \ldots, \mathsf{K}_\mathrm{s}(g_{n_\mathrm{c}})$ at the same time. Both can be achieved using recursive filtering [Williams 1983], which was used for environment maps [Heidrich and Seidel 1999] or radiance caching (Chapter 3). To this end, all that is required is a MIP map of a mirrored version of the incoming light $\mathbf{l}_\mathrm{i}$, denoted as $\hat{\mathbf{l}}_\mathrm{o}$. On level 0, such a map contains the reflected light only, corresponding to a mirror. On higher levels, increasingly blurred versions that correspond to lower glossiness values can be accessed in constant time. Diffuse reflection K_d is an extreme lowpass which can be stored in a small 4×4 texture. Please note, that a different linear basis, such as Spherical harmonics will not allow to perform the reflection faster either. Pre-convolved radiance caching will also be used to render the final result in Section 4.5.4.

Figure 4.11: Pre-convolution of one lumitexel from Figure 4.1.

4.5.3 Solver

Listing 4.1 summarizes our system in pseudocode. We implemented our approach using GLSL.

4.5.4 Rendering

Direct light is computed using common interactive rendering, e. g., shadow maps. To compute indirect reflected light efficiently [Scherzer et al. 2012], the MIP map $\hat{\mathbf{l}}_\mathrm{o}$ created in the solver is used as well. For improved quality, the correction used for environment maps by Szirmay-Kalos et al. [2005] is included. The rendering uses radiance cache splatting [Gautron et. al. 2008] to propagate the reflected lumitexel to a deferred framebuffer.

All radiometric units are stored as full-precission RGB float values into textures. We use a gamma tone-mapper that is applied forward when putting an image onto the screen and backwards when specifying colors. 2×2 supersampling is applied to all results.

4.5.5 Upsampling

While the solution for a discrete set of elements can be efficiently computed, $\mathcal{M}$ and R might contain fine details which are not represented well in P. To be able to propagate the

```
(P, r_r) := sample(M)
h := precalcVisibility(P, M)
r_f := r^m := r
while user interacts {
 for k := 0 to n_b { // Bounces
  for i := 0 to n_p in parallel { // Elements
   l_i := lookupVisibility(h, l_e)
   l̂_o := createMIPMap(l_i)
   l_o := reflect(P, r_f,i, l̂_o)
   b := tool(P, l_o)
   g := {g_1, g_2, ..., g_nc, r_g,i}
   X := {}
   for j := 0 to ||g|| in parallel { // Gloss
    A := (K_d * l_i, getMIPLevel(l̂_o, g_j))
    C := A^T * W * A
    d := A^T * W * b
    X.add((sol_a(C, d), g_j)) // Equation 4.3
    X.add((sol_d(C, d), g_j)) // Equation 4.5
    X.add((sol_s(C, d), g_j)) // Equation 4.6
    X.add((r_d,i, r_s,i, g_j))
   }
   r_i^m := alpha(X, l_o, l̂_o) // Equation 4.4
  }
  r_f := r^m
 }
}
```

Listing 4.1: Pseudo-code of the mixed-solver approach.

manipulation to such details, joint bilateral upsampling in the framebuffer is used. Let Q be a frame buffer created by rasterizing the detailed scene. It typically contains millions of elements. We upsample P to Q, using radial basis function (RBF) reconstruction, with two important properties. First, we interpolate deltas in the form of differences of the original reflectance and the new reflectance. This keeps fine details in Q instead of overwriting them. Second, the distance in appearance is included in the reconstruction kernel: Elements need have a similar original appearance to be changed in a similar way. Reconstruction is

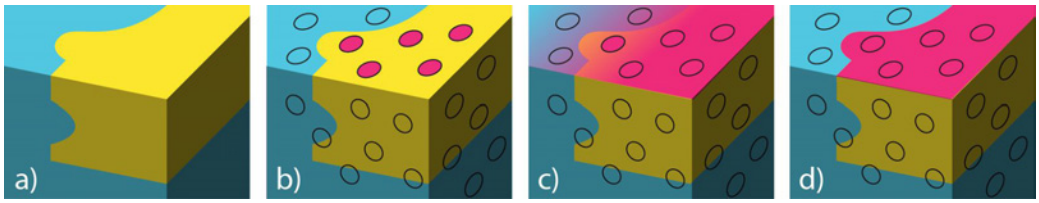

Figure 4.12: *(a)* A scene detail with appearance discontinuities. *(b)* Five elements *(circles)* change to pink. *(c)* Conventional upsampling according for normals and positions. *(d)* Including appearance discontinuities preserves details.

performed using RBF splatting similar to the reconstruction used in rendering. At every 3D location p in P a screen-aligned quad is drawn, large enough to overlap with all similar pixels q. The RGB color of every quad is constant and encodes the change of shading

parameters. The alpha value depends on the similarity between p and q. Additive blending in RGBA is used to combine multiple splats. After all splats are drawn, the RGB component in every pixel is divided by the alpha value for normalization.

4.6 Results

We report results in form of several use cases and a performance evaluation.

Usecases A basic manipulation is diffuse painting (Figure 4.13).

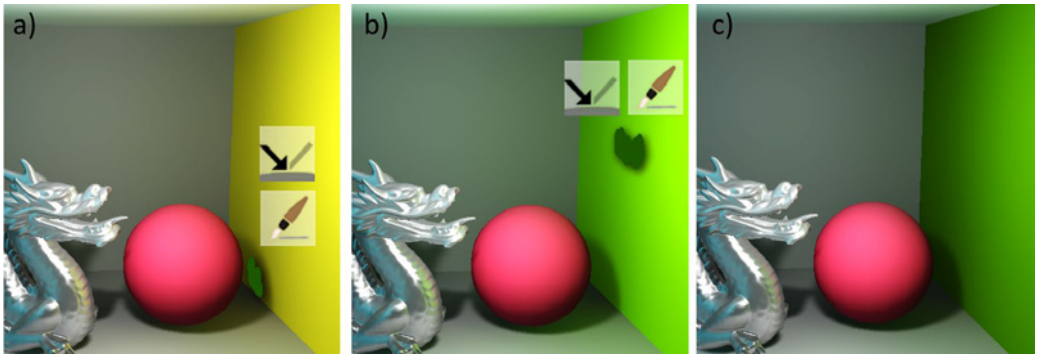

Figure 4.13: Diffuse manipulation. *(a)* A direct stroke in a shadow area and in the presence of GI. *(b)* The reflectance is optimized resulting in a matching appearance under this complex illuminant. A new direct stroke is applied outside the shadow. *(c)* The result again discounts for the illuminant.

In the presence of multiple bounces, we find our iterative approach to converge against a minimum that is close to the desired result (cf. Figure 4.14). While this optimum is likely not global, in our scenes a close local-minimum fit to the desired appearance was achieved.

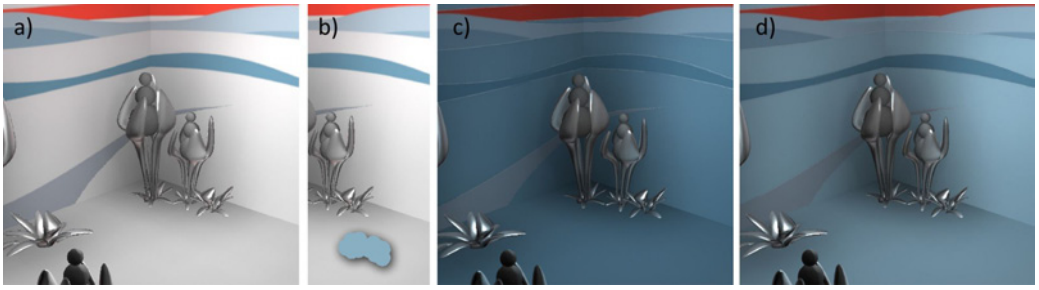

Figure 4.14: Diffuse painting with multiple bounces. *(a)* Input. *(b)* A pale-blue stroke. *(c)* After 1 iteration, the color is too dark because multiple bounces are not accounted for. *(d)* Iterating the solver three times with the newly optimized reflectance, the appearance converges against the users prescription.

A more advanced usage of our system is the design of view-dependent appearance (Figure 4.15). To our knowledge, no attempts were made in previous work to “understand” the

intention of such an input, which likely was to change specularity. Our mixed-$\mathcal{L}_p$ solver will detect that the best single change of material parameter is to adjust specularity.

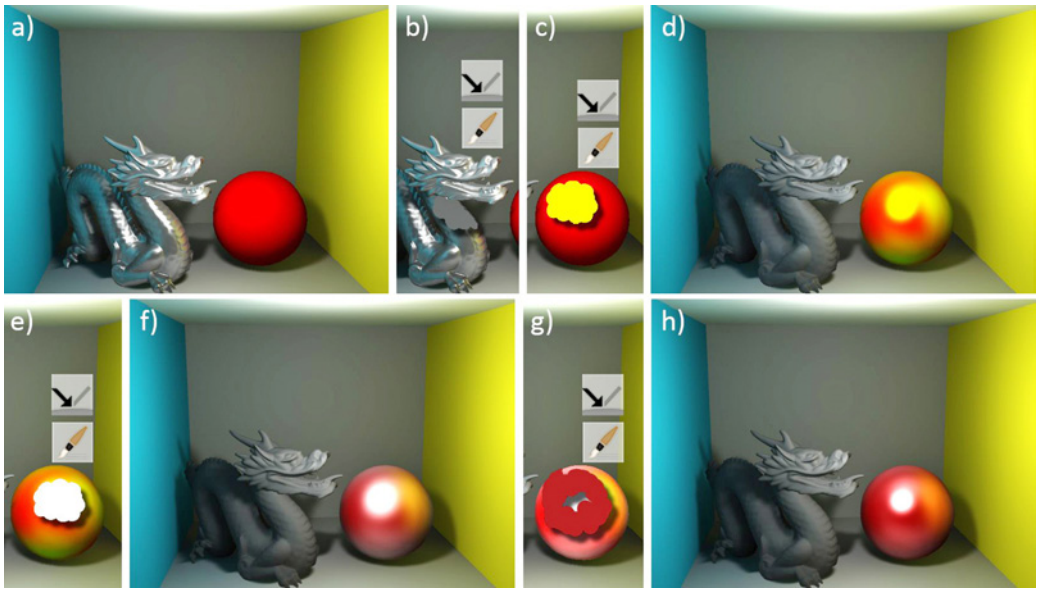

Figure 4.15: Sketching specular reflectance: *(a)* Input. *(b)* A gray stroke over a highlight. *(c)* A yellow stroke over the diffuse red ball. *(d)* The dragon has turned diffuse; the ball has turned specular. *(e)* A white stroke on the highlight. *(f)* The ball now reflects more. *(g)* A change of highlight shape. *(h)* The system inferred a change of glossiness; reflections are sharp, the highlight small.

Direct tuning of light color and reflectance of all surfaces that affect one surface location, can be a tedious process. Especially for diffuse surfaces, many other locations affect a single location. In Figure 4.16, the desired appearance of a location subject to indirect lighting is changed and all other locations alter their reflectance to achieve the desired appearance. Indirect painting generalizes also to specular (glossy) surfaces (Figure 4.17) .

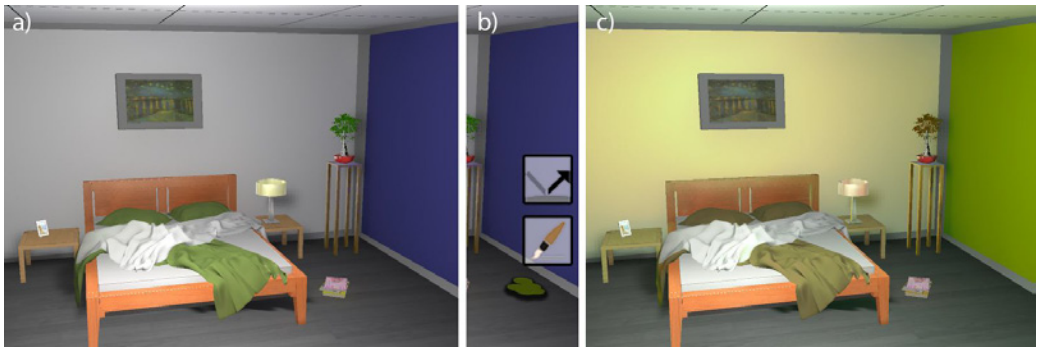

Figure 4.16: Simple diffuse indirect painting. *(a)* Input. *(b)* An indirect green stroke. *(c)* Reflectance of all other surfaces is changed to achieve the desired appearance.

Caustics are view-independent effects, but caused by light reflected from a specular object. Our system allows to tweak caustic appearance in an indirect way (Figure 4.18). We do not assume any particular type of lighting, all that is required is a vector $\mathbf{l}_e$. This allows to design

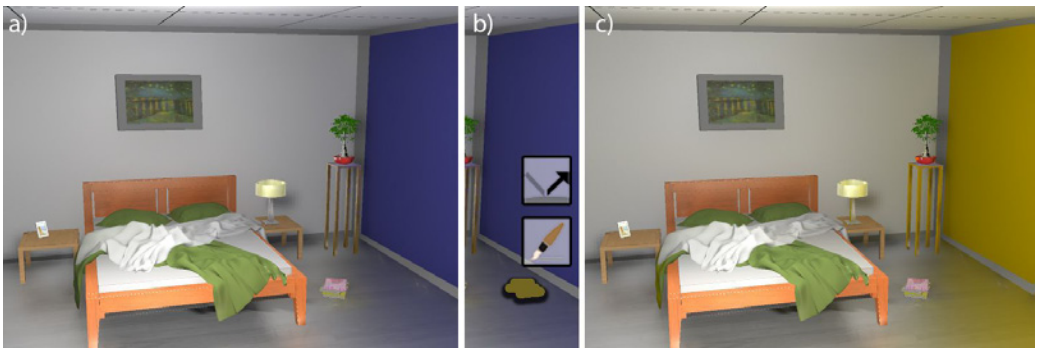

Figure 4.17: Specular indirect painting. *(a)* Input *(b)* An indirect orange stroke is made in the input scene onto the ground. *(c)* Reflectance is changed (right wall) where it contributes to the ground.

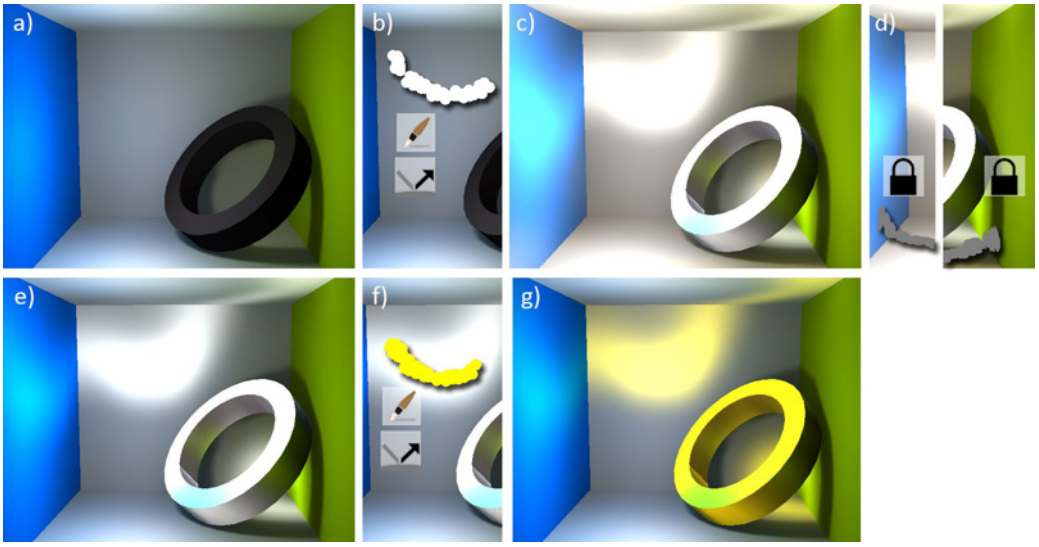

Figure 4.18: Caustic design session: *(a)* Input. *(b)* White indirect stroke. *(c)* Ambigue result. The system performs a change of reflectance on many surfaces. *(d)* Two fixate strokes are made. *(e)* Now, the best solution is to change the ring's material to white specular.. *(f)* Yellow stroke on the caustic. *(g)* The desired final appearance is achieved: a yellow caustic from a golden ring.

appearance under complex illumination, such as captured environment maps Figure 4.19. In Figure 4.20, a geometricaly detailled scene with detailled textures and bump maps is edited. Note how both material details are preserved, and manipulations are propagated to regions of similar appearance in the proximity of the stroke.

Besides sketching brushes, global stylization can be applied to the SLF. The result is a scene appearance, a valid reflectance, but yet with an unsharp-masked look (see Figure 4.21).

Analysis Figure 4.22 compares the results of $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ optimization from the same user input. Figure 4.23 and Figure 4.24 show the effect of increasing / decreasing spatial resolution, directional resolution and gloss levels. Figure 4.25 shows how different brush strokes results in different materials.

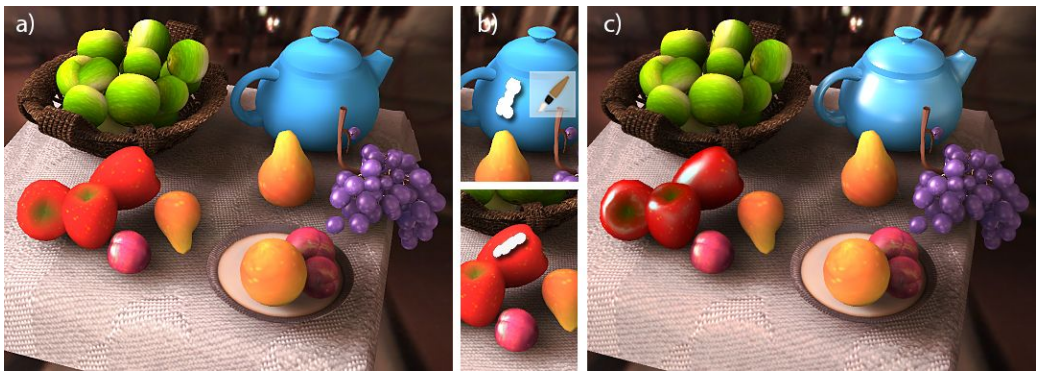

Figure 4.19: Editing under enviroment map lighting. *(a)* Input. *(b)* White stroke onto the teapot interpreted as a highlight *(c)*.

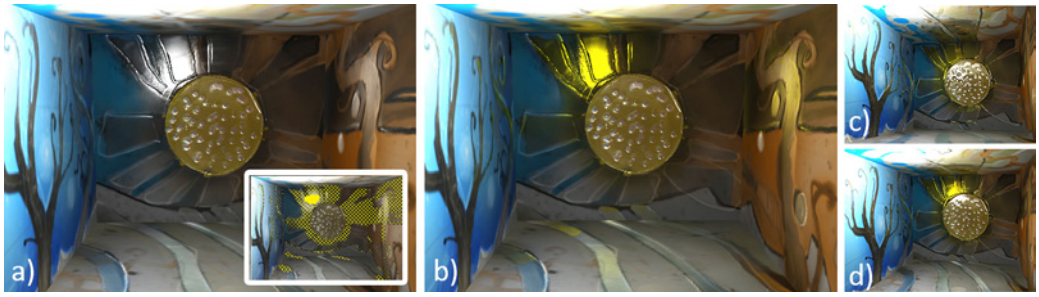

Figure 4.20: Editing detailed appearance. *(a)* Input and stroke with similarity marked using 2D checkers (inset) *(b)* The change of specular. *(c)* and *(d)* The result from different views. Note the specular highlight appearing as yellow, including the details from the bump map.

Performance For $n_p = 10000$ elements with a resolution of $n_d = 32 \times 32$ the solver takes around 270 ms. Reflectance upsampling from discrete points to current view take roughly 150 ms for a 900×450 resolution view port, radiance cache splatting is around 75 ms. In overall, our framework allows interactive feedback for a design session. A performance breakdown is showed in Table 4.3.

Step	Time (ms)	Memory (MB)
Precomputed Vis.	870	31.64
MIP map creation	138	47.46
Tool	50	47.46
Solver	270	-
Upsampling	150	-
Radiance splatting	75	-

Table 4.3: Performance breakdown for Figure 4.1.

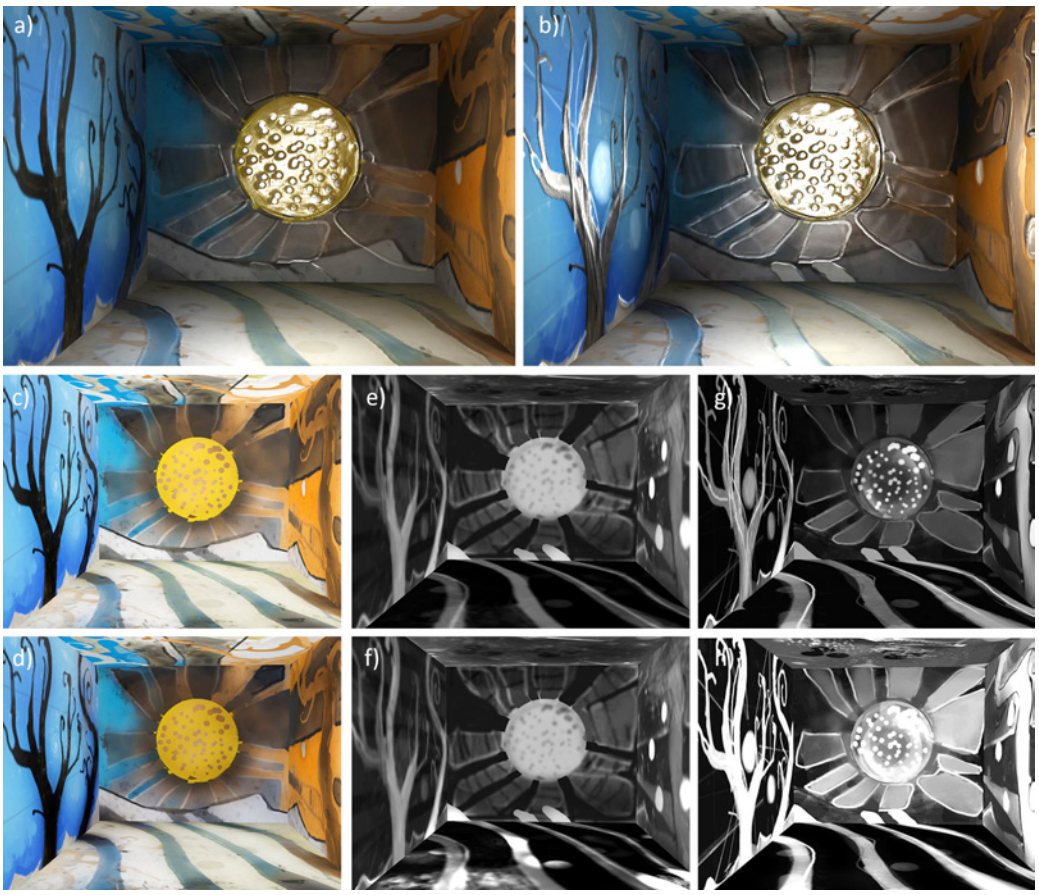

Figure 4.21: Stylization of a scene *(a)*, using unsharp masking *(b)* by changing the original reflectance *(c,e,g)* into the new reflectance *(d,f,h)*, where *(c,d)* are diffuse components, *(e,f)* are specular components and *(g,h)* are the glossiness components.

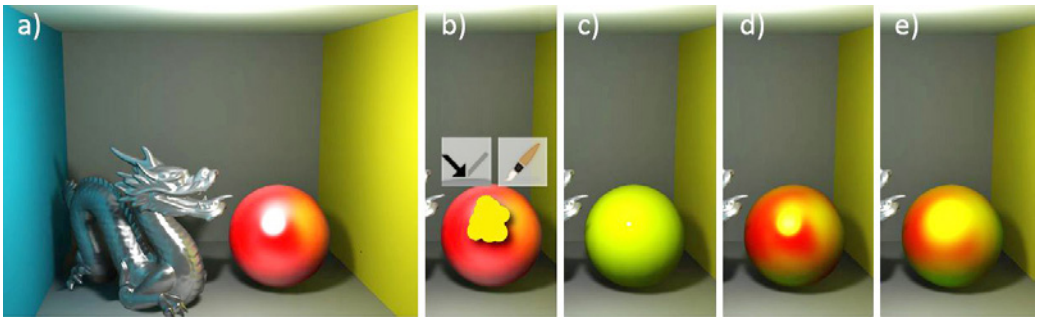

Figure 4.22: *(a)* Input image and *(b)* stroke. *(c), (d), (e)* show results produced by the $\mathcal{L}_2$, $\mathcal{L}_0$ and "mixed" solvers respectively. The $\mathcal{L}_2$ solution turns object into a diffuse-only material to match user's input. The $\mathcal{L}_0$ solution gives better result by keeping both old diffuse and old glossiness, modifying only specular color. The "mixed" solver further improves the result by keeping only the original diffuse color and modifying both specular color and glossiness to match the user's input.

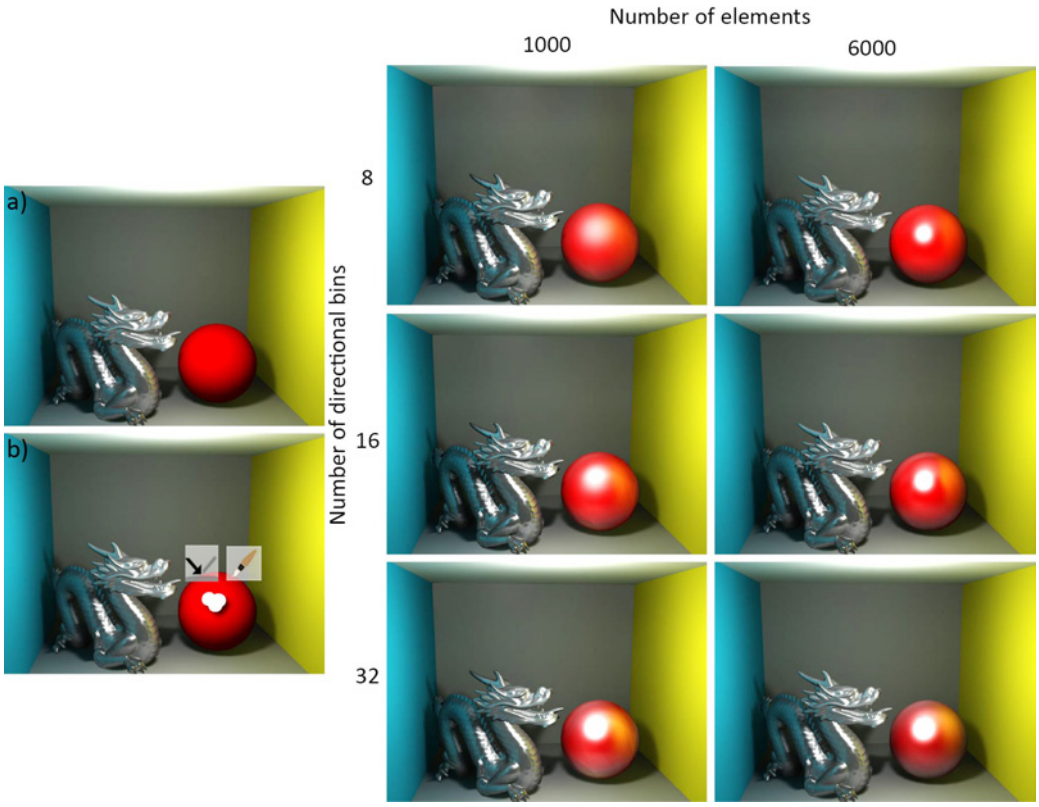

Figure 4.23: Effects on spatial and directional resolution. The original image *(a)* and the input user's stroke *(b)*. The images in the matrix show how our system performs with different number of element (n_{p}) and number of directional bins (n_{d}). The searching of glossiness values is fixed to $n_{\mathrm{c}} = 30$

4.7 Dicussion

The result of our editing is scene-dependent, which is both a strength, but also a limitation that e. g., disallows to simply transfer appearance to a different scene. Our approach is better suited for adjusting, rather than creating something from scratch. The resulting surfaces with reflectance can be used in the following steps of a common pipeline, and are in theory even fabricable, which both is not the case when stylizing radiance alone. Transparent surfaces, would require to exchange our pre-computed $\mathbf{G}$ with raytracing. There is a limit in rendering and editing detail which can be achieve using the proposed regular discretization. Beyond this limit, adaptive discretization would be required.

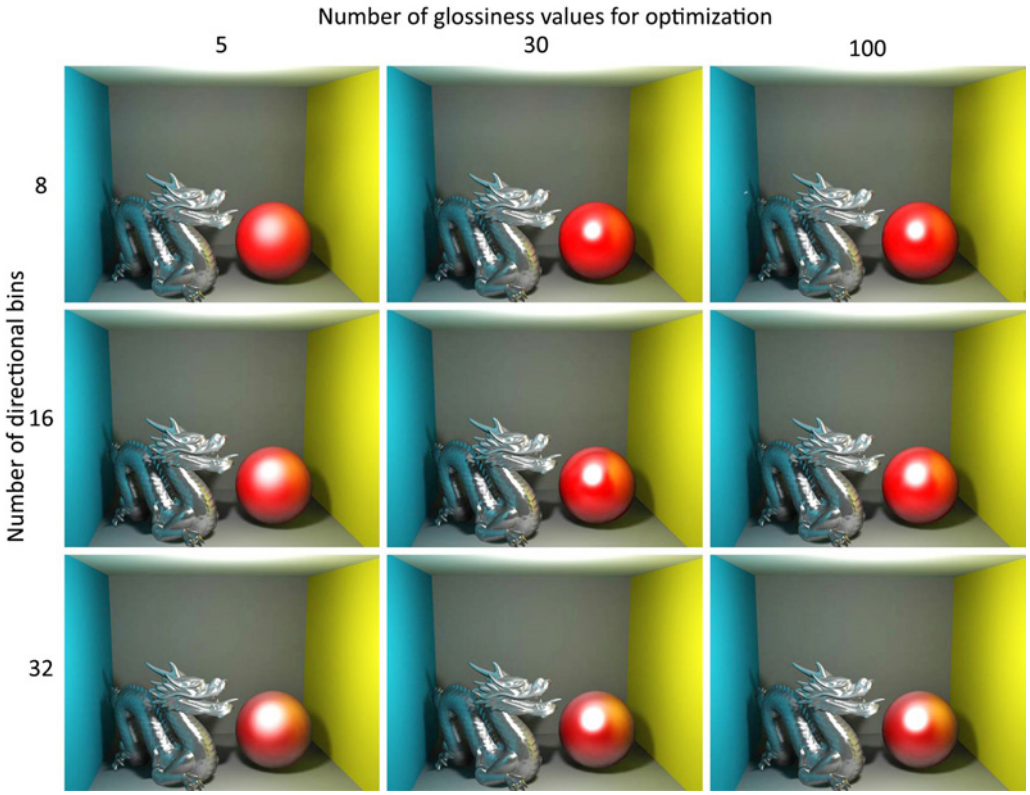

Figure 4.24: Effects on gloss level and directional resolution. Scene and use input is the same as in Figure 4.23. The matrix shows how our system performs with different nubmer of glossiness values (n_c) and directional bins (n_d) given the same number of elements $n_p = 6000$

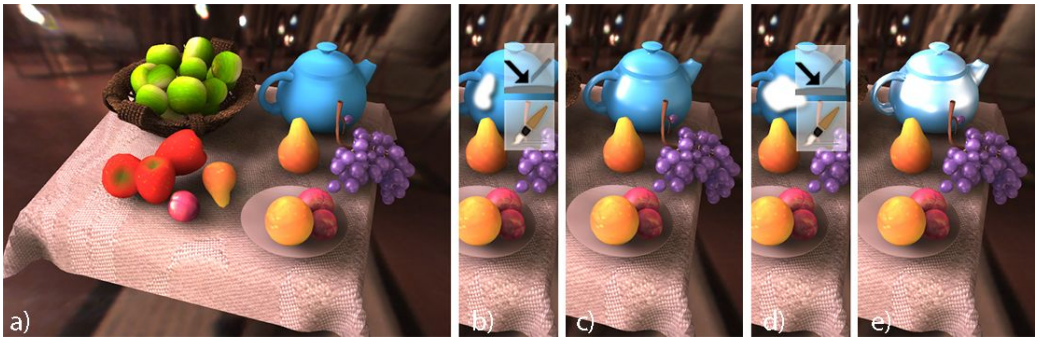

Figure 4.25: Effects of different brush size on the same object. The original image *(a)*, painted with different brush sizes *(b)*, *(d)* results in *(c)* and *(e)*.

5

3D Material Style Transfer

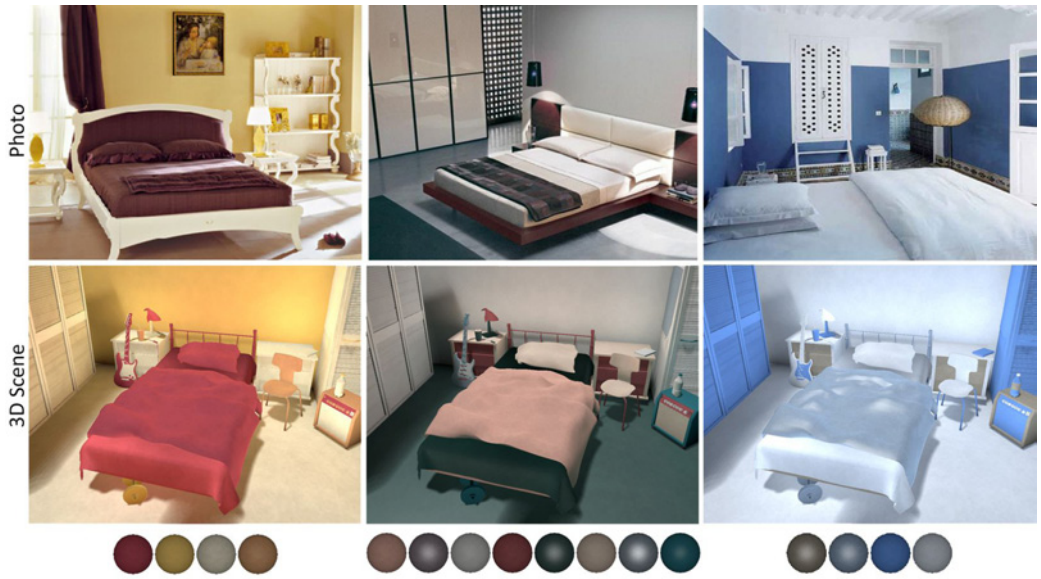

Figure 5.1: Automatic 3D material style transfer from different source images *(top)* to a target 3D scene *(bottom)* using our approach. In every column, the extracted materials from the source images are shown as colored sphere *(bottom)*.

5.1 Introduction

Not all 3D scenes come with assigned materials (i. e., reflectance properties); a 3D scanner might not deliver colors or a model from the internet was simply crafted without. Images rendered with such scenes do not appear realistic, as users expect certain materials for certain objects.

When creating thumbnails to browse large databases materials are often a must. When materials are not available, a manual assignment is tedious. Especially, when a high number

of objects and materials are involved, selecting, grouping, and navigating is challenging. Further, deciding on an object's material is not obvious, for example when colored light, e. g., due to indirect illumination is involved. The resulting appearance depends on the spatial context; a white object near a colored wall will exhibit color bleeding and not appear white anymore. In other situations, non-obvious rules, that might even be unknown to the user, should be applied, e.g., for architectural models, a common principle is to assign darker materials to the floor than the walls to convey a feeling of higher ceilings. In particular, when a number of design variants should be examined (Figure 5.1) from different viewpoints, transferring mood from examples can serve as an inspiration. Similarly, organizing material assignments according to a style, might be beyond the user's capabilities, but can be important, as humans make certain color assumptions depending on the context (i. e., the Stroop [1935] effect for chroma). Automatically accounting for such expectations makes scenes easier to understand and more pleasant.

Addressing these issues, this chapter proposes an automated system that extracts materials from a guide source, such as an image or video and assigns them to a target 3D scene. The material extraction approximately captures appearance from uncalibrated images of unknown geometry and lighting using image-filtering heuristics. The material assignment is formalized as an optimization problem that assigns discrete materials to discrete objects in order to minimize a cost function that evaluates the perceptual difference to a guide source. The cost function accounts for image differences as well as geometric matching between objects in the target 3D scene and the guide source. Furthermore, to a certain extent, the cost function grasps inter object relations including global illumination aspects. The user can therefore, by providing a simple guide source, attribute plausible materials to an entire 3D scene fully automatically.

5.2 Our Approach

Our system consists of two key components: *material extraction* (Section 5.2.2) and *material assignment* (Section 5.2.3). From a guide source, which can be an image or video, a set of materials is extracted. Then the system finds the best assignment of these materials to a target 3D scene (Figure 5.2).

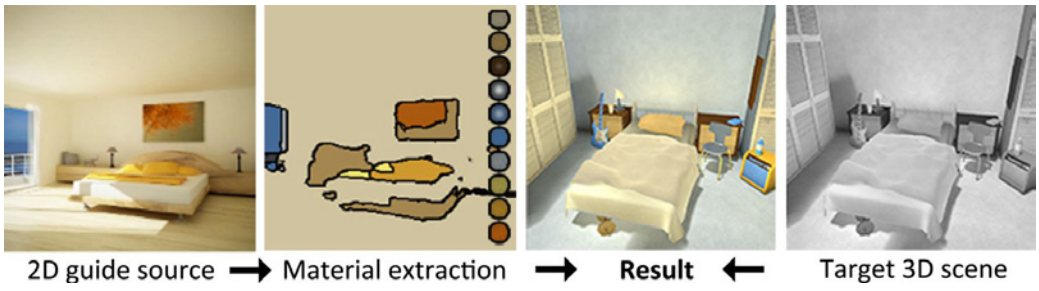

Figure 5.2: Our approach extracts materials *(2nd col.)* from a 2D guide source *(1st col.)* and assigns them to a target 3D scene *(4th col.)* so that its appearance preserves the style of the source after globally illuminated *(3rd col.)*.

5.2.1 Definitions

Here, we introduce the basic definitions that our system builds upon: materials, objects, material assignments and rendering.

We represent materials by (slightly modified) Phong coefficients [Phong 1975]: diffuse color, monochromatic specular intensity and glossiness. Textures are represented using anisotropic noise with a particular frequency spectrum [Perlin 1985] modulating the diffuse color. We decided to not optimize for the number of materials, but assume it to be n_m, a user-controlled parameter.

A scene is hand-crafted and therefore usually segmented into meaningful entities, or it needs to be segmented into objects that will receive the extracted materials. We assume a segmentation into n_o objects a given.

A material assignment $A := \{a_i \in \mathbb{N}^+ | a_i < n_\mathrm{m}, i < n_\mathrm{o}\} \in \mathcal{A}$ is a mapping from every object i to a material a_i. It is neither assumed that every material is used, nor do we enforce that it has to be used more than a certain number of times (or any other similar restriction).

We assume a set $\mathcal{V}$ of views on the scene which we will simply call "the views" for which the assigned materials should be optimal. It can either be a set with only one element defining a single camera position and orientation, a camera path, or a volume in space describing the potential viewpoints. When the views fill the complete space, no view-specific optimizations will be made.

Rendering in our context is an operator $r(A,V) := \mathcal{A} \times \mathcal{V} \to (\mathbb{R}^2 \to \mathbb{R}^3)$ that converts our fixed target 3D scene under some material assignment A and some views V into a two-dimensional RGB image.

5.2.2 Material Extraction

Extracting physical materials from images is a very ill-posed problem that we try to avoid. Instead, we extract plausible materials by splitting the image into several components that map to different shading parameters: Diffuse color, specularity and texture (Figure 5.3). First, the input image $\hat{L}$ is (inverse-gamma) converted into linear units and white-balanced to L. Then, L is split into a diffuse L_d and a specular scalar radiance L_s. For fast performance and simplicity, the method of Yang, Wang and Ahuja [2010] is used, but others are possible. Next, the diffuse radiance L_d is split into a base L_b and a texture part L_t. Bilateral filtering [Durand and Dorsey 2002] is used to decompose L_b in L_d and $L_\mathrm{t} = L_\mathrm{d} - L_\mathrm{b}$.

Accounting for a material's specularity is challenging: highlights are only visible in some parts (white dots in Figure 5.4) of an object although the specularity is the same everywhere on the object. We make the assumption that a continuous diffuse base color L_b implies the continuity of material (material segmentation). Conceptually, we can hereby associate a highlight to all places where it could have appeared for a different combination of light and viewpoint. The material segmentation is computed as follows: First, the CIELAB values of all base diffuse pixels L_b are clustered using k-means [MacQueen 1967] applying the CIEDE2000 [Sharma, Wu and Dalal 2005] color difference metric. Next, disconnected k-means clusters are split into individual segments. Finally, to compensate

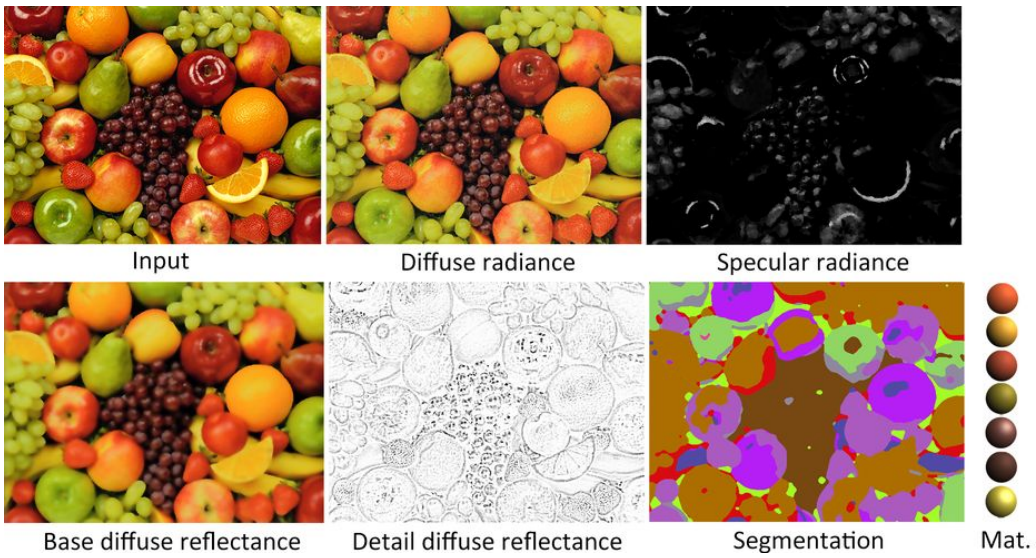

Figure 5.3: Our material extraction pipeline is shown from left to right, top to bottom. The input, a single image, is decomposed into diffuse and specular radiance. Next, the diffuse radiance is split into base- and detail diffuse reflectance. The latter will be represented as local statistics. Finally, the full high-dimensional material info (diffuse reflectance, specular intensity, and the detail statistics) is segmented into discrete clusters which are assigned to the target 3d scene.

Figure 5.4: From left to right: Starting from an input image L, with diffuse colors (yellow, blue) and different highlights (white spots), the diffuse base color L_b is used for the material segmentation to propagate specularity and glossiness.

for over-segmentation (e. g., large gradients in the background), neighboring segments are merged if the average color difference on their boundary pixels is less than a certain threshold.

To extract texture information, we use a bilateral Laplacian pyramid [Fattal, Agrawala and Rusinkiewicz 2007]. The pyramid is computed by convolving L_t with a bank of n_b bilateral filters and subtracting subsequent levels. Doing so, strong edges in L_t, are excluded from the frequency response. In practice, we use $n_\mathrm{b} = 4$ levels. An additional filtering using a 5×5 Gaussian filtering accounts for the local phase insensitivity of the HVS. This results in a spatially varying map $S : \mathbb{R}^2 \rightarrow \mathbb{R}^{n_\mathrm{b}}$.

Using the material segmentation, the diffuse color, specularity and texture coefficients are calculated within each segment using robust statistics [ODonovan et. al. 2011] resulting in k' materials. Depending on the scene structure, k' might be very different from the desired number of materials n_c. Therefore, we interpret the the k' materials once again as a high-dimensional (8 D) point cloud and cluster it once more, resulting in the desired n_c final materials. In practice, segmenting into a handful of discrete materials is sufficient to well-match the guide images. Figure 5.5 shows several material extraction using our approach.

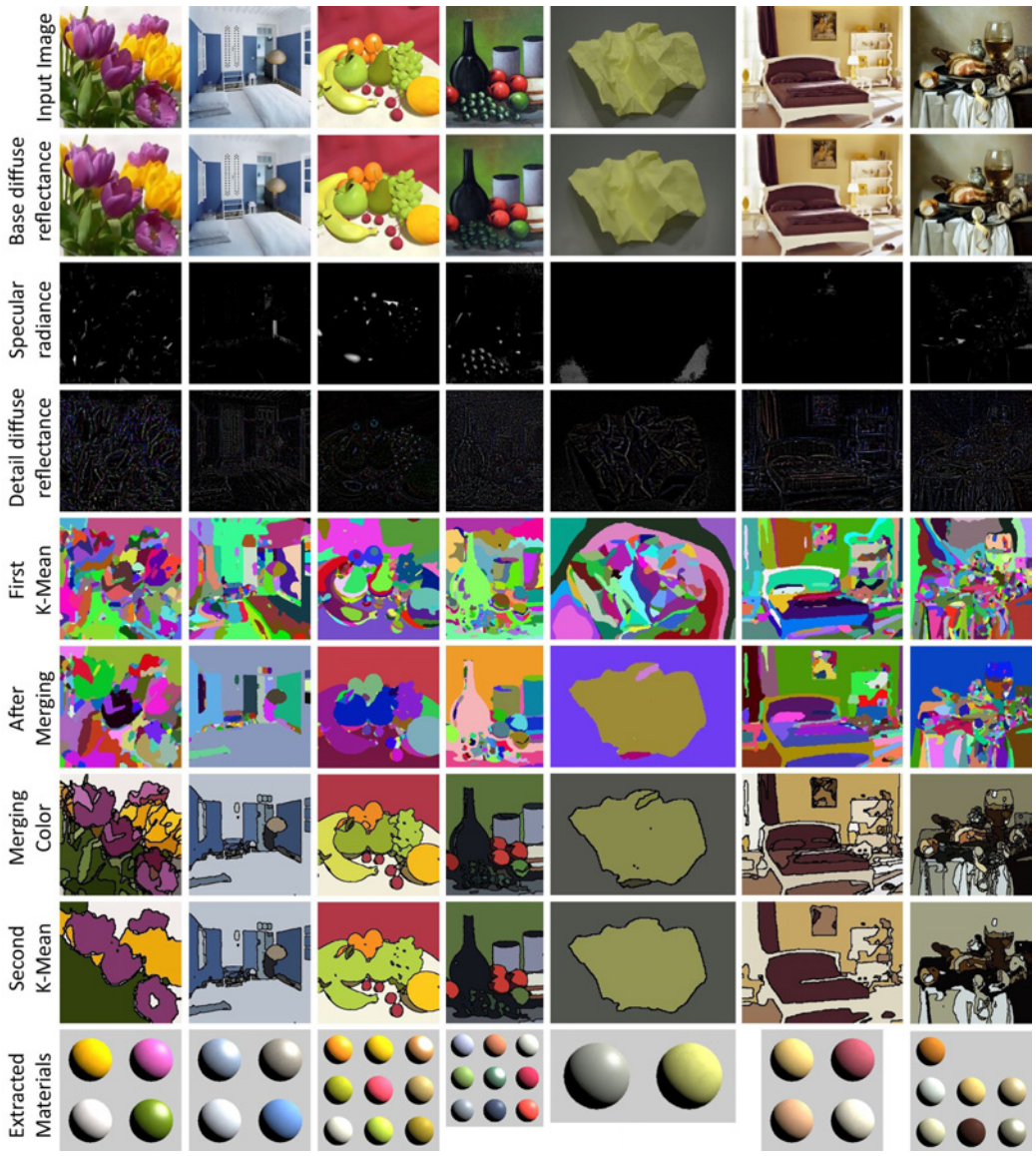

Figure 5.5: Several material extraction results. Input image *(1st row)* is decomposed into diffuse *(2nd row)*, specular *(3rd row)* and detail *(4th row)* components. Disconnected components are clustered using k-means *(5th row)* and merged *(6th row)*. The *7th row* shows the color of the merged clusters. Next are the results of the second k-means *(8th row)* and the extracted materials *(9th row)*.

We also support other guide inputs, such as image sequences that are processed on a per-frame basis and allow for direct sketching of a guide sources that we designed a simple user interface for. The system then updates the scene according to sketched materials (e. g., a red circle with a highlight and a blue square). To ease explanations, we will in the following consider a single guide image as input.

5.2.3 Material Assignment

An appropriate assignment A is found by minimizing a cost function $d(A,V)$. d is high, if the guide source is perceptually different for the material assignment A under view V. The cost is the sum of two components

$$d(A,V) = w_{\mathrm{i}} d_{\mathrm{i}}(r(A,V)) + w_{\mathrm{g}} d_{\mathrm{g}}(A). \tag{5.1}$$

A view-dependent image cost d_{i} compares the guide source to a rendered image using the current view and assignment. A view-independent geometric cost d_{g} verifies that the assignment is consistent with the geometry. $w_{\mathrm{i}}, w_{\mathrm{g}}$ determine the relative weighting between d_{i} and d_{g} respectively. We detail both components in the following, before showing how to efficiently optimize for this cost function.

Image Cost d_{i} penalizes perceived difference of material appearance for all rendered views with respect to the guide source. Histogram calculations are used to compare the material mood of two images. To account for spatial variation, different *local* histograms are used in different parts of the image. We use a grid of 3×3. The CIE LAB color space is used, defining a 3D histogram with 10 bins in each dimension. Continuous bins are computed using Parzen [1962] windows. Finally, the distance between two histograms is computed using their intersection [Swain and Ballard 1991].

The final cost is then defined as the integral of the image cost over all views. We could compute this integral using quadrature, i. e., computing it for a high number of views, but, as shown later, picking just random views is a sufficient Monte Carlo approximation.

Geometry Cost d_{g} penalizes geometrical differences between objects in the scene and the guide source that share the same material. A simple example illustrates the idea: A still-life-like guide source with a green apple and a yellow banana being is assigned to a 3D scene with an apple and a banana object. In terms of image costs, a green banana and a yellow apple are as plausible as a green apple and a yellow banana. Nonetheless, it is more intuitive to assign materials to similar shapes in the target and guide source, i. e., a yellow material to banana-like and a green material to apple-like shapes. Formally, we define:

$$d_{\mathrm{g}}(A) = \frac{1}{n_{\mathrm{o}}} \sum_{j=0}^{n_{\mathrm{o}}} \min_{0 \le i < n'_{\mathrm{o}}} (d_{\mathrm{a}}(i,j,A_i,A'_j))$$

with

$$d_{\mathrm{a}}(i,j,A_i,A'_j) := \begin{cases} d_{\mathrm{s}}(i,j) & \text{if } A_i = A'_j \text{ and} \\ \mathrm{float}_{\max} & \text{else,} \end{cases}$$

where n_{o} and n'_{o} are the numbers of objects in the target and source, i and j are shapes of target and source, A and A' are the corresponding material assignments and d_{s} is a shape difference metric. To compute the difference between two shapes we employ a 2D Angular Radial Transform (ART) [Bober 2001], and use the corresponding shape descriptor vector composed of 35 elements. We use ART because of its simplicity, quick computation, and robustness for shape-based geometry matching [Chen et al. 2003]. We use the minimal Euclidean

distance between the descriptor vectors over a range of orthographic views [Chen et al. 2003]. To make sure only meaningful matches contribute to the geometry cost, we redefine $d_g(A)$ as

$$d_g(A) = \frac{1}{n_o} \sum_{j=0}^{n_o} q_\rho(\min_{0 \le i < n'_o}(d_a(i,j,A_i,A'_j))) \tag{5.2}$$

where ρ is a constant to define the minimum requirement for meaningful shape matching and

$$q_\rho(x) := \begin{cases} x & \text{if } x \le \rho \text{ and} \\ 1 & \text{else,} \end{cases}$$

Note, that the geometric cost is view-independent.

If the guide source does not provide 3D shapes, i. e., for images and videos, the 2D shape is directly used in the shape metric. The shape extraction makes use of a different segmentation than can be deduced from the derived materials because, to use an example, two bananas in a guide source might form one material cluster, yet their shape is still the one of two single bananas, so $n_m = 1$ while $n'_o = 2$.

We use a multi-segmentation to decompose the guide source [Russell et al. 2006] because no single parameter setting for a segmentation approach will be sufficient to produce all meaningful segments. Mean shift [Comaniciu and Meer 2002] is used and the bandwidth parameter is varied to three different values to produce a fine, a medium, and a coarse segmentation. The resulting segments are different from the material segmentation, but only used to calculcate the geometry cost (Equation 5.2).

5.2.4 Optimization

Given a cost function, we find the best assignment via simulated annealing (SA). SA is an iterative improvement algorithm, in which permuted "neighboring" solutions are used to evaluate the cost function and submitted to an acceptance test. Permutations leading to a cost decrement are always accepted while permutations resulting in an increase are accepted according to a probability based on the Boltzmann factor [Kirkpatrick, JR and Vecchi 1983].

In general, a cost function is *sampled* for the current solution $c^i = d(A^i, V)$ as defined in Equation 5.1. Next, the current solution A^i is *permuted* to a neighboring solution B in a way that depends on the cost. If the cost c^i is high, a remote neighbor is chosen, if it is low, a close neighbor is selected. Once, a neighboring B is chosen, the next solution A^{i+1}.

To apply this principle to materials, we have to define a neighborhood on our solution domain, i. e., the space of material assignments $\mathcal{A}$. Intuitively, we want remote neighbors to change the material of many objects, and close neighbors to change the material of fewer objects, formally

$$B_j := \begin{cases} p(A^i_j) & \text{if } c^i > \xi_j \\ A^i_j, & \text{else,} \end{cases}$$

where $0 < \xi_j < c_{\max}$ (a user parameter) is a random number and $p(k) \in \{1, \ldots, n_m\}^2$ a material index permutation function to be defined next. The permutation $1 < p(k) < n_m$

maps a current material index k onto a new material index. In a simple implementation, $p(k)$ can be chosen to produce a random number between 1 and n_m. Our more advanced implementation permutes material assignments while maintaining a similar cost (Figure 5.6). To

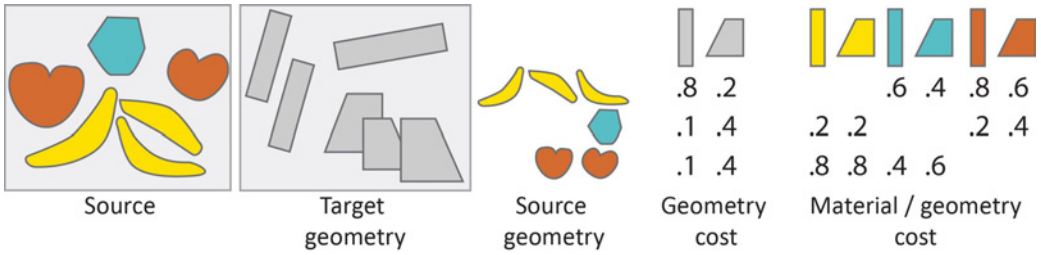

Figure 5.6: Left to right: Source image and target 3D scene; the cost matrix of assigning materials to geometry; the cost $p(k)$ of permuting a material k

find similar, i. e., nearby, materials, the metric of Pellacini et. al. [2007] is used; it computes the average intensity difference over a high number of random incoming and outgoing sampling directions. To accelerate the AS optimization, we can write the distance between all materials as a $n_\mathrm{m} \times n_\mathrm{m}$ matrix $\mathtt{M}$. For each material, a row $\mathtt{M}_k$ in this matrix represents the similarity to all other materials. In the matrix $\mathtt{F}$ a row

$$\mathtt{F}_k := (\sum_{l=1}^{l=n_\mathrm{m}} \mathtt{M}_{k,l})^{-1} \mathtt{M}_k$$

is a probability density function. Finally,

$$f_k(x) := \mathrm{argmin}_l \sum_{m=1}^{m=l} F_{k,l} > x$$

is a cumulative density function. Using f, another random number $0 < \xi' < 1$ allows us to choose a similar material $p(k) := f_k(\xi')$ with a probability proportional to its similarity.

5.2.5 Implementation Details

Evaluating the cost function – which involves rendering the scene and computing global illumination for every sample – naïvely, is far too costly. A GPU in combination with pre-computation is used to accelerate this computation. First, a deferred buffer [Saito and Takahashi 1990] storing position, normal and material ID for all views is pre-computed. As described above, we pick a random view for every sampling. We use 4 views, each in a resolution of 256×256 pixels. At runtime, only the assignment of materials to material IDs in the buffer is changed. It is done in parallel over all views and all pixels using a simple shader program. To simulate light transport, Instant Radiosity [Keller 1997] is used. 256 virtual point lights with low-resolution shadow maps have shown to provide sufficient accuracy here. The shadow maps for all VPLs are independent of the samples and can be pre-computed as well.

5.3 Results

Results produced by our system are shown in Figure 5.7. In the first and second rows, kitchen scenes are stylized, notice how different guide sources result in different moods (column). In the third row, the different combinations of specularity and texture frequency from the stones in the guide source are captured and transferred to the target. The scene shows a strong resemblance to the input photo despite the process being fully automatic. In the fourth row, even though the scene is mostly lit by indirect lighting, our system successfully captures the guide images' mood. While in Figure 5.12 a guide painting is used, the construction of our Phong BRDF ensures that materials stay realistic. Figure 5.8 uses a guide video. Our algorithm managed to detect the round shape of the juggling balls and associated them adequately to the scene. The results presented are produced without human intervention and computed in under 2 minutes.

Our approach can also serve as the basis of a rough user sketch-based system (Figure 5.9). By analyzing the drawn shapes, similarly shaped 3D objects are attributed the material that is indicated by the user. By drawing specularities, the user can further modify the specular coefficient. The process is interactive and the transfer on target scenes can facilitate material attribution in 3D scenes. One can handle all similar objects can at once, or restrict modifications to a given target view.

When our system is presented a guide source where ground truth material information is available, it successfully recovers those materials to a large extent (Figure 5.10). The two additional materials in our reconstruction, are due to shadows that were segmented into additional individual materials.

5.4 Discussion

Perceptual Study To validate the performance of our approach, a perceptual study was performed in two steps. In the first step *(assignment tasks)*, human performance in terms of quality and speed of assigning materials to a 3D scene according to a guide source was analyzed. Using the free software "Blender", 14 subjects were asked to assign materials to a 3D scene according to a guide image. Both expert (computer graphics graduate students, faster half) and non-expert (university graduate students, slower half) participated. The subjects received instructions and training on how to perform assignments and their results were captured after finishing (usually 20 minutes). There was a total of 13 assignment tasks and on average every subjects performed 4 assignments. Figure 5.11 shows the assignment task and several results edited by the subjects. At the end of the study, when asked "How pleasant was the material assignment tasks?" the common answer was "Conveying mood using material assignment is not intuitive, especially in case of dominant indirect lighting". There were a couple of comments indicating that material assignment task was really difficult, such as "I think, my results were not good enough but had no idea how to improve them anymore, searching for the right colors was so difficult and tedious".

In a second step *(ranking tasks)*, 20 other subjects were asked to rank the result images obtained in the first step. For each of the 13 assignment tasks we grouped our result and

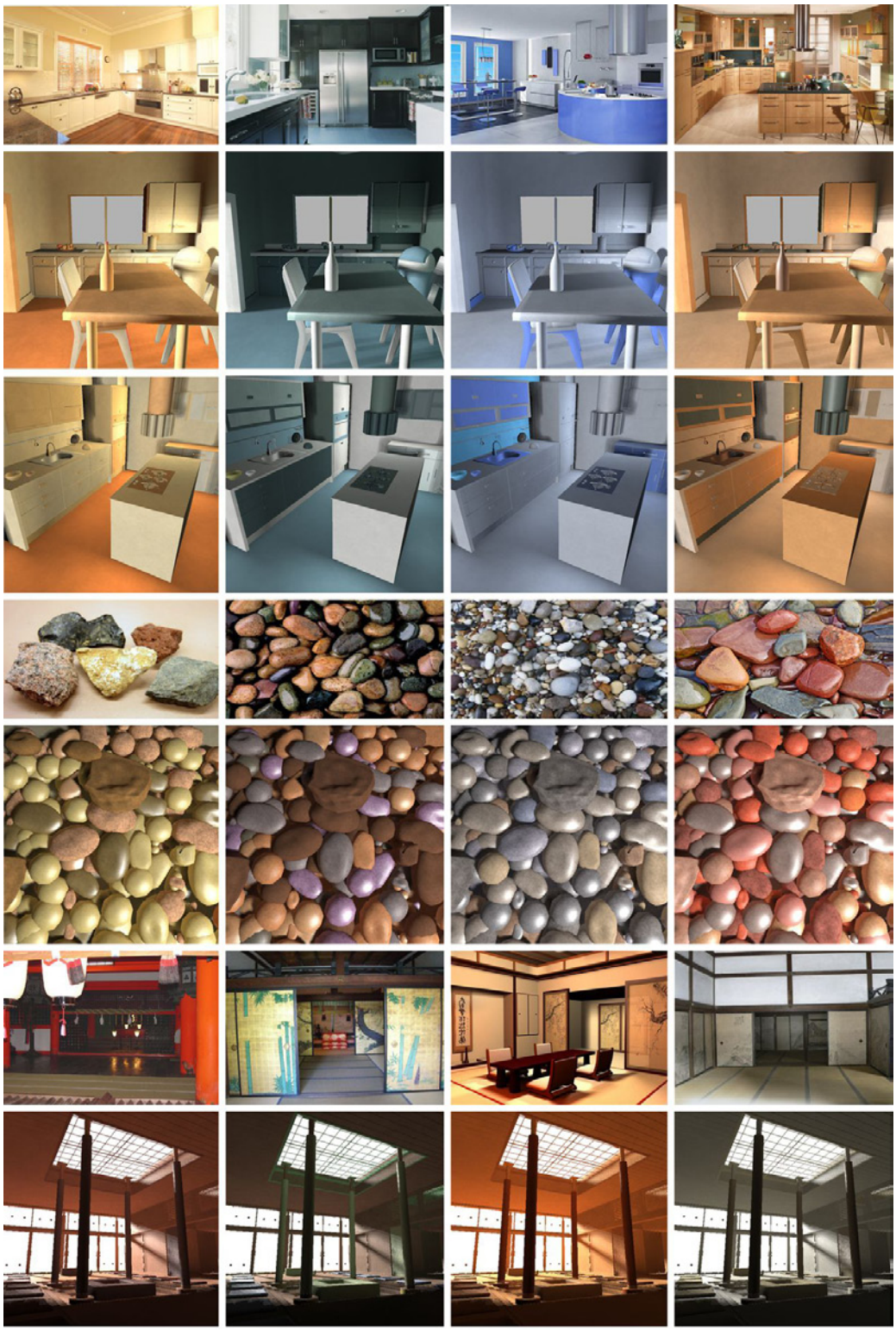

Figure 5.7: 3D Material style transfer to different target scenes *(2nd and 3rd, 5th, and 7th rows)* from different sources *(1st, 4rd, and 6th rows)*.

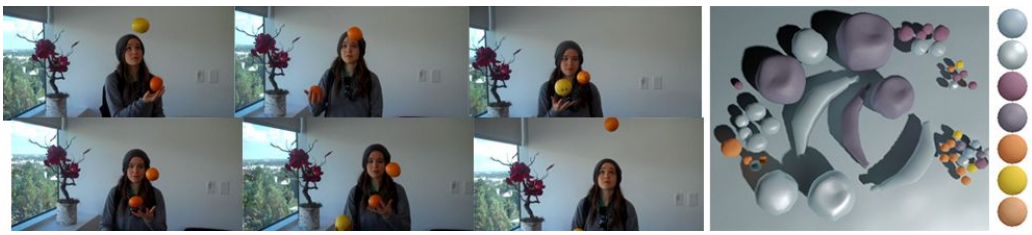

Figure 5.8: Transfer from a video *(left)* to a target 3D scene *(right)*. Note how the juggling balls are mapped to the small spheres.

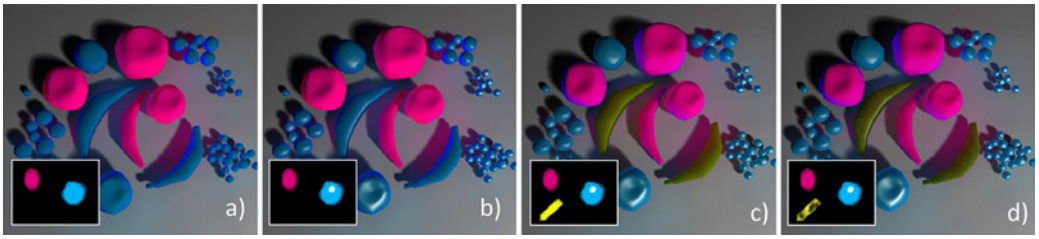

Figure 5.9: Four steps of user interaction. *(a)* A user draw two spheres in two colors, being mapped to distinct shapes. *(b)* A highlight is added to the blue patch, resulting in highlights to appear on blue shapes in the target. *(c)* An elongated yellow shape is added that is mapped to the banana. *(d)* Noise on the yellow shape makes the banana appear textured.

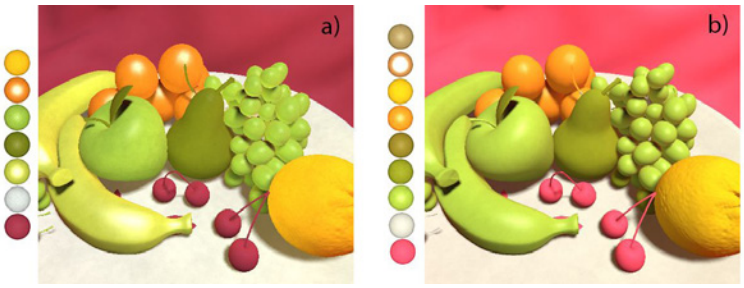

Figure 5.10: *(a)* Our material extraction applied to *(b)* a rendering of a 3D scene where ground truth materials are known.

all results of the manual assignment after 5 minutes (avg. group size: 5 images). Subjects were then asked to "sort the images from *best* to *worst* in order of mood similarity to the guide image". Our algorithm can produce an assignment in less than 2 minutes, however, 64 % rank our automatic assignment best (84 % best or second-best). Asking the same question for a different group of images that contained our result and all results of the final assignment, 60 % rank our automatic result better than all other assignments (80 % best or second-best). One conclusion from the relatively low improvement between 5 and over 20 minutes achieved by manual assignment is that, it is very hard to converge from an acceptable initial result to a final, global illumination-compatible assignment that captures the scene's mood such as produced by our system. Figure 5.13 shows several assignments that we asked subjects to perform the ranking task.

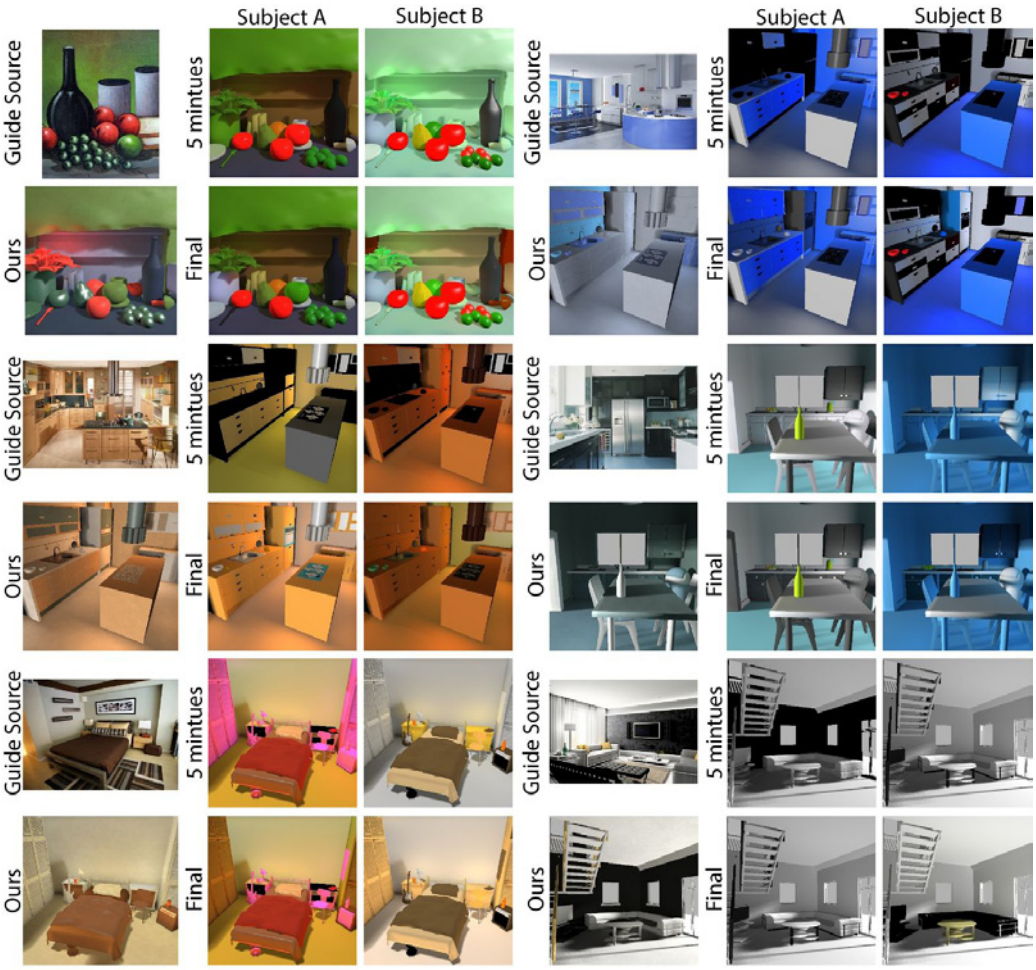

Figure 5.11: Our *assignment tasks* and several results obtained from the subjects. In every set, the first column shows the guide source *(1st col., 1st row)* and the result using our system *(1st col., 2nd row)*. The second column and third column show the result of two arbitrary subjects after 5 minutes *(1st row)* and the final assignment *(2nd row)*.

Figure 5.12: Comparison between using our combination of image and geometry cost *(left)*, using only the image cost *(middle)* and using only the geometric cost *(right)*.

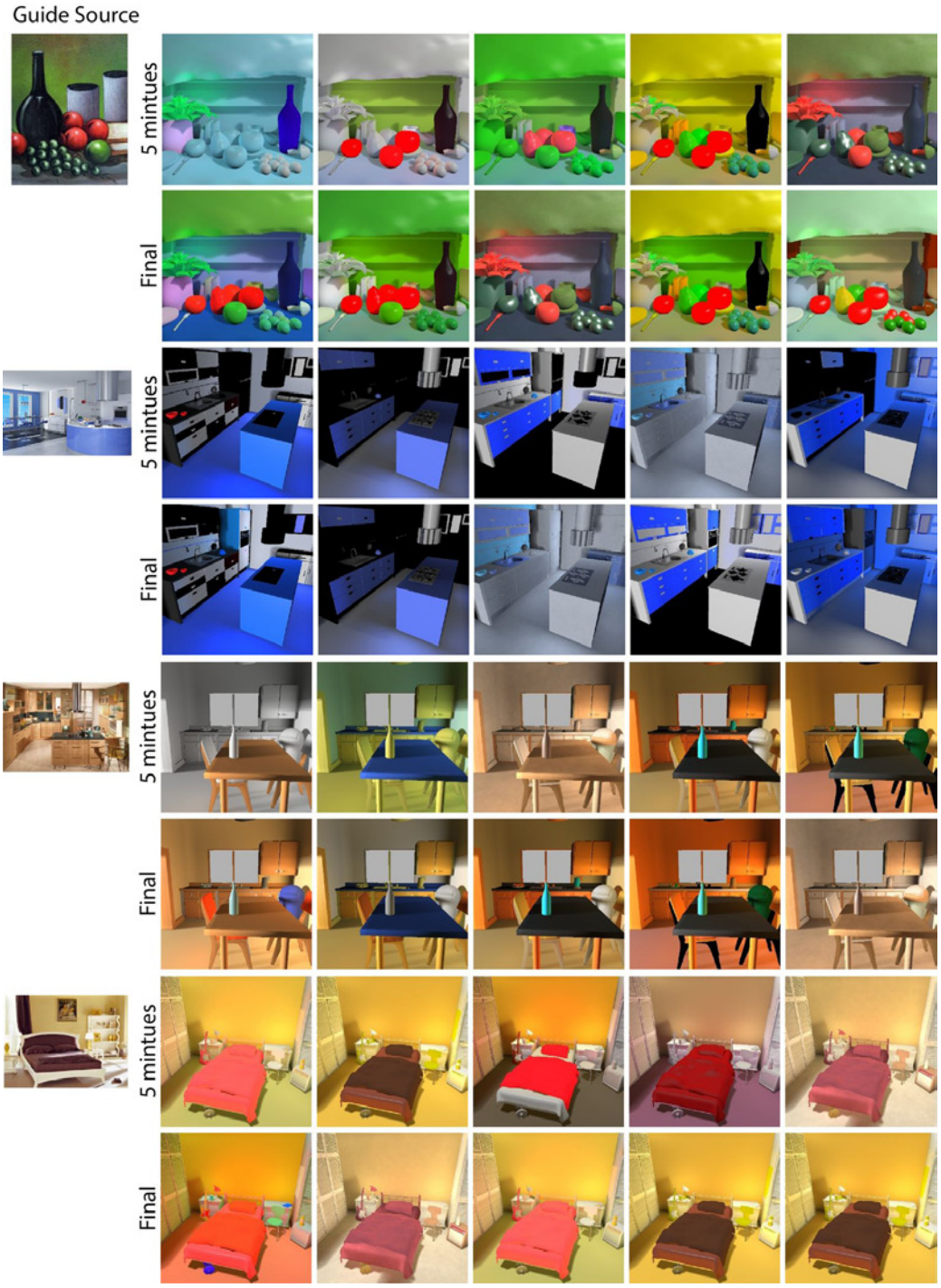

Figure 5.13: Several assignments showed to the subjects in the *ranking tasks*. In every set, on the top left is the guide source. Next are two different sets: the 5 minutes *(1st row)* and final assignments *(2nd row)*.

Image and Geometry Cost Our cost function consists of an image and a geometry term. Both play an important role and only the combination of the both aspects will lead to a successful assignment results (Figure 5.12). Using only the image term will give a similar image appearance, but wrong individual object materials. Only using the geometric term will assign the right materials to objects, but with no consistent global organization.

Limitations and Assumptions While the system often succeeds in transferring the mood from a source image onto a 3D scene, it is subject to several limitations. In summary, our system performs best for input images and target 3D scenes, which can be well-segmented with neutral directional lighting and similarities between source and target.

Certain assumptions about the image and the 3D scene have to be made. The target 3D scene has to be segmented into meaningful objects, i. e., objects that can be assigned a discrete material label. The source image has to be automatically segmented, which is a hard problem and we can not expect it to always work. While material extraction works well with imperfect segmentations, shape similarity is more sensitive. To circumvent this problem, we use multiple segmentations. If no matches are found, the geometry cost (Equation 5.2) will approach a constant for all assignments and optimizing Equation 5.1 will revert to optimizing for the image cost alone.

Many limitations stem from the difficulty to robustly extract materials from images. We focus on visually plausible materials in combination with an optimization based on similarity to a guide image. Perfectly reconstructing physically-correct reflectance from a single image is very difficult without extra assumptions on the object geometry and scene lighting that we avoid. Nonetheless, our assumption of material appearance constancy might not always hold; hard shadows or unusual highlights happen to be erroneously interpreted as individual objects. Also, textures under strong perspective transformations are sometimes grouped into different materials. Special materials like glass and other transparent materials are not yet supported.

6

Shape and Color Subspaces

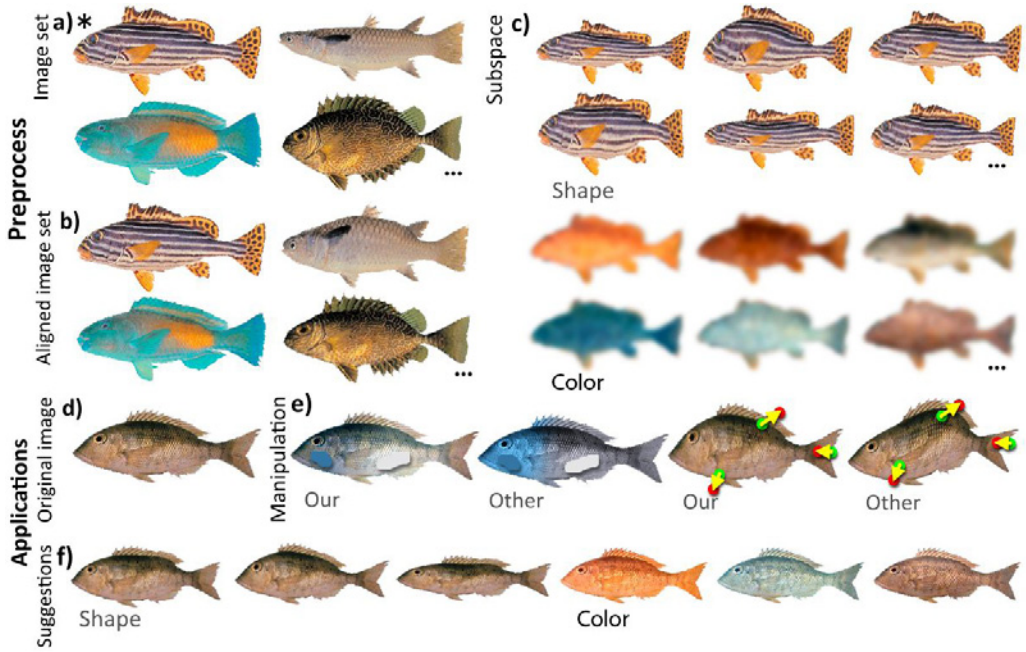

Figure 6.1: Our approach automatically aligns a set of images *(a)* showing instances of one object class to a "reference image" *(b)* and constructs a subspace of shape and color *(c)*. This subspace is used to guide manipulations of a different image *(d)*. When painting a colored stroke, our subspace is used to propagate plausible colors to plausible locations *(e, top)* or to restrict shape deformations to plausible shapes *(e, bottom)*. Additionally, the space is used to suggest relevant shape and color alternatives *(f)*.

6.1 Introduction

The ability to manipulate digital images is a fascinating opportunity taken both by professional artists producing digital content as well as by casual users editing their home photo

collection. While the option to change shape and color of an image into any possible other shape or color sounds like a good idea at first, in practice too many possible options actually decrease the human ability to make the right decision [Tversky and Kahneman 1981]. Therefore, the right balance between generality and reduction of choices has to be found. We devise a computational way to automatically suggest such choices for a class of images.

One option to restrict manipulations in a meaningful way is the construction of subspaces within the space of all possible images. Here, images are understood as points in a high-dimensional space. Images from a certain class, such as faces, do not cover the entire space but a lower-dimensional manifold which is likely related to the human mental representation of this class [Seung and Lee 2000]. This idea was first proposed for human faces, both in 2D [Turk and Pentland 1991; Cootes, Edwards and Taylor 2001] and 3D [Blanz and Vetter 1999], 3D human bodies [Allen, Curless and Popović 2003] or other specialized 3D shapes [Cashman and Fitzgibbon 2013], where the manifold is approximated using principal component analysis (PCA).

However, to construct subspaces, training data needs to be aligned manually by means of careful selection of a template and intervention [Cootes, Edwards and Taylor 2001], such as clicking correspondences [Blanz and Vetter 1999]. This excludes casual users, such as a hypothetical biologists seeking to create a subspace of leaf images for a class of plants captured in a collection of 1000 non-calibrated images or even non-professional users striving to create a subspace of a special breed of dogs from a dozen of images. No simple and efficient way exists to align a large collection of images allowing for the creation of subspaces.

In this work, we propose a simple alignment strategy applicable to casual image collections such as images of butterflies. As finding perfect or even sufficiently correct correspondences for all images in a large set is infeasible, we first build an incomplete partial alignment that is later completed to align all images.

Our key application is shape and color manipulation in images as seen in Figure 6.1. Selecting the right color for a fish and assigning it to the right spot is difficult as the selection is from a high-dimensional appearance space. While selection of color alone is already challenging, selecting shape and respecting the combination of shape and color are even harder. Our system allows the user to change color or shape in real-time and the result is restricted to meaningful changes by finding a close image in the subspace corresponding to the image class. If desired, changes in color and shape can be locked to result in correlated changes of both. As the images forming a subspace lack detail, our approach only captures the change in shape and color and transfers this change to the image being edited.

6.2 Our Approach

Overview After acquiring a set of images, our approach proceeds in three steps: alignment of example data (Section 6.2.1), construction of a subspace from the aligned images (Section 6.2.2) and application to novel user interfaces (Section 6.3). The first two steps are performed offline while our user interfaces always provide real-time feedback. Table 6.1 and Table 6.2 show the sets, constants and general symbols used in alignment (Section 6.2.1) and subspace manipulation (Section 6.2.2 and Section 6.3) respectively.

Input of our system is a set of unaligned RGB images from one class (Figure 6.2). Those images have to show instances in roughly the same pose on a constant background. If the background is not constant, it has to be removed manually. The instances have different appearance, slightly different pose and perspective and are centered. Our example classes were collected from Internet image queries, manually removing outliers in terms of the above requirements. No other manual intervention was performed.

The core of our approach automatically aligns every image to a reference image (Figure 6.3). An alignment is a deformation field, that, when applied to the respective instance from the set, produces an image with the same color but in the shape of the reference, i. e., shape and color are factored out (Figure 6.4). From the aligned images and their deformation fields, a subspace of color and shape is created which allows for novel image manipulation interfaces. Such user interfaces constrain the result to be part of the subspace in terms of color, shape, or both.

Symbol	Meaning
	General:
a, b	Image a and b
	Alignment graph:
$d(a,b)$	Distance metric between images a and b
s	Neighborhood size of the min-pooling
ε	A small constant value to penalize longer paths
A	Pairwise distance matrix $\mathsf{A}_{a,b} = d(a,b)$
	Alignment:
$\mathbf{r}$	Searching neighborhood size
$f_{b,a}(\mathbf{x})$	The best corresponding pixel in a for pixel $\mathbf{x}$ in b
$E^{\text{dat}}_{b,a}(\mathbf{x})$	The data term of the alignment cost
l	Patch size of the data term
w_s	A spatial smoothness weighting function
σ_s	The spatial smoothness parameter of the data term
$E^{\text{mag}}_{b,a}(\mathbf{x})$	The flow magnitude term of the alignment cost
w_{mag}	Weight for the flow magnitude term
$E_{b,a}(\mathbf{x},\mathbf{y})$	The energy function of the pair $\mathbf{x}$ in b and $\mathbf{y}$ in a
$\Delta_{b,a}(\mathbf{x})$	Flow vector at $\mathbf{x}, \Delta_{b,a}(\mathbf{x}) =: \Delta(\mathbf{x})$
$c_{b,a}(\mathbf{x})$	The confidence of the flow $\mathbf{x}$ in b, $c_{b,a}(\mathbf{x}) =: c(\mathbf{x})$
κ	The curvature of $E_{b,a}(\mathbf{x}, f_{b,a}(\mathbf{x}))$
γ	A parameter to control the agreement check
$g(\mathbf{x})$	The blurred correspondence field
$w(\mathbf{x},\mathbf{y})$	The modified blur kernel
σ_d	The spatial smoothness parameter of w
σ_c	The steepness of the confidence function in w
$h(\mathbf{x})$	The locally-rigid regularization of $g(\mathbf{x})$

Table 6.1: Table of notations for alignment

Symbol	Meaning
	Subspace construction:
n	Number of pixels
m	Number of basis vectors
α	A weight to control the contribution of shape/app.
$\{\mathbf{b}_j\}_{j=1}^{m}$	The set of basis vectors
λ_j	The eigen value corresponding to the basis vector $\mathbf{b}_j$
	Shape and color manipulation:
u	An image for manipulation
i	An iterator over the number of pixels n
j	An iterator over the number of basis vectors m
$\mathbf{v}$	A manipulated image vector
$\bar{\mathbf{v}}$	The closest projection of $\mathbf{v}$ in the subspace
$\mathbf{x}$	The coordinate of $\bar{\mathbf{v}}$ in the subspace
$\mathbf{k}$	The spatial weighting vector for subspace reconstruction
μ	A scalar regulating the prior's contribution
	Color transfer:
i	An iterator over the number of pixel in the query image u
j	An iterator over a square patch around i
$\mathbf{p}_j$	The source color at pixel j
$\mathbf{q}_j$	The target color at pixel j
$\mathbf{w}_j$	Weight depending on the distance between j and i and the alpha channels
$\bar{\mathbf{p}}$	The weighted centroid of the $\mathbf{p}_j$
$\bar{\mathbf{q}}$	The weighted centroid of the $\mathbf{q}_j$
R_i	The rotational component of the color transformation at i
$\mathbf{t}_i$	The translation component of the color transformation at i
δ	A regularization parameter
	Shape and color suggestions:
n_d	The number of directions used for suggestion

Table 6.2: Table of notations for subspace construction and manipulation

Figure 6.2: Some instances from the example class “Horses”.

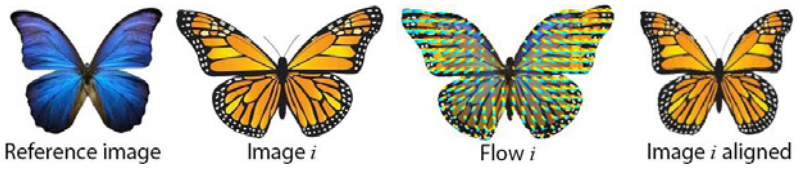

Figure 6.3: All exemplars are aligned to a reference image, here shown for the case of a butterfly class. In the 3rd column, the gradient-colored lines show the flow direction (from blue to orange) that align *(4th col.)* the exemplar *(2nd col.)* to the reference *(1st col.)*.

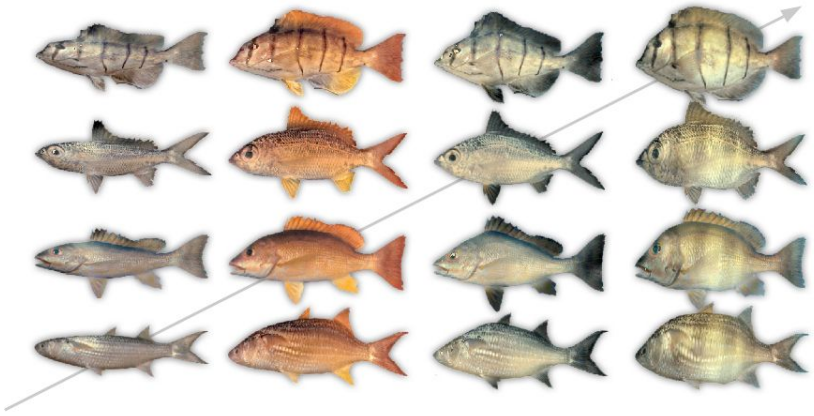

Figure 6.4: We separate shape and color of a collection of images (here shown on the diagonal) to create a space that contains arbitrary, continuous re-combinations (off-diagonal elements).

6.2.1 Alignment

Alignment is difficult as our images are not taken under controlled conditions and vary drastically in their appearance. These challenges are addressed as follows: First, we predict how well each pair of images might align. Next, from this information, we identify a reference image to which all other images might align well. The actual alignment to this reference is performed as a concatenation of simple alignments along the shortest path in a graph over the images. For each pair that needs to be aligned, we first find the best per-pixel

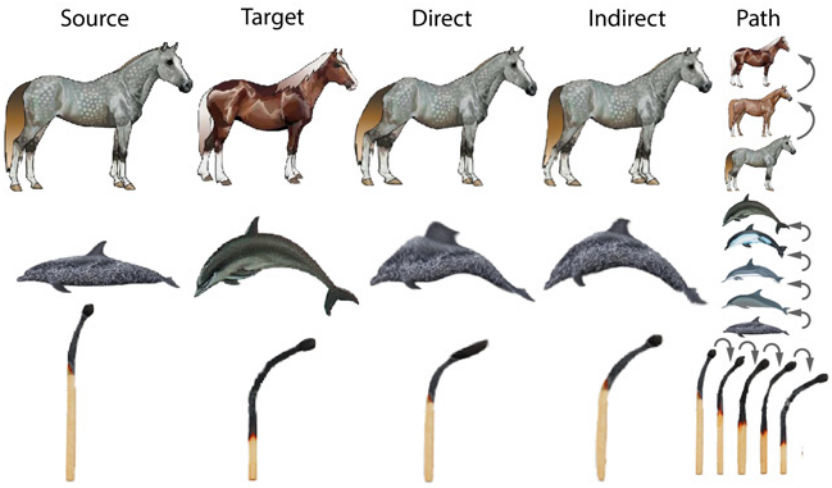

Figure 6.5: Comparison between the alignment of source and target exemplars using either direct or indirect of alignment. The alignment paths between exemplars are shown on the right.

correspondences and a measure of confidence, before we blur areas with low confidence and regularize the resulting flow to locally rigid transformations. Each step will be detailed in the following paragraphs.

Alignment Graph Directly aligning a large number of images with a reference image is likely to fail as appearance of our input images show substantial variation. Establishing an alignment between similar images however is routinely done. Regrettably, this does not suffice to align every image with the reference image as required for our needs. As a solution, we perform indirect alignment: We create an alignment graph, where similar images are aligned directly and the alignment of dissimilar image pairs is found as a sequence of edge hops (an *alignment path*) in this graph. This idea has been successfully applied to the alignment of multi-view stereo images [Huber 2002] and 3D shapes [Huang et al. 2012].

To create the alignment graph, we first define a distance metric d between images. The Gram matrix A of this metric, with $\mathsf{A}_{a,b} = d(a,b)$ holds in each entry the distance between each pair a, b.

As image distance d, a robust measure is required since common metrics such as L_2 or perceptual ones such as SSIM [Wang et al. 2004] are not resilient to the, at this point unknown, deformations. However, we would like to use a metric that reports a small difference, even if the images differ by a small deformation, as long as their appearance is similar. Conversely, a high value should be returned if the appearance is very different or the deformation is large. Let, $a, b \in [0..1]^2 \to \mathbb{R}^3$ be images encoded in the LAB-color space. We define d by

$$d(a,b) = \int_{[0..1]^2} \min_{\mathbf{y},\mathbf{z}\in[-s..s]^2} \|a(\mathbf{x}+\mathbf{y}) - b(\mathbf{x}+\mathbf{z})\|_2 \, \mathrm{d}\mathbf{x},$$

i. e., as the sum of the min-pooling LAB-color-difference in a neighborhood of size $s = 0.01$. Similar spatial pooling is believed to be used by the human visual system too and has shown to provide robustness in recognition [Serre et al. 2007].

When viewed as an adjacency matrix, A implicitly defines a fully-connected graph with the image distances as edge weights. We further add a small constant $\varepsilon = 0.1$ to each entry of A to penalize longer paths in the graph. The reference image is chosen to be the one with minimum total length of the shortest paths to all other images. To align each image a with the reference, first, the shortest path to the reference according to A is found. Then, the pairwise alignments along this path are performed as described in the next subsection and finally concatenated. Figure 6.5 shows several examples where a concatenation of alignments (indirect alignment) improves over a single direct alignment. In addition to these "backward" alignments, we analogously compute the "forward" deformation fields, aligning the reference to each respective image. The latter are later used to build the subspace of shape variations (Section 6.2.2). Note that the shortest paths from the reference to other images form the shortest-path tree (See Figure 6.6).

To improve the quality of the resulting subspace, we discard images that have a low alignment-quality, which is judged by the SSIM image difference between the reference and the backward-deformed image. Figure 6.12 shows the forward and backward alignments for several classes.

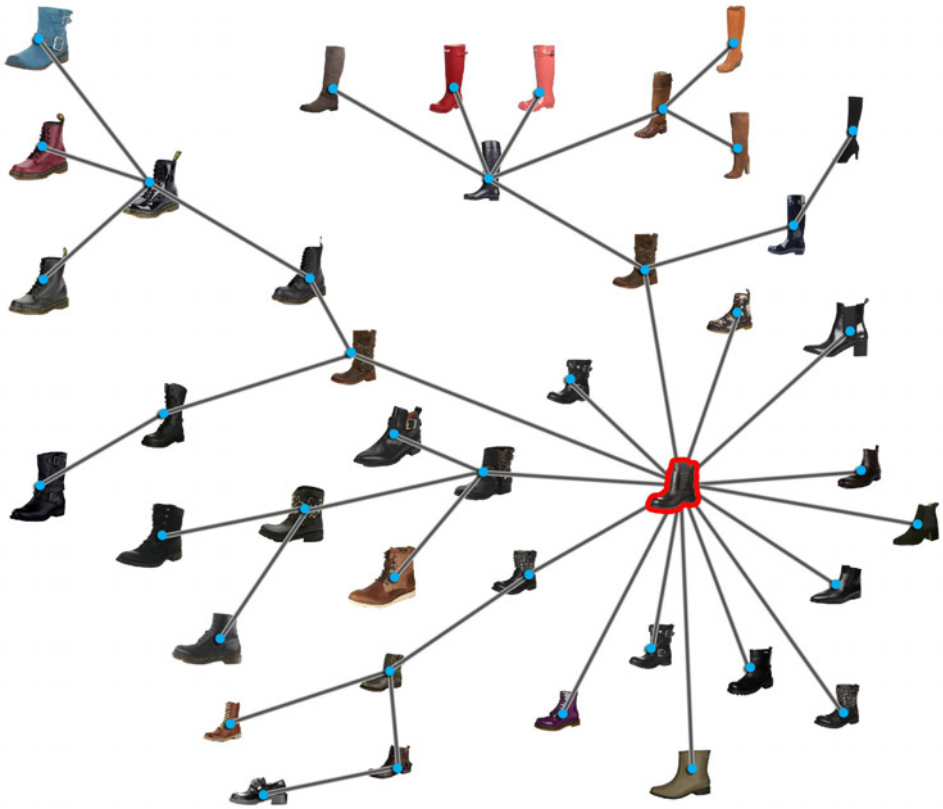

Figure 6.6: The shortest-path tree of the "boot" class formed by the shortest paths from the reference (highlighted in red) to all other images in the class. Every node *(blue circle)* represents an image of the class and the undirected edges *(gray lines)* represent the connection between two nodes. An alignment path from the reference to another image is defined by walking through intermediate nodes using the edges in the graph.

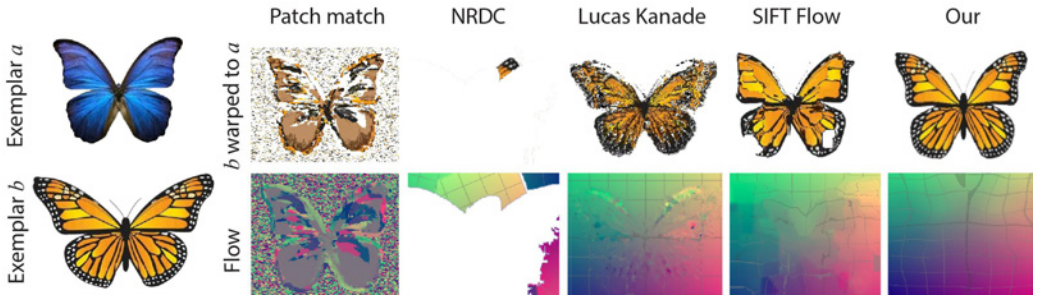

Figure 6.7: Alignment of exemplar b to exemplar a *(1st col.)* using different methods. As our exemplars differ drastically, both, PatchMatch [Barnes et al. 2009] *(2nd col.)* and NRDC [HaCohen et al. 2011] *(3rd col.)*, failed to produce a reasonable flow. Optical flow methods based on the assumptions of small disparity and image similarity such as Lucas Kanade [Lucas and Kanade 1981] *(4th col.)* or Simple Flow [Tao et al. 2012] are not designed to work for our problem. SIFT Flow [Liu, Yuen and Torralba 2011] *(5th col.)* allows robust matching between objects of different appearance at the cost of only piecewise smooth flow. Our method *(6th col.)* works best as it is specifically designed for images containing a single object.

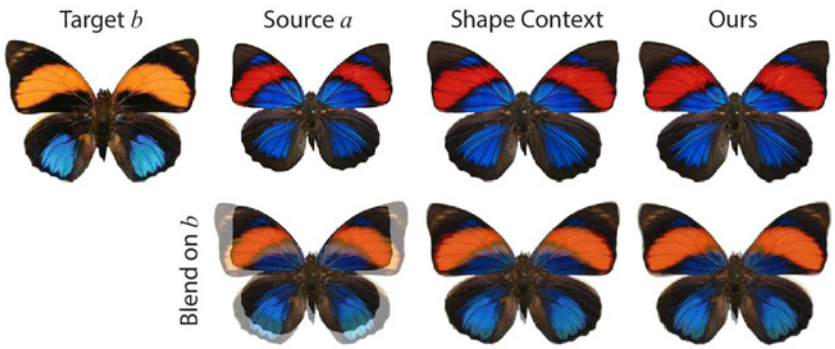

Figure 6.8: Alignment of a source *(1st row, 2nd col.)* to a target image *(1st col.)* using Shape Context [Belongie, Malik and Puzicha 2000] *(3rd col.)* and our approach *(4th col.)*. The second row shows blends of the respective image in the first row and the target image. Note how our method improves the alignment of interior structures, e. g., of the red stripe on the wings of the butterfly to the orange one, while Shape Context only aligns the outer boundary.

Alignment We seek to find a flow field that is smooth and aligns the images well but cannot assume it is produced by a simple camera motion, even if it was 3D such as in alignment for structure-from-motion. Possible techniques to deal with such problems are SIFT Flow [Liu, Yuen and Torralba 2011], Patch Match [Barnes et al. 2009] or NRDC [HaCohen et al. 2011]. We found however that, while those techniques are good at aligning images that are different in a plausible way, this comes at the cost of creating flow fields that are often close to meaningless. E. g., in the case of our butterflies they are more successful than our approach to transfer color from one exemplar to another, but at the price of a puzzle-like flow field that reassembles image a using image b in a piecewise smooth manner (Figure 6.7). Silhouette-based matching [Belongie, Malik and Puzicha 2000; Cheng et al. 2010] might succeed to align boundaries (Figure 6.8) but fails to align internal structures (Figure 6.9). Finding a plausible flow is a key requirement to capture the shape variation underlying our data, though.

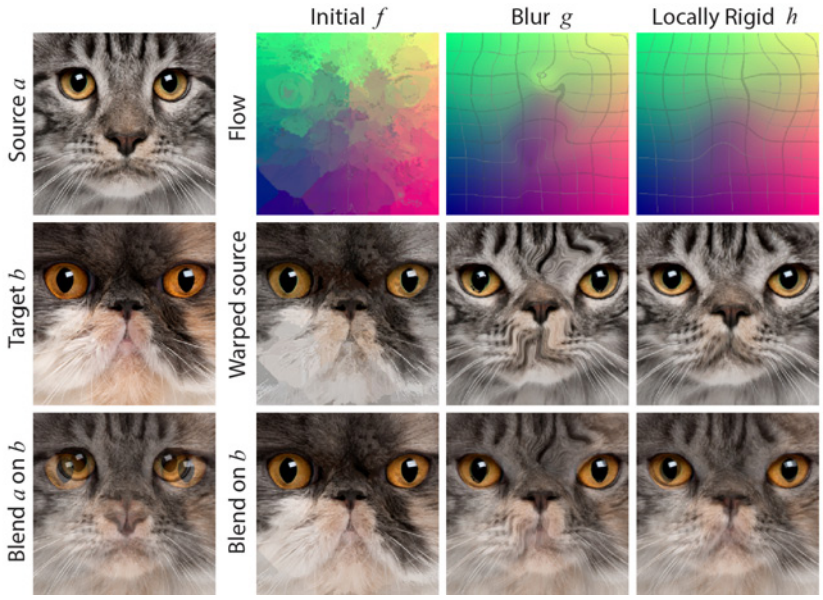

Figure 6.9: Given source and target images with different appearance *(1st col.)*, we compute an initial flow and its confidence *(cf. Figure 6.10, 3rd col., bottom)*. This initial flow field is blurred using the confidence as guidance and regularized to enforce locally rigid transformations *(1st row, 2nd to 4th col.)*. The second row shows the warped source using the respective flow from the first row. The third row shows the blends between the warped source and the target.

To this end, we devise the following alignment (Figure 6.9) between two images a and b which is computed for each pixel independently and in parallel: Each $\mathbf{x} \in [0..1]^2$ in b searches over a two-dimensional neighborhood of size $\mathbf{r} \in [0..1]^2$ in a to find the best corresponding pixel $f_{\mathrm{b,a}}(\mathbf{x})$, i. e., the one with the lowest cost $E_{b,a}$:

$$
\begin{aligned}
f_{b,a}(\mathbf{x}) &= \arg\min \mathbf{y} \in [\mathbf{x}-\mathbf{r}, \mathbf{x}+\mathbf{r}] E_{b,a}(\mathbf{x},\mathbf{y}), \qquad \text{with} \\
E_{b,a}(\mathbf{x},\mathbf{y}) &= E^{\mathrm{dat}}_{b,a}(\mathbf{x},\mathbf{y}) + w_{\mathrm{mag}} E^{\mathrm{mag}}(\mathbf{x},\mathbf{y}), \\
E^{\mathrm{dat}}_{b,a}(\mathbf{x},\mathbf{y}) &= \frac{\int_{[-l..l]^2} w_s(\mathbf{z}) \left\| b(\mathbf{x}+\mathbf{z}) - a(\mathbf{y}+\mathbf{z}) \right\|_2 \mathrm{d}\mathbf{z}}{\int_{[-l..l]^2} w_s(\mathbf{z}) \, \mathrm{d}\mathbf{z}}, \\
E^{\mathrm{mag}}(\mathbf{x},\mathbf{y}) &= \|\mathbf{x}-\mathbf{y}\|_2 \,/\, \|\mathbf{r}\|_2
\end{aligned}
$$

where $E^{\mathrm{dat}}_{b,a}$ is the data term defined as the sum of the weighted LAB-color difference of the corresponding patches of size l in a and b, $w_{\mathrm{s}}(\mathbf{z}) = \exp(-\sigma_{\mathrm{s}} \|\mathbf{z}\|_2^2 / (2l^2))$ is a spatial smoothness weighting function, controlled by σ_{s} [Tao et al. 2012]. The flow magnitude term $E^{\mathrm{mag}}(\mathbf{x},\mathbf{y})$, weighted by w_{mag}, constrains the flow vectors to be as small as possible.

Besides the corresponding pixel's location $f_{b,a}(\mathbf{x})$, we also compute a measure of confidence $c_{b,a}(\mathbf{x})$ as the product of two factors: The first is the curvature κ of the energy function $E_{b,a}(\mathbf{x}, f_{b,a}(\mathbf{x}))$, a common indicator of reliability in stereo correspondence problems [Tombari et al. 2008]. The second factor is an agreement check, to test if the flow from a to b is the same as the flow from b to a: $(1 - \|\Delta_{b,a}(\mathbf{x}) + \Delta_{a,b}(f_{b,a}(\mathbf{x}))\|_2 / (2\|\mathbf{r}\|_2))^{\gamma}$, where γ is a parameter to control the function and $\Delta_{b,a}(\mathbf{x}) = f_{b,a}(\mathbf{x}) - \mathbf{x}$. We then remove all areas with confidence below a certain threshold. Figure 6.10 illustrates the effect of the

two factors and the thresholding on our alignment. For brevity, we denote $\Delta_{b,a}(\mathbf{x})$ by $\Delta(\mathbf{x})$ and $c_{\mathrm{b,a}}(\mathbf{x})$ by $c(\mathbf{x})$ in the following.

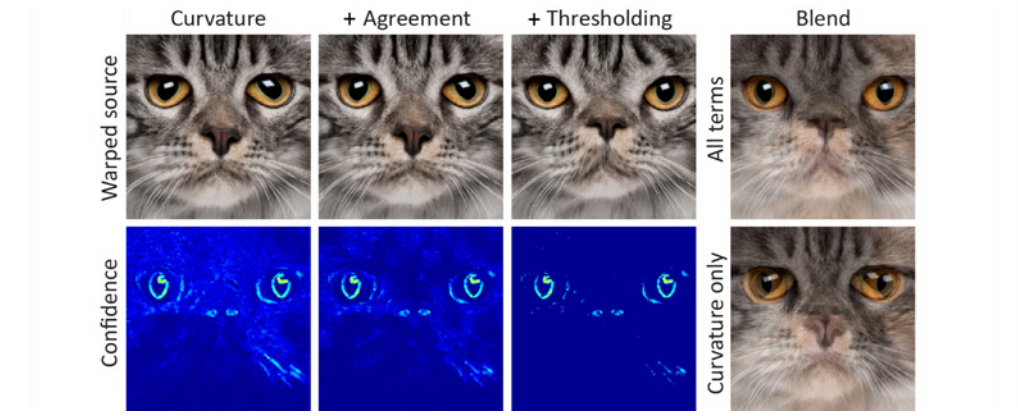

Figure 6.10: Our alignment computes the confidence from the curvature of the matching quality *(1st col.)*, by an agreement check term *(2nd col.)* and by removing low-confidence areas *(3rd col.)*. An alignment using all three terms *(4th col., top)* improves over using only the first term *(4th col., bottom)*.

The confidence $c(\mathbf{x})$ is used to replace unreliable areas of $\Delta(\mathbf{x})$ by extrapolating from more confident areas [Tao et al. 2012; Lang et al. 2012]. This is achieved by the following modified blur filter:

$$g(\mathbf{x}) = \mathbf{x} + \frac{\int_{[0..1]^2} w(\mathbf{x},\mathbf{y})\Delta(\mathbf{y})\,\mathrm{d}\mathbf{y}}{\int_{[0..1]^2} w(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{y}},$$
$$w(\mathbf{x},\mathbf{y}) = \exp(-\sigma_{\mathrm{d}}\,\|\mathbf{x}-\mathbf{y}\|_2^2)\,\exp(-\sigma_{\mathrm{c}} c(\mathbf{x})\,\|\mathbf{x}-\mathbf{y}\|_2^2)\,c(\mathbf{y}).$$

The first term of the blur kernel w is the spatial smoothness term, controlled by the parameter σ_{d}. Furthermore, the blur kernel has a two-fold dependency on the confidence c: First, pixels with a high confidence are less affected by other pixels (second term), σ_{c} controls the steepness of the function. Second, pixels with a high confidence contribute more to the value of other pixels (third term).

The resulting blurred flow field g is now smooth but might distort the shape. An improved flow h is found by mapping the flow in a neighborhood of each pixel to the closest flow in this neighborhood being a rigid transformation. This is done using the flow as point correspondences and orthogonalization [Horn 1987; Schaefer, McPhail and Warren 2006]. In discrete settings, we use a neighborhood of size 32×32 pixels for our results. The final output is a flow field that extrapolates from high-confidence areas to fill unreliable areas by a smooth and locally-rigid deformation. Figure 6.11 shows how our orthogonalization can be used to improve the flow field generated using other approaches, nevertheless, our alignment achieves better quality.

Discussion Our method is simple but efficient for the difficult problem of aligning substantially different images that undergo non-rigid deformations without manual intervention. At each step (Figure 6.9), the result for every pixel is computed independently and in parallel. In a discrete setting, for a pair of images with size 256×256 pixels, we set $\mathbf{r} = 80 \times 80$

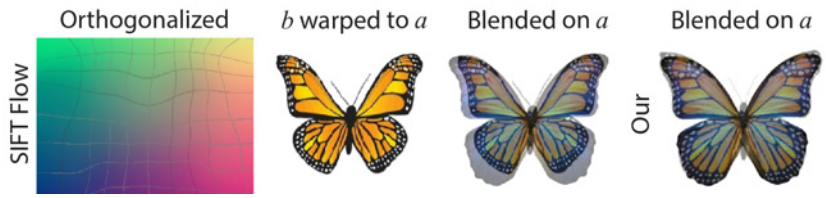

Figure 6.11: Orthogonalization of the flow field generated using SIFT Flow *(Figure 6.7, 2nd row, 5th col.)*. The two exemplars are shown in the first column of Figure 6.7. The first column shows the orthogonalized flow. Next are the warped source *(2nd col.)* generated using this flow and its blend with the target *(3rd col.)*. Finally, we show the blend *(4th col.)* of our result *(Figure 6.7, 1st row, 6th col.)* and the target.

pixels, $l = 8$ pixels, $\sigma_s = 15, w_{\text{mag}} = 0.1, \gamma = 5, \sigma_d = \sigma_c = 100$ and the computation of the deformation field takes 2 seconds. For a reef fish collection of 50 exemplars, it takes less than 3 minutes to construct the alignment graph and to align the fish along the shortest paths. This makes our method highly scalable for large image collections.

Similar to Lang et al. [2012], we use a set of confident areas to construct a flow field. However, different from their method, our initial flow and confidence are dense [Tao et al. 2012]. Furthermore, as their framework aims for scene flow, an edge stopping-based confidence propagation was exploited to improve the quality; in our case, as images come from drastically different sources, we propagate the flow solely based on the confidences. High-confidence areas, including, but not limited to, the global boundary as for shape contexts [Belongie, Malik and Puzicha 2000] are propagated to low-confidence areas. Yet, compared to boundary-based approaches, we are also able to match more general internal features (Figure 6.9) without manual intervention [Goldberg et al. 2012].

Still, our method has limitations. First, by using an alignment graph, we assume that the input contains enough image pairs with similar appearance. Second, we use rotationally variant features based on patch difference and flow magnitude. As a result, our method fails to align drastically rotated objects. Those unsuccessful alignments however can be detected using SSIM and removed from the construction of the subspace. Third, as for the construction of our subspace we do not need pixel-accurate alignment (Section 6.2.2), we focus on a simple approach aiming scalability. Further processing might be necessary for applications requiring high quality alignment. Our confidence contains many false positives; although we try to filter out those areas by an agreement check and thresholding, they strongly affect the quality of the reconstructed flow. Last, finding the initial flow is the bottleneck of our method's performance. As our images are diverse in appearance, we need to search over large neighborhoods ($\mathbf{r} = 80 \times 80$ pixels for all of our results) in a to search for $f_{\text{b,a}}(\mathbf{x})$, the best corresponding pixel in a for $\mathbf{x}$ in b. Optimizing this step would greatly improve scalability.

6.2.2 Subspace Construction

Given the appearance, in form of the aligned images (Figure 6.12, the 3rd row of every class) and shape variations, in form of the forward deformation fields, we compute subspaces using PCA [Blanz and Vetter 1999]. An image with n pixels is considered a point (*image vector*) in a $5n$-dimensional space: 3 dimensions for RGB color and 2 for the forward deformation

Figure 6.12: Alignment of different exemplars *(columns)* from several classes *(rows)*. For each class, the first column shows the reference image, the first row shows the input, the second row shows the forward alignment of the reference to the input and the third row shows the backward alignment of the input to the reference. The color below each exemplar indicates the match quality determined by SSIM, cf. the legend on the right. Smaller values indicate better alignment quality. Instances marked with a tick improved by using indirect alignment while instances marked with a cross were excluded because their image difference after alignment remained high.

field, per pixel. For the color, if the input images come with an alpha channel, the RGB values of transparent pixels are set to the average of the RGB values of all other images where alpha is non-zero.

The PCA is performed on the set of all image vectors, resulting in a set of basis vectors which, in combination with an average, captures the variation in shape and color best with respect to the L_2-norm. Each such "basis image" represents one direction of conjoint variation of color and shape.

To control the relative importance of color and shape, the respective components in the image vectors are multiplied by weights before the PCA ($0 \leq \alpha \leq 1$ for color and $1-\alpha$ for shape). Alternatively, using the extreme cases where $\alpha \in \{0,1\}$, we can also create shape and color subspaces independently. In the following, we assume that this weighting is performed before "projecting" any data to the subspace and, consequently, undone after reconstruction from the subspace by dividing by the same respective weights.

For all classes shown, we observed that the first $m = 10\,/\,20\,/\,30$ basis vectors $\{\mathbf{b}_j\}_{j=1..m}$ with the largest corresponding eigenvalues λ_j contain more than $87\,/\,97\,/\,99\%$ of the deformation and $72\,/\,85\,/\,93\%$ of the appearance variation. We keep $m = 20$ basis vectors to represent the space. A spatial resolution of $n = 32 \times 32$ was found to be sufficient to capture the amount of detail present in the data used. As the precision is limited by the alignment and the number of exemplars per class, we cannot expect fine details to be reproduced in practice.

6.3 Applications and Results

Our approach allows for a range of novel user interactions when manipulating a *query image* unknown at training time. The query image is to be aligned with the reference image, either using our approach or manually. If the image comes with a segmentation in form of an alpha matte, this matte is also used. The basic idea of all applications is to use the subspace to restrict the options a user has to interact with an image to a few plausible ones.

6.3.1 Shape and Color Manipulation

Given a user-manipulated query image u in RGB space from a known class, with a known deformation and appearance change, we can restrict the manipulation to become a plausible one by constructing its image vector $\mathbf{v} \in \mathbb{R}^{5n}$ (cf. Section 6.2.2) and finding the closest point $\bar{\mathbf{v}}$ which can be represented in the subspace using numerical minimization.

All processing happens with respect to the reference shape, so u is first warped to the reference and then down-sampled to obtain the color components of $\mathbf{v}$. The deformation components correspond to the (down-sampled) deformation field of u w. r. t. the reference image.

We adopt an idea from geometric modeling using blend shapes [Seol et al. 2012]. The image vector $\bar{\mathbf{v}}$ which we regard as the closest to $\mathbf{v}$ inside the subspace is the vector with

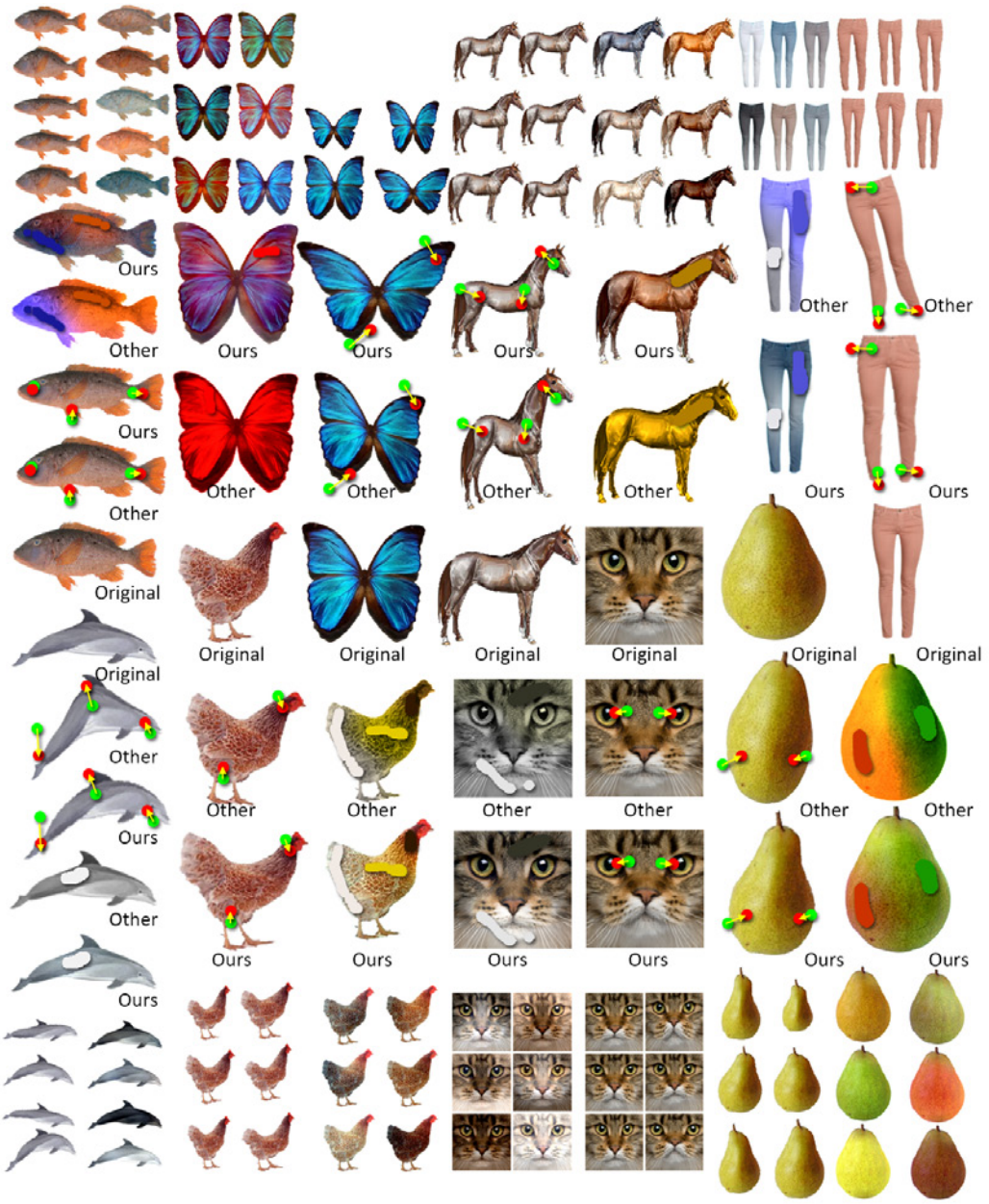

Figure 6.13: Results of our method *("Our")* and other methods *("Other")* applied to images from several classes *("Original")*. Shape manipulation is compared to Schaefer, McPhail and Warren [2006]'s rigid MLS approach, color manipulation to Gastal and Oliveira [2011]'s colorization system. For every class, we show suggestions for color and shape. Please see Section 6.3.1 and Section 6.3.2 for a discussion.

the coordinates $\mathbf{x}$ that minimizes

$$\sum_{i=1}^{5n} \mathbf{k}_i \left|\left| \mathbf{v}_i - \sum_{j=1}^{m} \mathbf{x}_j \mathbf{b}_{j,i} \right|\right|^2 + \mu \sum_{j=1}^{m} \frac{\mathbf{x}_j^2}{\lambda_j}.$$

The first term is the data term, forcing the reconstruction of $\bar{\mathbf{v}}$ to be close to $\mathbf{v}$, while the second term is a prior term that gives more weight to coordinates $\mathbf{x}$ which are plausible in the subspace [Seol et al. 2012]. The vector $\mathbf{k}$ is used for spatial weighting. For the color components, we use a 20 times higher weight at the positions of the painted strokes than for the rest of the image. This also allows us to apply our method to the colorization of gray images by setting the weight of non-stroke pixels to 0, as demonstrated in Figure 6.21. We also exclude pixels from optimization that are not part of the query image, i. e., have an alpha value of zero, by assigning them zero weight. For the deformation components, we use a smooth fall-off of the weights around the source controls of the user's constraints. The weights for both, color and deformation components, are normalized separately. Finally, μ is a scalar regulating the prior's contribution.

The resulting guidance $\bar{\mathbf{v}}$ is then transferred back to the original, high-resolution query image u as follows. First, we up-sample the color and deformation components of $\bar{\mathbf{v}}$ separately using bicubic up-sampling. The deformation, which is w. r. t. the reference image, is transformed to be relative to the instance's original shape before we apply it to the query image. As the deformations are expected to be low-frequency, no special measures are taken during up-sampling.

Color Transfer We preserve high-frequency color details of the query image u which are not captured in the subspace itself. To this end, at each pixel i of u, we find the moving least squares (MLS) solution for the rigid transformation [Schaefer, McPhail and Warren 2006] in color space that aligns the query image's colors around i to best fit those of the guidance $\bar{\mathbf{v}}$, similar to affine transformations used to transfer patch color [Shih et al. 2013].

These transformations can be computed independently and in parallel for each pixel i as follows. Let $\mathbf{p}_j, \mathbf{q}_j \in \mathbb{R}^3$ be the query and guidance pixel's colors, respectively, at pixels j in a square patch around i. Let $\mathbf{w}_j$ be Gaussian weights depending on the spatial distance of j to i, which have been multiplied by the respective alpha values at j in the query and guidance image and normalized such that $\sum_j \mathbf{w}_j = 1$. We find the rotational matrix R_i that minimizes $\sum_j \mathbf{w_j} \left\| \mathsf{R}_i(\mathbf{p}_j - \bar{\mathbf{p}}) - (\mathbf{q}_j - \bar{\mathbf{q}}) \right\|_2^2$, where $\bar{\mathbf{p}} = \sum_j \mathbf{w}_j \mathbf{p}_j$ and $\bar{\mathbf{q}} = \sum_j \mathbf{w}_j \mathbf{q}_j$ are the weighted centroids of $\mathbf{p}_j$ and $\mathbf{q}_j$, respectively. We use a polar decomposition $\mathsf{R}_i \mathsf{S}_i = \mathsf{A}_{\mathbf{pq}} + \delta \mathbf{I}$ where $\mathsf{A}_{\mathbf{pq}} = \sum_j \mathbf{w}_j (\mathbf{p}_j - \bar{\mathbf{p}}) (\mathbf{q}_j - \bar{\mathbf{q}})^\top$, similar to [Müller et al. 2005]. The parameter $\delta > 0$ controls the regularization which is necessary to avoid singular matrices. Combining R_i with the translation vector $\mathbf{t}_i = \bar{\mathbf{q}} - \bar{\mathbf{p}}$ yields the complete rigid transformation. Figure 6.14 compares our color transfer method to alternative approaches. We use a patch size of 32×32. Figure 6.15 shows how our locally-rigid color transfer behaves with different patch sizes.

Most notable, manipulations in shape lead to changes in color and vice versa. If this is not desired or required, manipulation of shape and color can also be performed independently by using a subspace created from shape or color alone.

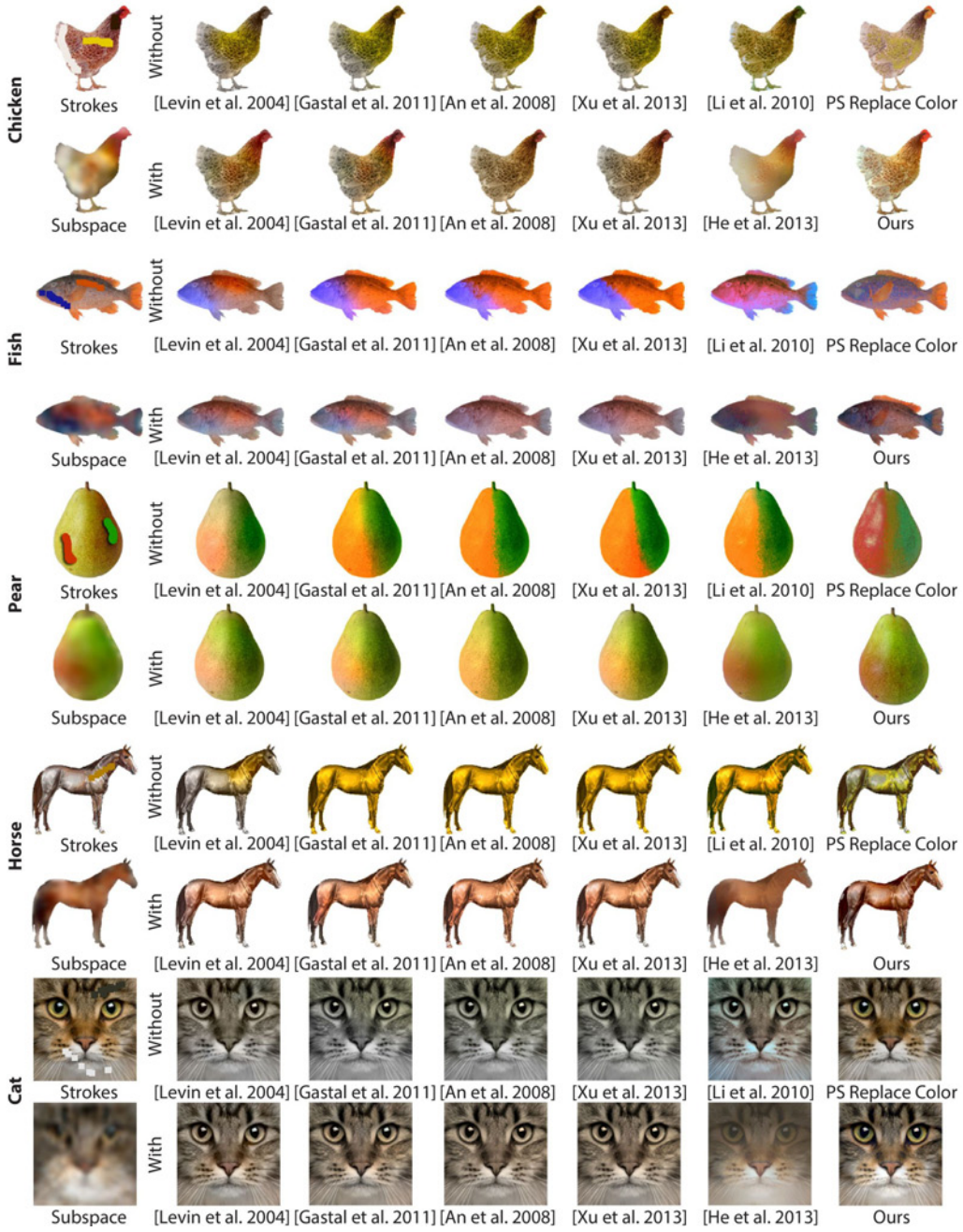

Figure 6.14: Comparison of color transfer methods. For every set, the first column shows the original with the user's strokes *(top)* and the image reconstructed from the subspace *(bottom)*. Starting in the second column, the first row shows stroke-based colorization using the methods of Levin, Lischinski and Weiss [2004], Gastal and Oliveira [2011], An and Pellacini [2008], Xu, Yan and Jia [2013], Li, Ju and Hu [2010] and Adobe Photoshop's "replace color" function, respectively. On the second row, the second to the fifth column shows colorization given the subspace guidance using the methods used on the first row, the sixth column shows guided image filtering [He, Sun and Tang 2013] using subspace as guidance and finally our locally-rigid color transfer on the final column.

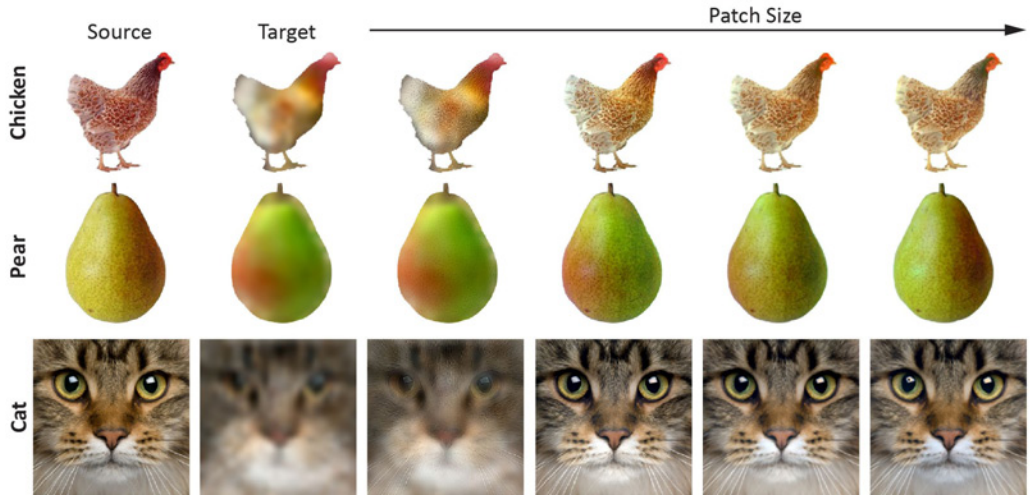

Figure 6.15: The first and second column show the original and target images, respectively. From left to right are the results of our locally-rigid color transfer with increasing patch size.

Discussion Typical results for manipulation of shape and color are shown in Figure 6.13. In the following, we will discuss a selection. The dolphin space is created from 29 exemplars. Deforming the straight dolphin into a curved version using the common approach leads to unnatural bending that is not found in our approach. The change of color in the reef fish (a space created from 50 exemplars) propagates in a plausible way to all body parts while the reference struggles with the subtle texture of the fish. Changing the posture of the chicken (40 exemplars) is easy to accomplish using only two constraints which only lead to rotation for the common approach. For the case of the butterfly, the preservation of symmetry is particularly striking when using our manipulation tools. When moving the eyes of the cat further apart, the nose will move up to keep cat-like proportions, while the MLS deformed cat appears unnatural. Assigning an unlikely color to a horse (72 exemplars), can still produce a reasonable result with our approach because of the subspace restriction and regularization we use. Using our approach, we can easily create a realistic green pear with a red spot (47 exemplars) while the common method fails to position the spot properly. From the space of jeans, we can easily produce a classic stonewashed jeans whereas the reference neither knows about the correct extent of the bright spots, nor about the symmetry. Combined manipulation of shape and color is demonstrated in Figure 6.16.

All PCA-related computations are carried out on a standard CPU while still providing real-time performance due to our modest subspace resolution. The image processing components are computed on the GPU at interactive rates.

Study We compared subspace-aware against common shape and color manipulation interfaces in a user study. Common manipulations were provided by means of Gastal and Oliveira [2011]'s interactive colorization for color and a rigid MLS image deformation [Schaefer, McPhail and Warren 2006] for shape. Color and shape manipulation tasks were performed separately. For each task, 7 subjects were asked to adjust color (by painting color strokes) or shape (by setting position constraints) of a source image to become visually

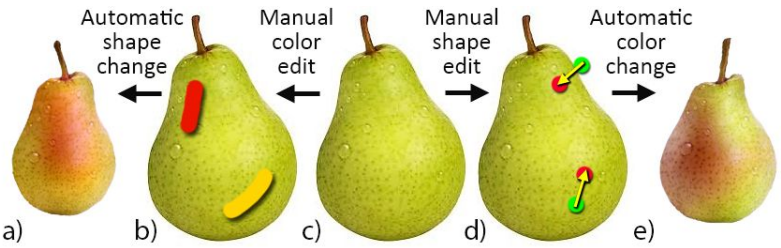

Figure 6.16: Co-manipulation of color and shape: The subspace has captured the fact that smaller pears tend to be more red than larger ones. When making a pear *(c)* smaller *(d)*, its color changes to reddish at the same time *(e)*. Conversely, painting the pear with red and yellow *(b)* will also result in a smaller pear *(a)*.

similar to a target image; once using our and once using the common interfaces. Pairs of source and target images, both not from the training set, from three classes (butterfly, pear, fish) were presented to the subjects in combination with one of the interfaces in random order. The task had to be finished either within a 20 second time budget or in an open-ended setting.

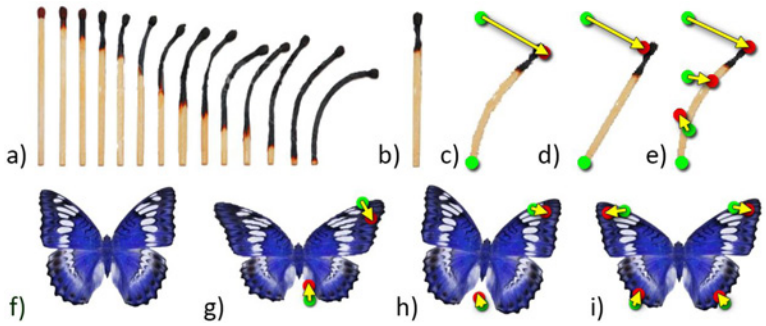

Figure 6.17: Our approach achieves meaningful deformations using a lower number of constraints. *Top row:* A subspace is created from frames of a match animation *(a)*. As we learned the most important deformation, two constraints *(c)* on the match *(b)* will be enough to curve it *(c)*. Using MLS *(d)*, more constraints *(e)* are necessary to achieve a smooth bending. *Bottom row:* Using only two constraints on a butterfly *(f)*, we achieve an interesting new shape *(g)*. For MLS *(h)*, more constraints *(i)* are required to preserve symmetry.

In the open-ended setting, for color manipulation, the average time-to-finish was $42.5\,s$ using our and $67.0\,s$ using a common interface (significant at $p < .01$, paired t-test; large effect size: Cohen's $d = 0.9$). For shape manipulation, the average time-to-finish was $40.0\,s$ (our) and $58.0\,s$ (common interface), the average number of constraints used was 2.95 (ours) and 4.57 (common interface), cf. Figure 6.17 (all significant at $p < .01$, paired t-test; large effect sizes: Cohen's $d = 0.8$ and 0.9). This indicates less effort with our approach compared to the common one.

After verifying that our interface can indeed simplify manipulation, we were interested to see if it could also improve the quality of the results. In a rating task, for each image pair produced in the manipulation study, we asked a second group of 12 subjects to choose the one they found visually more similar compared to the target image. The subjects preferred our method in 84.6 % (limited-time color), 83.0 % (open-ended color), 77.9 % (limited-time shape) and 75.8 % (open-ended shape) of the manipulation tasks. This indicates with sta-

tistical significance ($p < .01$, binomial test) that our shape and color interfaces outperform common approaches in terms of quality, too.

In a final study, we tested whether the common colorization approach by Gastal and Oliveira [2011] could be improved by using our subspace, independent of our suggested locally-rigid color transfer. The average time-to-finish was $54.0s$ for this approach. The difference to color manipulation without using a subspace is statistically significant ($p < .01$, ANOVA), indicating that our subspace indeed improves task performance for color manipulation, independent of the transfer method used. Preference ratings by subjects lead to a similar significant overall picture. Figure 6.18 and Figure 6.19 show the manipulated results achieved by one subject for the color and shape manipulation tasks, respectively. Figure 6.20 shows the results of our pairwise rating task. All differences are statistically significant ($p < .01$, binomial test).

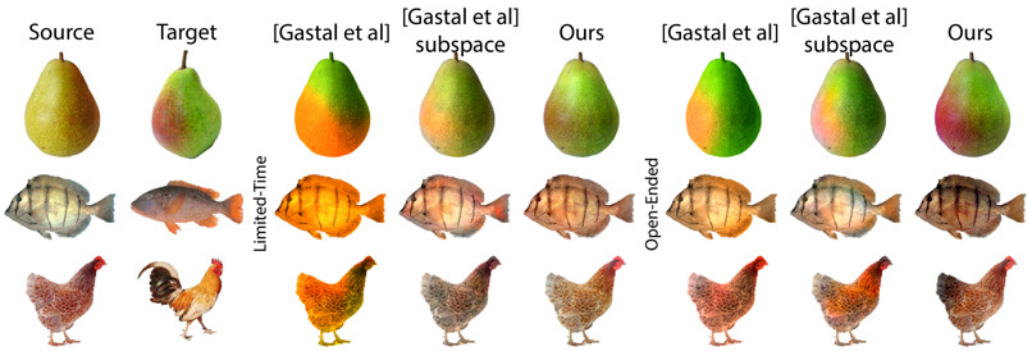

Figure 6.18: Images manipulated by a subject for the color manipulation tasks.

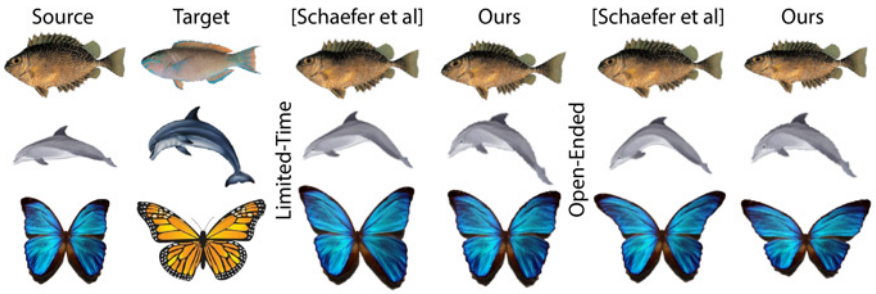

Figure 6.19: Images manipulated by a subject for the shape manipulation tasks.

6.3.2 Shape and Color Suggestions

The principal directions of variation of shape and color can also serve as suggestions for manipulations. This allows for an interface similar to "Design Galleries" [Marks et. al. 1997], just that the parameter space is found automatically. We support suggestions for both shape and color in combination or in isolation.

Suggestions are presented by showing potential positive and negative steps along the n_d most relevant eigen-directions. The number of interesting directions n_d can be determined by

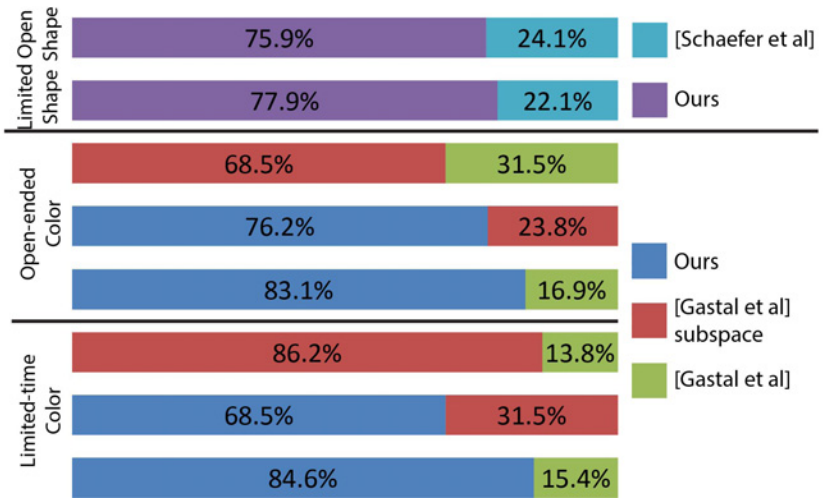

Figure 6.20: Results of the pairwise rating tasks where each row shows the percentage of subjects that preferred either interface. The top two rows show the results for shape manipulation in the limited and open-ended settings. The bottom six rows show the results of the pairwise comparison tasks for the different color manipulation methods in the limited and open-ended settings.

using heuristics based on the respective eigen values. In practice, we use the smallest number of eigen vectors such that the sum of their eigen values is larger than a given percentage, e.g., 70%, of the sum of all eigen values, adding them in decreasing order of their eigen values.

For determining a number of sufficiently diverse and specific suggestions, we found it best to use the intercept points of lines, going through the initial point along the eigen vectors, with an n_d-dimensional sphere of a certain radius, which is scaled with respect to the distance of the most-extreme image in the training set from the origin. We assume a previous normalization of the eigen vectors and values. These guidance samples are then transferred as done for manipulation and shown as altered copies of the original.

Discussion Typical results for suggestion of shape and color are showed in Figure 6.13. For the dolphin, typical shapes like jumping or bent exemplars show up as shape suggestions. The color suggestions for the given reef fish come up in a range between blue, orange and gray. The most important shape changes for butterflies turn out to affect the size and shape of the pair of wings, respectively, and are symmetrical like the butterflies themselves. For horses, their postures is an important component, as well as their height and length. The color space of pears nicely brings up all variants of more or less ripe pears while the shape suggestions produce small, large or differently tilted fruits.

6.3.3 Manipulation of Complex Images

To apply our method to complex images, a user has to provide a mask for an instance of an image class, e. g., for a dolphin in an underwater image (Figure 6.22, top). The segmented object can then be aligned to the reference from the respective class using our automatic alignment (by constructing the alignment graph and accumulating multiple alignments

along the shortest path as described in Section 6.2.1) or manual alignment and both, manual edits and suggestions, work the same as previously described. The altered instance is then pasted in, either in place of the original or as an additional element. Figure 6.22 and Figure 6.23 demonstrates both use cases. Figure 6.21 demonstrates how our subspace can also be used for colorization of gray images.

Figure 6.21: For an intial gray scale image *(a)*, a user provides a mask *(inset)* and paints green strokes to colorize the grass. For this part, Levin's colorization [Levin, Lischinski and Weiss 2004] was used. The user then defines a mask *(inset)* for the chicken and paints some color strokes *(b)*. *(c)* shows the colorization results for Levin's colorization method. *(d)* shows how our subspace-aware color manipulation greatly helps the colorization process. *(e)* shows four suggestions computed based on the masked chicken *(b)* provided by user.

6.4 Limitations

Our system is subject to several limitations. First, the use of PCA assumes that the deformation and color changes are linear. This however is only true for simple classes and invalidated in the presence of strong perspective and occlusion. We are not able to deal with multiple objects that appear in one image and always assume the image either shows a single instance or that the latter has been manually selected beforehand. To be applicable to an instance from one class in a general image, an alignment has to be available. Finally, our method is not yet ready to change arbitrary objects appearing, for example, in a home photo collection if the subspace of the object class is not available. Expanding the collection of objects would greatly increase the practicability of our method.

Figure 6.22: Subspace-supported manipulation in complex images. After a user has marked an instance of the class *(inset)*, it can be used for cloning with new shape, new color or both from the space *(1st and 3rd rows)* or to constrain color manipulation *(2nd and 4th rows)*.

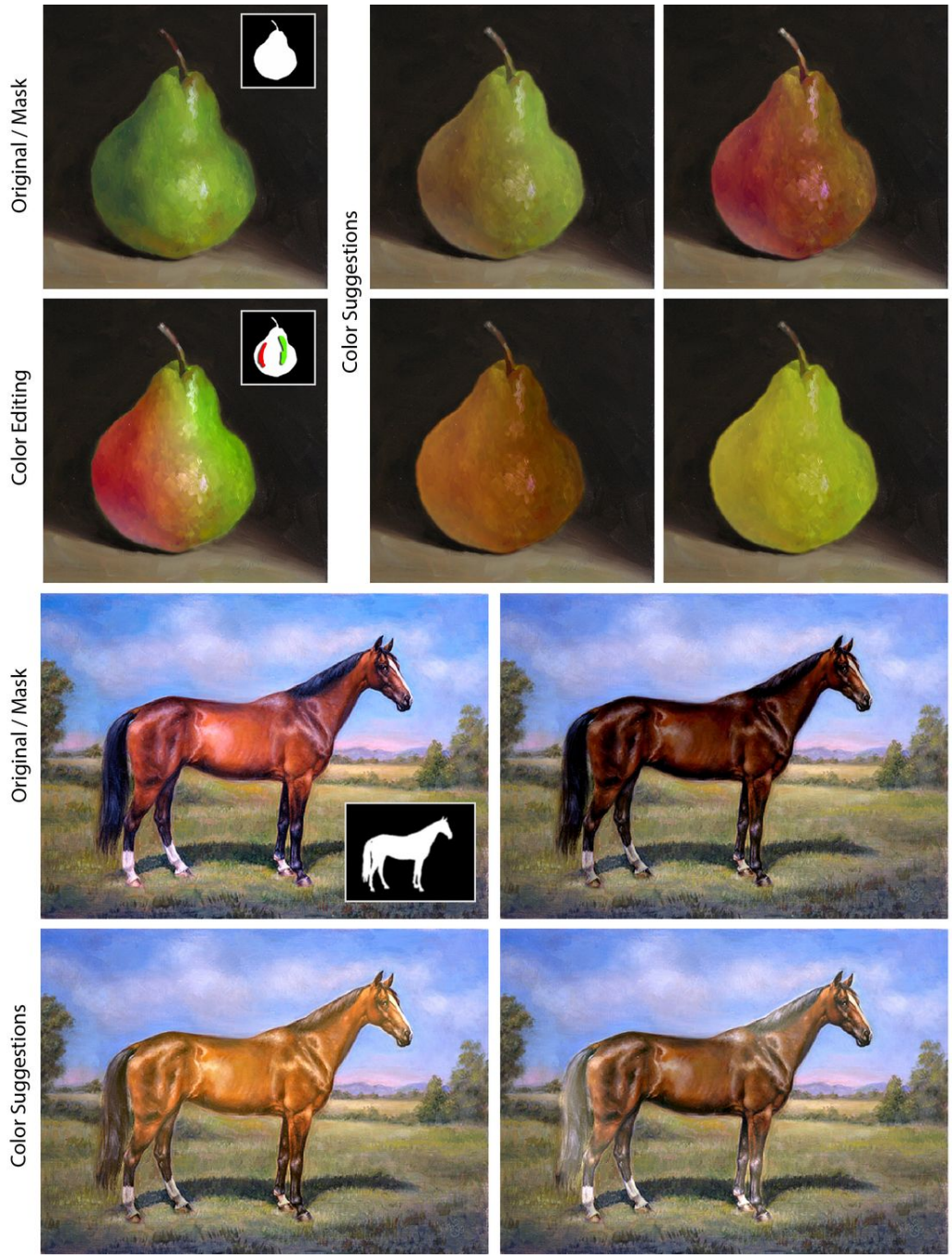

Figure 6.23: After a user has marked an instance of the class *(inset)* of an image *(1st and 3rd rows, left)*, it can be used for color suggestions from the subspaces of pear *(1st and 2nd rows)* or horse *(3rd and 4th rows)*.

7

Data-driven Color Manifolds

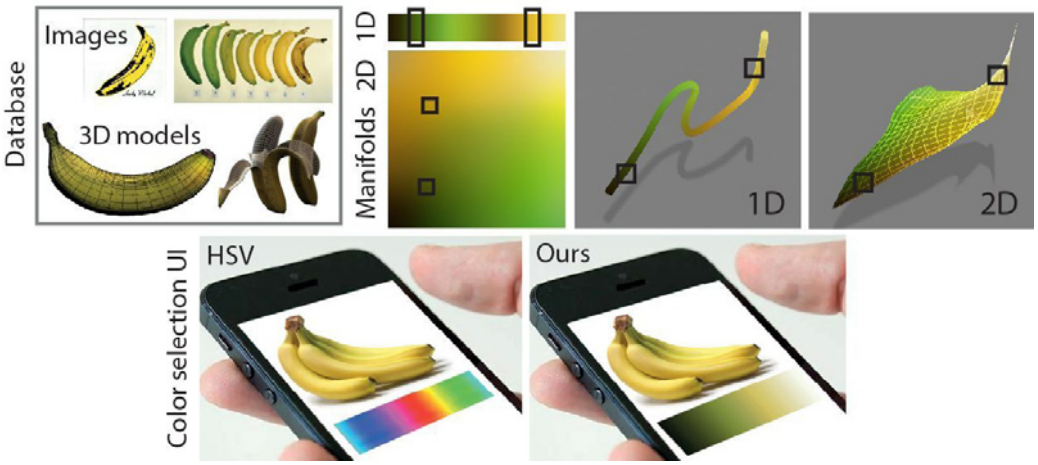

Figure 7.1: Our approach uses Internet image collections *(1st row, 1st col., only subset shown)* to learn color manifolds. The 1D manifold for the class "Banana" with a single degree of freedom *(1st row, 2nd col., top)* and as a patch for two degrees of freedom *(1st row, 2nd col., bottom)*. The same manifold shown as a 1D line *(1st row, 3rd col.)* or a 2D patch *(1st row, 4th col.)* in 3D color spaces. The same two colors are marked as squares in all visualizations. A key application is user interfaces where the manifold (here 1D) is used as a slider which show only the appropriate colors *(2nd row, 1st col.)* instead of all colors as in common color pickers *(2nd row, 2nd col.)*.

7.1 Introduction

The seemingly simple task of color selection is highly important in many computer graphics applications, ranging from casual photo manipulation to professional 2D and 3D content creation. Despite being an often-used and important operation, exploring high-dimensional colors using low-dimensional user interfaces such as linear or angular 1D slider or 2D widgets that control parameters of a certain color space [Schwarz, Cowan and Beatty 1987; Douglas and Kirkpatrick 1999] is often disappointing. Color selection using color templates (e. g., PANTONE), or more specific, hue templates [ODonovan et. al. 2011], is an alternative but

does not scale well to many colors (too large palettes) and fine-tuning of colors (too coarse palettes).

In this work, we seek to improve upon both paradigms in an example scenario as follows: A user takes a picture of a wooden chair and wants to adjust its color using a single 1D sweep or a single click on a mobile device. In any common color space, such a change is likely impossible: Going more red will require more saturation, going less red, will require less, otherwise the result is a color, but not a wood color anymore. Our approach learns this relationship from labeled exemplar data and present the user a single 1D slider to traverse the manifold of plausible wood colors. Additionally, we want the manifold to compress the color space in ranges that appear less frequently and to enlarge in frequently used areas. For the chair examples, we will enlarge the brown-beige-red areas while we shrink the green or blue areas.

The technical challenge addressed in this chapter is to project a high-dimensional color space with a density acquired from Internet data, to a lower-dimensional color space such that neighborhood of important colors is preserved, and embedded area is proportional to density.

7.2 Our Approach

We will now describe our approach to create an n-dimensional color manifold from images of a certain class represented in a $m \geq n$-dimensional color space. We experiment with values of $n = 1$ (line) or $n = 2$ (surface) and $m = 3$ (radiance, reflectance).

In future work, higher values of m could be used for spaces like spectral color; higher values of n in other applications or 3D color selection. Every manifold belongs to an image class like “sky”, “banana”, etc. We will not consider how to classify images and assume a state of the art-classifier to be correct in many cases [Lazebnik, Schmid and Ponce 2006] in combination with a simple but effective background removal. The extraction of all manifolds for each class is independent and performed for all possible classes in a pre-process step. As the manifolds are smooth, they can be serialized into a file of a few hundred bytes. Exchanging those files is sufficient for standardization, e. g., in print. As our manifold construction can be performed at interactive rates, we optionally allow the user to create a set of images and extract their color manifold on-the-fly.

Overview An overview of our approach for one class and manifold is given in Figure 7.2. First, we acquire color samples from images or 3D models returned from an Internet query (Section 7.2.1). Typically, this results in many millions of colors that are put into a m-dimensional histogram in a certain high-dimensional *source* color space such as RGB or CIE Lab (Section 7.2.2). Next, this histogram is thresholded and only the $\alpha\,\%$ most important colors are kept. On the remaining colors, we either use principal component analysis (PCA), multi-dimensional scaling (MDS) or self-organizing maps (SOM) to perform dimensionality reduction (Section 7.2.3). Table 7.1 shows the sets, constants and general symbols used in this chapter.

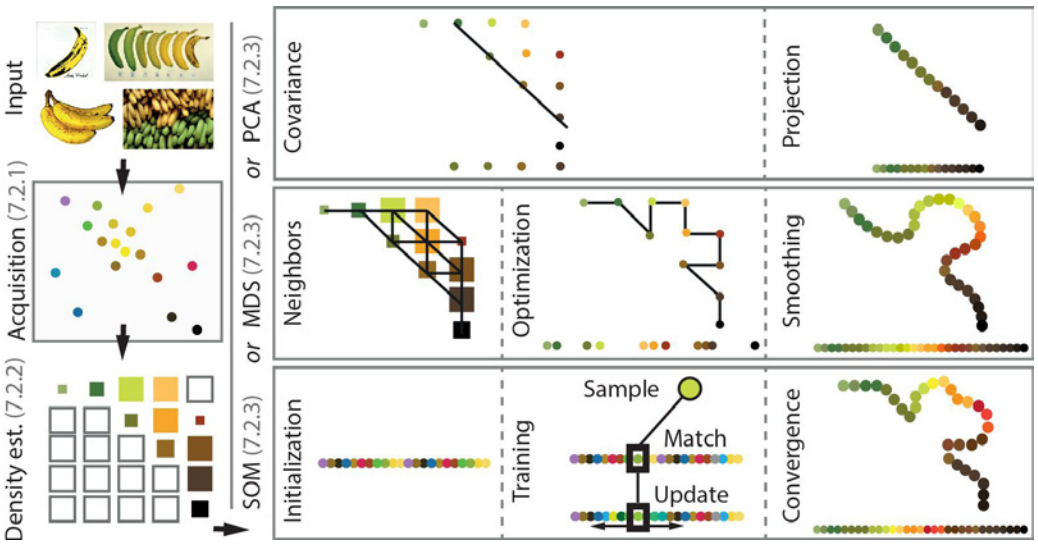

Figure 7.2: Flow of our approach: The first two steps are always identical *(left)*: Acquisition of color statistics (Section 7.2.1) and density estimation (Section 7.2.2). The dimensionality reduction is analyzed for three variants *(right)*: PCA (Section 7.2.3), MDS (Section 7.2.3) or SOM (Section 7.2.3).

7.2.1 Acquisition

Color samples are acquired using 2D Internet image search and online 3D model repositories. For images, Google Image Search is used to acquire the top 100 images for one class. Note that our input image set contains images with improper camera calibration (improper photometric calibration, incorrect white balance) and different drawing styles. Figure 7.3 shows some sample images of several classes. We do not filter this set or compensate for any different camera calibration or image style, which would likely further improve our results. We will distinguish in our results between *weakly supervised* acquisition (where images are used as they return from an Internet search) and *strongly supervised* acquisition (invalid pixels are manually excluded by an alpha mask). Optionally, users can *interactively* remove and add images. If not mentioned otherwise, all of our results are from weakly supervised sources without user interaction.

The problem with 2D images is that they are acquired under unknown illumination and contain shading. Consequently, images can be used better to study a luminance-free, 2D color space of hue and saturation. Ideally, we would like to have a repository of true reflectance data for the purpose of studying 3D color. For this, data from online 3D model repositories are used (Figure 7.4). We assume that textures of such models do indeed have reflectance of 3D models. This is justified, as artists tend to use proper white balancing, shadow removal, etc. on their textures. The only remaining difficulty is that textures contain areas with pixel values that do not map to the surface and should be excluded as they are not part of the reflectance we seek to sample. We solve this by setting alpha to zero in all textures and then draw the UV mapping polygons with alpha set to one into the alpha channel. The result is a 2D RGBA image that can be processed like other images. Unless stated otherwise, 2D images are used to study the full 3D color space and for results in this chapter, we use manifolds of radiance and not of reflectance.

Symbol	Meaning
$\mathcal{S}$	Source color space ($\mathcal{S} = \mathbb{R}^m$)
m	Input samples dimension
n	Embedded manifold dimension
k	Number of input samples
α	The percentage of bins kept after density thresholding
l	Number of compact samples ($l \ll k$)
$\mathbf{a}$	Set of input color samples ($\mathbf{a} \in \mathcal{S}^k$)
$\mathbf{b}$	Set of compact color samples ($\mathbf{b} \in \mathcal{S}^l$)
$\mathbf{d}$	Density of compact color samples $\mathbf{d} \in \mathbb{R}^l$
D	Pairwise geodesic dist. mat. of samples ($\mathsf{D} \in \mathbb{R}^{l \times l}$)
$\mathcal{T}$	Embedded space ($\mathcal{T} = \mathbb{R}^n$)
$\mathbf{t}$	Parameterized values, e. g., time, of input samples $\mathbf{t} \in \mathbb{R}^k$
	PCA-specific:
μ	Density-weighted mean of $\mathbf{b}$ ($\mu \in \mathcal{S}$)
C	Density-weighted covariance matrix ($\mathsf{C} \in R^{m \times m}$)
	MDS-specific:
$\mathbf{c}$	Set of embedded samples ($\mathbf{c} \in \mathcal{T}^l$)
$r(\mathbf{x}, \mathbf{y})$	Radial basis function kernel
s_m	Smoothness parameter ($s_\mathrm{m} \in \mathbb{R}^+$)
	SOM-specific:
h	grid resolution
W	Grid of weight nodes($\mathsf{W} \in \mathcal{S}^{h^n}$)
t	Number of training steps ($t \in \mathbb{N}^+$)
i	A particular training step ($i \in \mathbb{N}, 1 \leq i \leq t$)
s	Number of samples processed in each training step
$\mathbf{p}$	Training sample set ($\mathbf{p} \in \mathcal{S}^{s \cdot t}$)
r_0	Smoothness ($r_0 \in \mathbb{N}^+, 1 \leq r_0 \leq h/2$)

Table 7.1: Table of notations

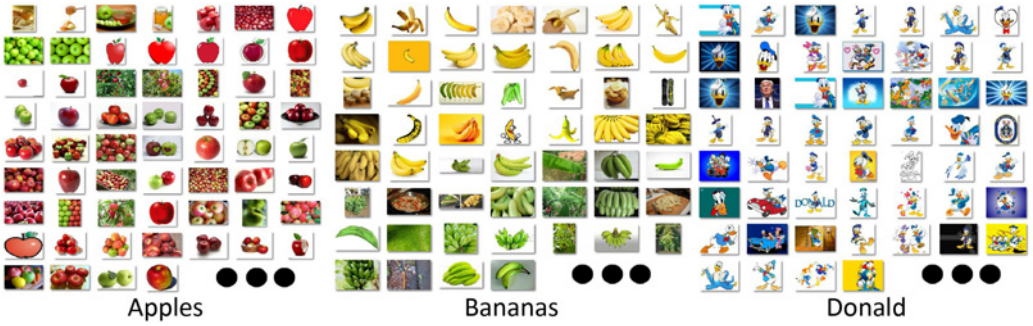

Figure 7.3: Several image classes downloaded using Google Image Search.

Figure 7.4: Several reflectance classes downloaded using Google Sketchup.

For a certain class, image search results contain a foreground object belonging to the class in front of a background. We use a simple heuristic to remove this background. First, the image is blurred using Bilateral filtering. Next, two neighbor pixels are connected only if their CIEDE2000 color difference is less than a threshold. Finally, all connected components that contain pixels on the image frame boundary are considered as background. Figure 7.5 shows several background-removed images. Our removal tends to be conservative and while it potentially removes pixels that belong to the class, it rarely keeps background pixels, as seen e. g., around the "Snow White" example.

Figure 7.5: Input images before *(1st and 3rd col.)* and after background removal where removed pixels are marked as gray *(2nd and 4th col.)*.

The acquired images are now in the RGB color space, with alpha channel used as a mask. Pixels with an alpha value of zero are skipped from further consideration. If the source color space $\mathcal{S} = \mathbb{R}^m$ should be different, a conversion e. g., to CIE Lab is performed now. Finally, we randomly sample k (e. g., four million) pixels from the m-D color space of our query data into a vector $\mathbf{a} \in \mathcal{S}^k$.

7.2.2 Density Estimation

The purpose of this step is to turn a large collection of samples that implicitly describe the frequency of a certain color into a simple explicit representation of density, i. e., color frequency. To this end, the source space is conceptually covered by a lattice of 16 bins along each dimension, into which the k sampled pixels of $\mathbf{a}$ are inserted. We call the normalized bin value the *density* of this color. We choose a regular grid instead of clustering for simplicity and to allow for an efficient (i. e., interactive-rate) construction of manifolds with a user

in the loop. Next, this histogram is thresholded and only the $\alpha\,\%$ of bins with the highest density are kept. An $\alpha = 15\,\%$ is used to produce our results. This is done to eliminate outliers that corrupt the manifold structure or colors that might not belong to the class. See Section 7.3.2 for more details on our parameter choice. The result is a set $\mathbf{b} \in \mathcal{S}^l$ of l points in the m-D source color space with a density vector $\mathbf{d} \in \mathbb{R}^l$. We clamp the density $\mathbf{d}$ between $0.1\,|\mathbf{d}|\,/l$ and $2\,|\mathbf{d}|\,/l$ to avoid under- or over-estimated color importance. While the input contained k i. e., millions of elements, the compact color point cloud $\mathbf{b}$ typically contains $l \ll k$ i. e., several hundreds of colors only.

7.2.3 Dimensionality Reduction

We explore several dimensionality reduction methods for our embedded manifolds: linear PCA (Section 7.2.3), as well as MDS (Section 7.2.3) and SOM (Section 7.2.3) which are non-linear methods.

Principal Component Analysis (PCA)

Let $\mu \in \mathbb{R}^m$ be the density-weighted mean of $\mathbf{b}$ and $\mathsf{C} \in \mathbb{R}^{m\times m}$ the density-weighted covariance matrix of the color distribution. The n eigenvectors of C with the highest eigenvalues are a linear embedding into the n-dimensional space. Our method embeds a line $(n = 1)$ or plane $(n = 2)$ in the $m-$dimensional source space with higher weight to high-density colors.

Multi-dimensional Scaling (MDS)

To embed colors using MDS, we establish color neighborhoods, optimize a layout to keep those neighborhoods and finally optimize for smoothly and completely filling the lower-dimensional space. The three steps are explained in the next paragraphs.

To extract the manifold structure, we create a graph with weighted edges. First, all colors in $\mathbf{b}$ are interpreted as nodes and an edge is created between neighboring colors / nodes. The neighborhood is still in the m-D source space and simple to find due to the regular histogram structure. For the rare case, when the resulting set of points has several disconnected components, we discard all but the largest component. Next, we label each edge with the average density $(\mathbf{d}_i + \mathbf{d}_j)/2$ of the two nodes i and j it connects. Conceptually, edges in dense areas get longer, edges in sparse areas get shorter. Finally, we approximate the geodesic distance between all pairs of nodes (not just the neighbors) by shortest paths using the Floyd-Warshall algorithm and insert it into a pairwise distance matrix $\mathsf{D} \in \mathbb{R}^{l\times l}$. This step is similar to the extension of Isomap [Tenenbaum, De Silva and Langford 2000] over MDS, but using weighted edges instead.

The distance matrix D computed in the previous step is fed into a classic MDS [Cox and Cox 2000]. The output are l new n-dimensional color points $\mathbf{c} \in \mathcal{T}^l$ $(\mathcal{T} = \mathbb{R}^n)$ that preserve the desired distance matrix D in the lower-dimensional space in the least-squares sense. As an MDS solution is unique up to a rotation and a uniform scaling, we additionally normalize it to the unit hypercube and rotate it, such that the direction of largest luminance variation (found using PCA of the luminance of $\mathbf{c}$) aligns with the first axis.

While $\mathbf{b}$ was a regular grid of a simple structure in the host color space $\mathbb{R}^m$, the embedding $\mathbf{c}$ is an irregular point cloud in $\mathbb{R}^n$. Therefore, it is not clear which, how many and if at all an element in $\mathbf{c}$ maps to any coordinate location $\mathbf{x}$ in $\mathbb{R}^n$. However, we would like to use $\mathbb{R}^n$ for smooth navigation to enumerate $\mathbb{R}^m$ in a plausible way. To reconstruct a smooth unique mapping $f \in \mathbb{R}^n \rightarrow \mathbb{R}^m$ defined on the entire domain $\mathbb{R}^n$, we employ radial basis function (RBF) reconstruction: $f(\mathbf{x}) = \sum_{j=1}^{l} r(\mathbf{c}_j, \mathbf{x})\mathbf{b}_j / \sum_{j=1}^{l} r(\mathbf{c}_j, \mathbf{x})$ where r is the kernel $r(\mathbf{x}, \mathbf{y}) = \exp(-(s_\mathrm{m}||\mathbf{x}-\mathbf{y}||)^2)$ with a constant s_m to control smoothness. Please see Section 7.3.2 for different settings of the smoothness parameter s_m.

Self-organizing Map (SOM)

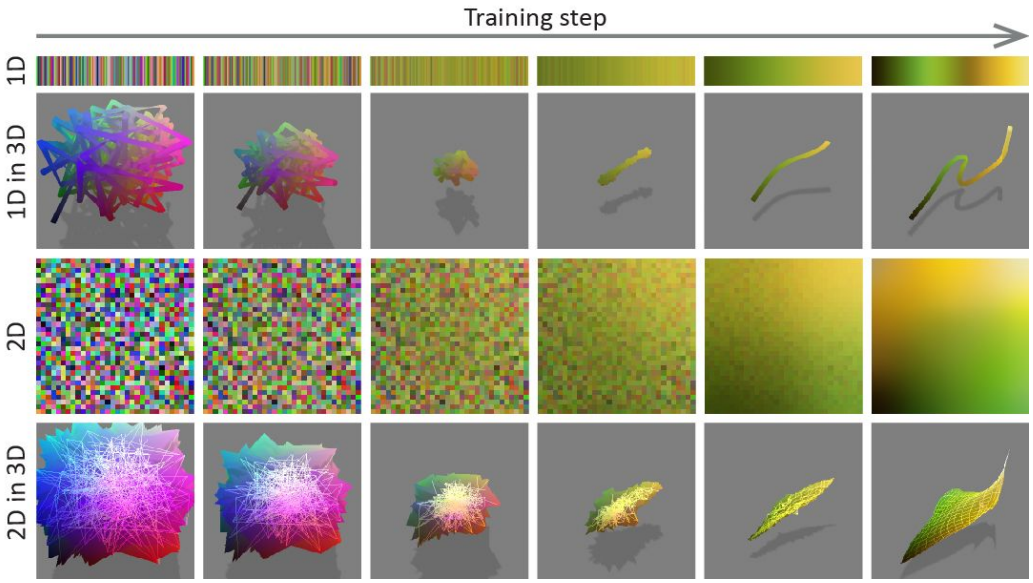

Figure 7.6: Banana manifold construction (Figure 7.1) using SOM: The first and third row show the colors in the evolving 1D and 2D manifold. The second and fourth row show the 1D and 2D color path resp. patch in the 3D RGB cube. From left to right, the manifold is refined as new samples are added.

SOM produces a non-linear mapping from the m-dimensional space to an n-dimensional grid of weight nodes $\mathsf{W} \in \mathcal{S}^{h^n}$ [Kohonen 1990], where h is the grid's size.

The nodes of W are initialized to random values and updated in t training steps. In a training step i $(1 \leq i \leq t)$, the node with the minimum distance to a training sample is called the best matching unit (BMU). The weight of the BMU and its neighbors are adjusted toward the training sample. The magnitude of change and the neighborhood size from the BMU decrease after each training step [Kohonen 1990]. Figure 7.6 shows the weight grid W during the construction of the "Bananas" manifold at different training steps.

Let $\mathbf{p} \in \mathcal{S}^{s \cdot t}$ be the training sample set where s is the number of training samples processed in each training step. Note that s equals to 1 in the previous paragraph. The training sample set $\mathbf{p}$ is constructed to contain colors from the compact color point cloud $\mathbf{b}$ with distribution propotional to the density vector $\mathbf{d}$.

In order to fully use the advantange of modern GPUs, at training step i, we process a set

of s samples $\mathbf{p}_{(i-1)s+1}, ..., \mathbf{p}_{is}$ in parallel as proposed in Batch SOM (Fig. 2 in [Lawrence, Almasi and Rushmeier 1999]). The distances between the training samples and nodes are calculated using the CIEDE2000 color difference. Nodes are then updated as described in Lawrence, Almasi and Rushmeier [1999]. To enforce smoothness on the weight grid W, we further constrain the neighborhood size to be bigger than a predefined smoothness parameter r_0 ($r_0 \in \mathbb{N}^+, 1 \leq r_0 \leq h/2$).

For the 2D manifolds ($n = 2$), we empirically set the grid size $h = 32$, $s = 64$ training samples each step, the smoothness parameter $r_0 = 10$ and $t = 2000$ training steps. For the 1D manifolds ($n = 1$), we set $h = 128$, $s = 64$, $r_0 = 15$ and $t = 4000$ respectively. Section 7.3.2 contains more details on our parameter choices.

7.3 Algorithm Evaluations

7.3.1 Algorithm Comparison

Figure 7.7 shows several dimensionality-reduced color spaces produced using SOM, MDS and PCA respectively. PCA produces a line or plane in 3D that minimizes the variance of the color distribution $\mathbf{b}$. If the colors do not follow a plane or line, which is mostly the case, many colors can not be addressed. While simple to compute and easy to store, PCA-based color manifolds consistently perform worse, as also shown in our perceptual study. MDS preserves intrinsic (geodesic) distances of the color distribution $\mathbf{b}$ and performs well if a 1D or 2D manifold exists. The bottom set of Figure 7.7 shows a typical failure case of MDS where the distribution forms a cycle and geodesic distances fail to produce an embedding into a 1D or 2D disk. Such cyclic paths are rare in color distributions but do exist. SOM naturally handles cyclic distributions which can be an issue for MDS. Figure 7.38 shows more detailed results for different classes using different dimensionality reduction approaches.

7.3.2 Algorithm Analysis

It is instructive to analyze the behavior of our algorithm with different parameter settings. We perform some analyses on the manifolds generated using MDS and SOM described in Section 7.2.

MDS Figure 7.8 shows the result of the MDS with different bin sizes. To further understand MDS, Figure 7.9 shows our its results on analytical data with constant and varying density distribution of color. Finally, the effect of the smoothness parameter s_m is shown in Figure 7.10.

SOM Figure 7.11 shows the SOM approach performed on analytical data with different distribution of colors. Next, we show SOM results with different initializations of the weight grid (Figure 7.12), different grid resolutions h (Figure 7.13), different histogram sizes (Figure 7.14) and different smoothness parameters r_0 (Figure 7.15). Finally, Figure 7.16 shows how SOM performs for different batch sizes s and training step counts t.

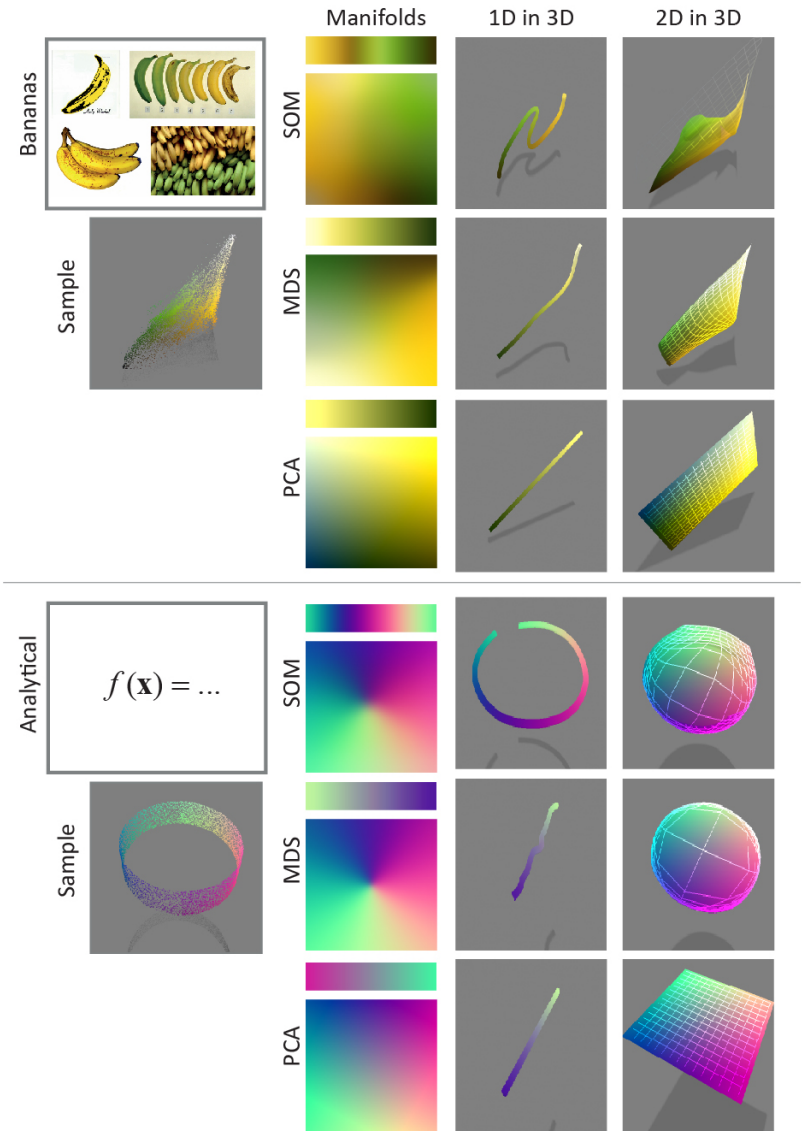

Figure 7.7: Different manifolds produced using different approaches of two different sets. In every set, the 1st row, 1st column *("Input")* shows the high-dimensional input used. The 2nd row, 1st column *("Samples")* shows the color distribution in 3D RGB space. The 2nd column shows the 1D and 2D manifolds *("Manifolds")* generated using SOM (1st row), MDS (2nd row) or PCA (3rd row). Finally, we show these 1D manifolds *("1D in 3D")* (3rd column) and 2D manifolds *("2D in 3D")* (4th column) as paths and patches in 3D RGB space. The top set shows the results of the "Bananas" class. The bottom set shows the results of an analytical color distribution. Here, SOM outperforms both PCA and MDS.

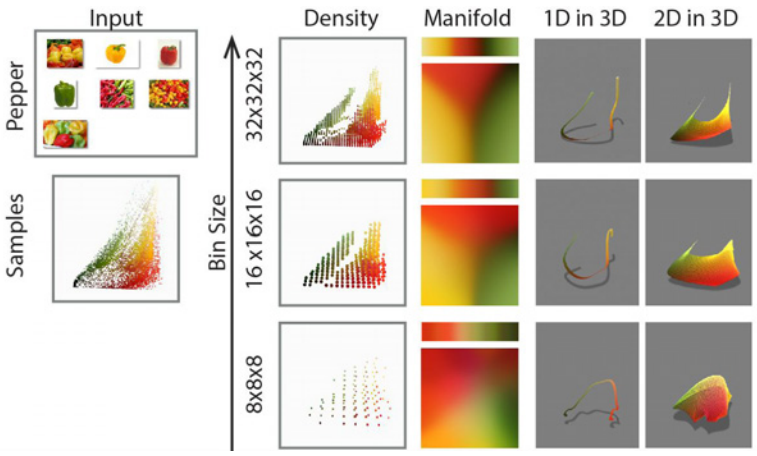

Figure 7.8: MDS approach: Comparison between using a bin size of $32 \times 32 \times 32$ (1st row), $16 \times 16 \times 16$ (2nd row) and $8 \times 8 \times 8$ (3rd row). Input images returned by Google Image Search *(1st col., 1st row, only subset shown)*, color distribution in 3D RGB space *(1st col., 2nd row)*, the discrete histogram *(2nd col.)* and 1D as well as 2D manifolds *(3rd col.)* produced by MDS with different bin size *(rows)* are shown. Finally, we show these 1D manifolds *(4th col.)* and 2D manifolds *(5th col.)* as paths and patches in 3D RGB space. We chose bin size $16 \times 16 \times 16$ *(2nd row)* as it gave results similar to $32 \times 32 \times 32$ while being faster to construct, on the other hand, $8 \times 8 \times 8$ gave worse results.

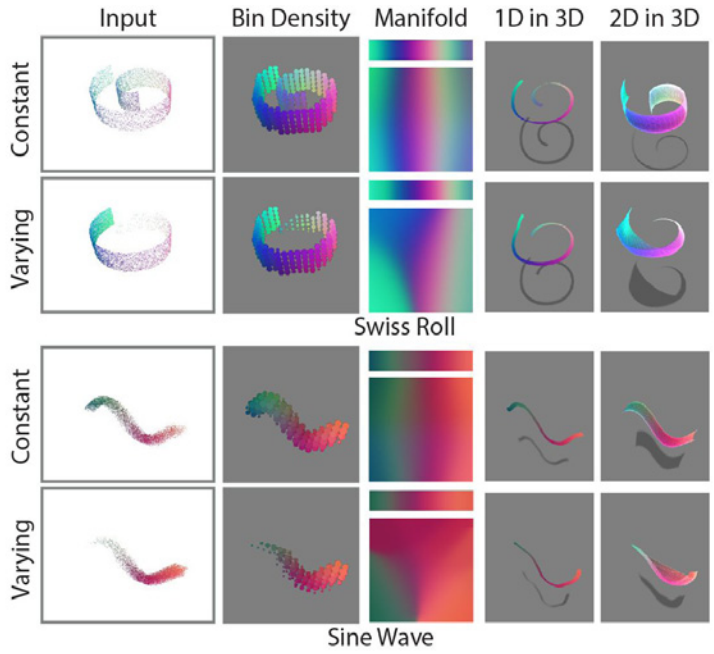

Figure 7.9: MDS approach: Input points are generated according to the Swiss roll and sine wave pattern with constant *(1st/3rd row)* and varying *(2nd/4th row)* density. Color distribution in 3D RGB space *(1st col.)*, the discrete histogram *(2nd col.)* and 1D as well as 2D manifolds *(3rd col.)* produced by MDS with different input density *(rows)* are shown. Finally, we show these 1D manifolds *(4th col.)* and 2D manifolds *(5th col.)* as paths and patches in 3D RGB space. In the bin density images *(2nd col.)*, larger spheres indicate a higher density. Note how our algorithm enlarges the space of color with higher density and shrinks the space of color with less density *(2nd/4th row)*.

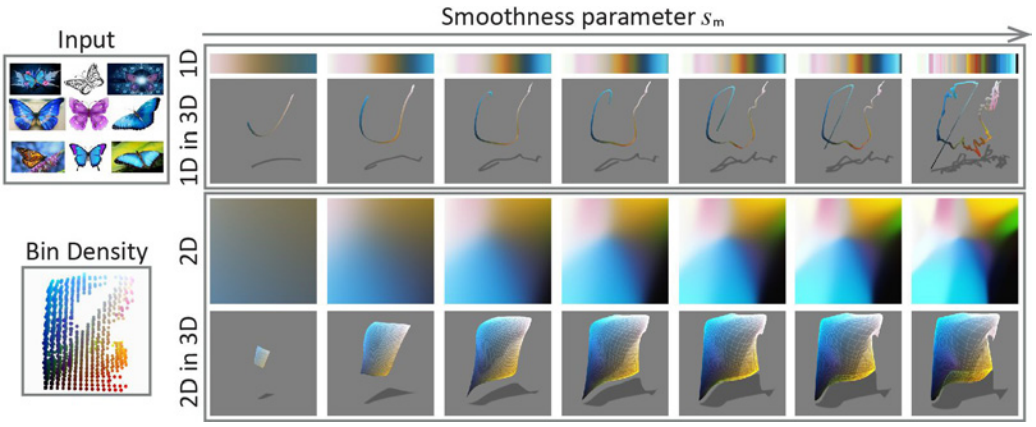

Figure 7.10: MDS approach: A "Butterfly" manifold, reconstructed using increasing smoothness s_{m}, which will preserve high-frequency signals from the input data but lose the smoothness property on the manifold. Input images returned by Google Image Search *(1st col., 1st row, only subset shown)*, discrete histogram *(1nd col., 2nd row)* and 1D *(from 2nd col., 1 row)* as well as 2D manifolds *(from 2nd col., 3rd row)* produced by MDS with different smoothness parameters *(from 2nd col. on)* are shown. Finally, we show these 1D manifolds *(from 2nd col., 2nd row)* and 2D manifolds *(from 2nd col., 4th row)* as paths and patches in 3D RGB space. We empirically set $s_{\mathrm{m}} = 7.5$ for the 1D manifold and 3.5 for the 2D manifold respectively as they gave a good balance between preserving input data color and smoothness on the manifolds.

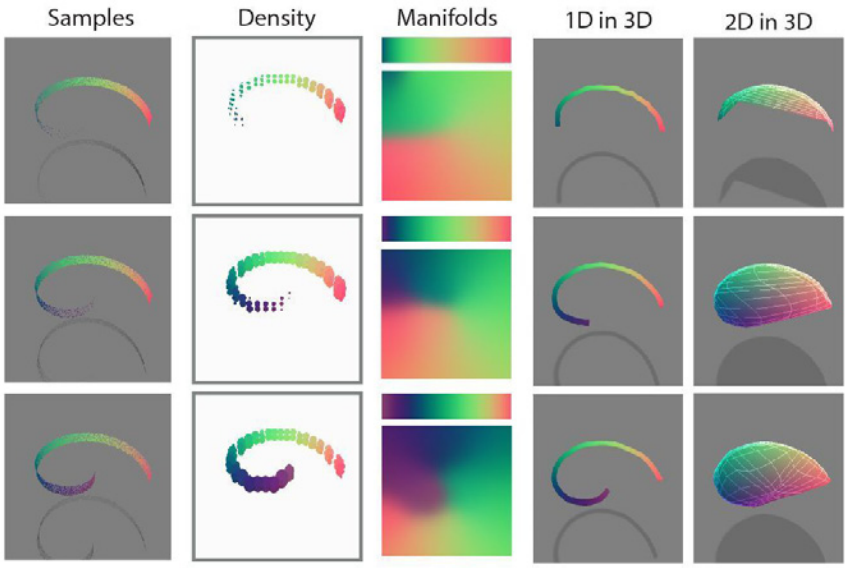

Figure 7.11: SOM approach: Input points are generated according to the Swiss roll pattern with different density. Color distribution in 3D RGB space *(1st col.)*, the discrete histogram *(2nd col.)* and 1D as well as 2D manifolds *(3rd col.)* produced by MSD for different input density *(rows)* are shown. Finally, we show these 1D manifolds *(4th col.)* and 2D manifolds *(5th col.)* as paths and patches in 3D RGB space. In the bin density images *(2nd col.)*, larger spheres indicate a higher density. Our algorithm enlarges space of color with higher density (more important) and shrinks space of color with less density (less important).

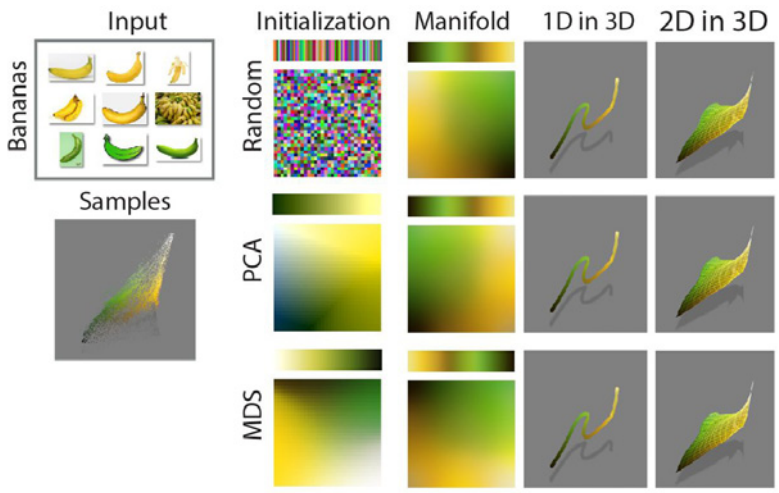

Figure 7.12: SOM approach: SOM performs quite consistently with different initializations. Input images returned by Google Image Search *(1st col., 1st row, only subset shown)*, color distribution in 3D RGB space *(1st col., 2nd row)*, the initialized 1D as well as 2D manifolds *(2nd col.)* and the final 1D as well as 2D manifolds *(3rd col.)* produced by SOM are shown. Finally, we show these 1D manifolds *(4th col.)* and 2D manifolds *(5th col.)* as paths and patches in 3D RGB space. The first row shows a random initialization, the second and third show initialization using the results from PCA and MDS approach respectively.

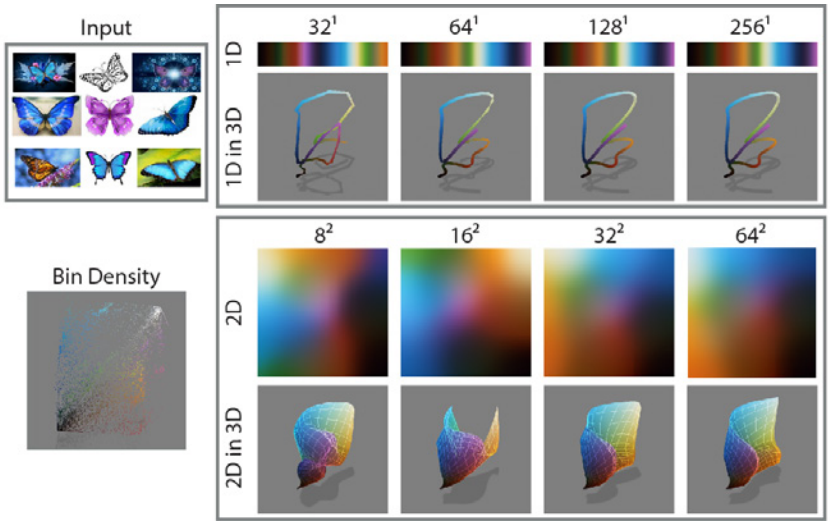

Figure 7.13: SOM approach: SOM for different grid resolutions. Input images returned by Google Image Search *(1st col., 1st row, only subset shown)*, color distribution in 3D *(1nd col., 2nd row)* and 1D *(from 2nd col., 1 row)* as well as 2D manifolds *(from 2nd col., 3rd row)* produced by SOM with different grid resolution *(from 2nd col. on)* are shown. Finally, we show these 1D manifolds *(from 2nd col., 2nd row)* and 2D manifolds *(from 2nd col., 4th row)* as paths and patches in 3D RGB space. In 1D, $h = 128$ gives results similar to $h = 256$ while being faster to construct. Similarly, in 2D, $h = 32$ gives results similar to $h = 64$ but is faster to construct.

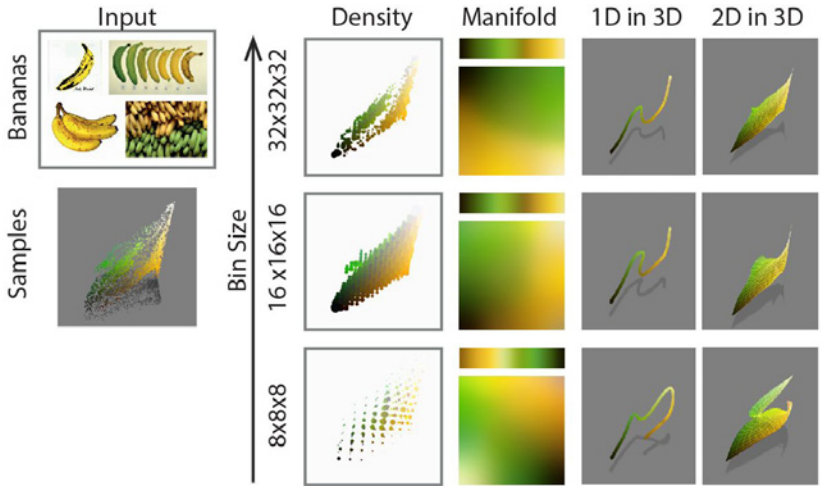

Figure 7.14: SOM approach: A "Banana" manifold for different bin sizes. Comparison between using a bin size of $32 \times 32 \times 32$ (1st row), $16 \times 16 \times 16$ (2nd row) and $8 \times 8 \times 8$ (3rd row). Input images returned by Google Image Search *(1st col., 1st row, only subset shown)*, color distribution in 3D RGB space *(1st col., 2nd row)*, the discrete histogram *(2nd col.)* and 1D as well as 2D manifolds *(3rd col.)* produced by SOM with different bin size *(rows)* are shown. Finally, we show these 1D manifolds *(4th col.)* and 2D manifolds *(5th col.)* as paths and patches in 3D RGB space. We chose bin size $16 \times 16 \times 16$ as it gave results similar to $32 \times 32 \times 32$ but was faster to construct, while $8 \times 8 \times 8$ gave worse results.

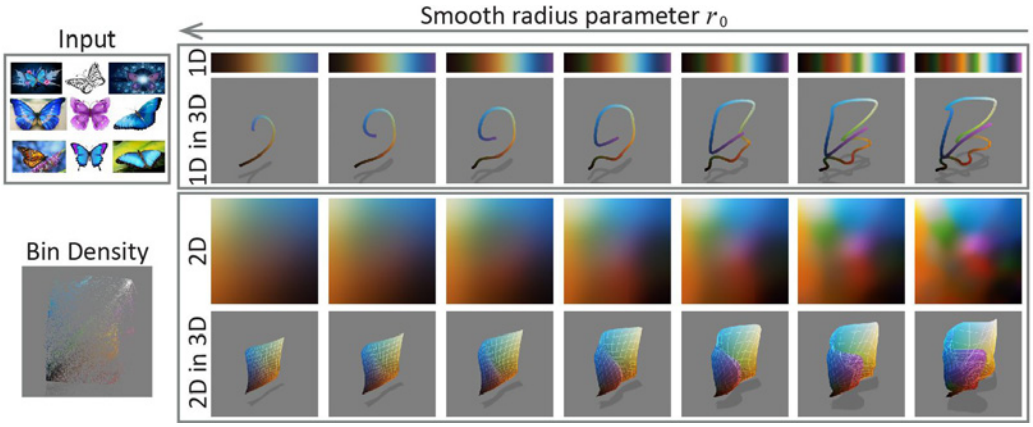

Figure 7.15: SOM approach: A "Butterfly" manifold, reconstructed with decreasing smooth radius parameter r_0. Smaller r_0 preserves high-frequency signals from the input data but loses the smoothness property on the manifold. Input images returned by Google Image Search *(1st col., 1st row, only subset shown)*, discrete histogram *(1nd col., 2nd row)* and 1D *(from 2nd col., 1 row)* as well as 2D manifolds *(from 2nd col., 3rd row)* produced by SOM with different smoothness parameters *(from 2nd col. on)* are shown. Finally, we show these 1D manifolds *(from 2nd col., 2nd row)* and 2D manifolds *(from 2nd col., 4th row)* as paths and patches in 3D RGB space. We empirically set $r_0 = 15$ for the 1D and $r_0 = 20$ for the 2D manifold respectively as they give a good balance between preserving input data color and smoothness on the manifolds.

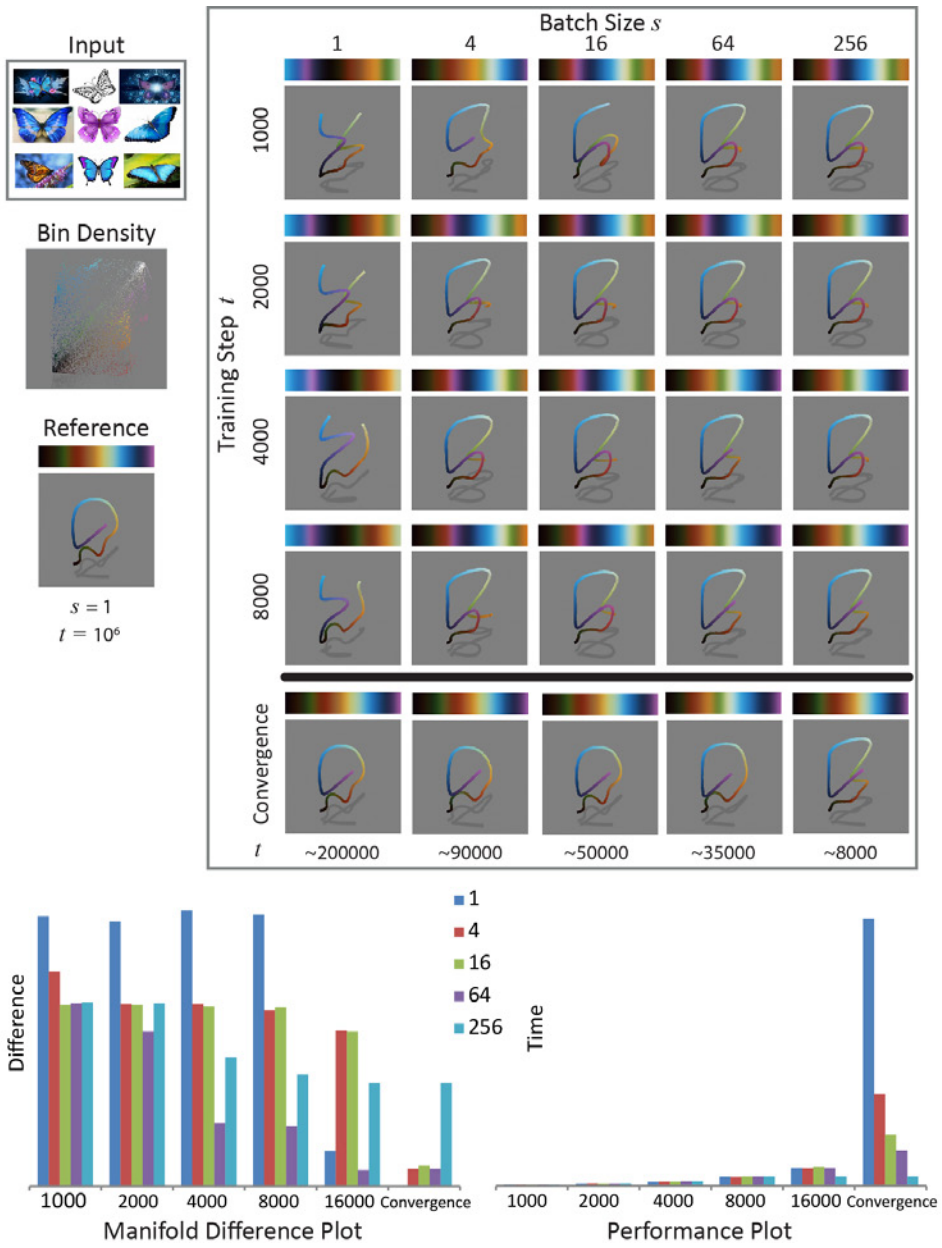

Figure 7.16: SOM approach: The top block shows results with different number of iterations and processed samples s in every training step; more details, input images returned by Google Image Search *(1st col., 1st row, only subset shown)*, color distribution in 3D RGB space *(1st col., 2nd row)*, the reference 1D manifold produced with batch size $s = 1$ after $t = 10^6$ iterations and its path in 3D RGB space *(1st col., 3rd row)* are shown. From the second column, the 1D manifolds generated with different training step t *(row)* and batch size *(col.)*, and their paths in 3D RGB space are shown. From the second column on, the final row shows the converged results at different training step t when using different batch size s. Note that with $s = 64$ and $t = 4000$, the result is similar to the convergence as $s = 1, t = 10^6$. In the bottom block, the left part shows the L2-differences of the reference *($s = 1, t = 10^6$)* and the manifolds generated using different training step t *(col.)* and batch size s *(colored bars)*, a lower bar indicates that the generated manifold is more similar to the reference manifold. The performance plot on the right part shows their correspondence performances, a higher bar indicates a slower performance.

7.3.3 User Study

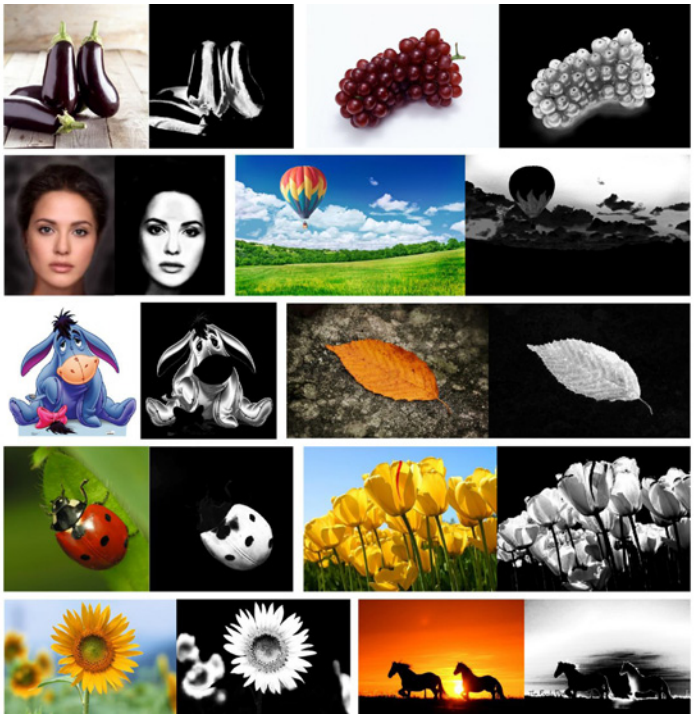

Figure 7.17: Images used in our color adjustment study and their mask are defined manually. We chose 10 different images from 10 classes: “brinjal”, “eeyore”, “grape”, “human skin”, “ladybug”, “leaf”, “sky”, “sunflower” , “sunset” and “tulips”.

Figure 7.18: In the creation phase , images were edited by the same subject using 12 different types of color pickers. In the ranking phase , other subjects were asked to sort the images in increasing order of color appearance quality using Microsoft Powerpoint. The top row shows the random layout that was presented to the subject. The second row shows the same set after sorting by a random subject.

We evaluated the usefulness of the proposed color manifolds in a color adjustment and a color exploration task.

Our subjects aged between 23 and 35 years old. All of them were graduate students in engineering or art. None of them reported to have issues with color perception. First, the subjects were introduced to the color pickers and interacting with picking the color. Then

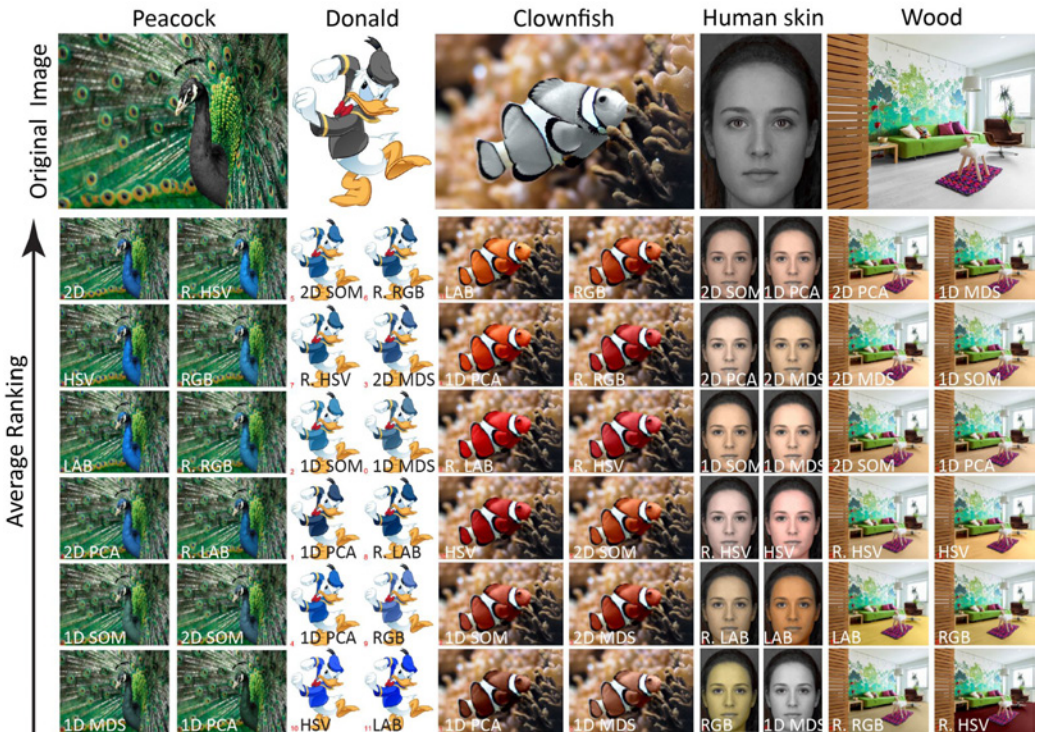

Figure 7.19: Details of our exploration study. The top row shows the original images used in the study. We chose 5 different classes: “peacock”, “human skin”, “clownfish”, “wood” and “Donald”. From the second row on, some of the edited images are shown in decreasing order of average ranking from top to bottom.

they were given a training session to learn how to perform color adjustment using several test cases until they felt comfortable to begin the study. We gave the subjects a short break between the studies.

The studies were conducted on a desktop setup using a 24" LCD graphics display for visual stimuli. The subjects were seated at approximately 60 cm distance to the screen and we asked them to place the mouse on the table such that they felt comfortable.

In the color adjustment task, 13 subjects were asked to adjust the color of an image to match a reference image using different interfaces. Images used for manifold construction were excluded from the user study. Figure 7.17 shows the set of images used for our study. In each trial, the participants were presented a color image and a second version of the same image that had a random color assigned to a soft region defined by the authors. Subjects were asked to adjust the color of the image back to the reference using a specific color picker UI. Adjustment was performed by picking a color inside the color picker and the image was modified by means of recoloring [Gastal and Oliveira 2011]. This objective is a typical color selection task in everyday photo editing. The UI was randomly either a common RGB, Lab, HSV (Figure 2.5 (a)-(j)), a “reduced” RGB, “reduced” Lab, “reduced” HSV where colors outside the color distribution of the image class are grayed out as un-selectable as well as our proposed 1D or 2D manifold interfaces, produced using either PCA, MDS or SOM. The “reduced” color spaces acted as a control group to detect whether an increase of performance

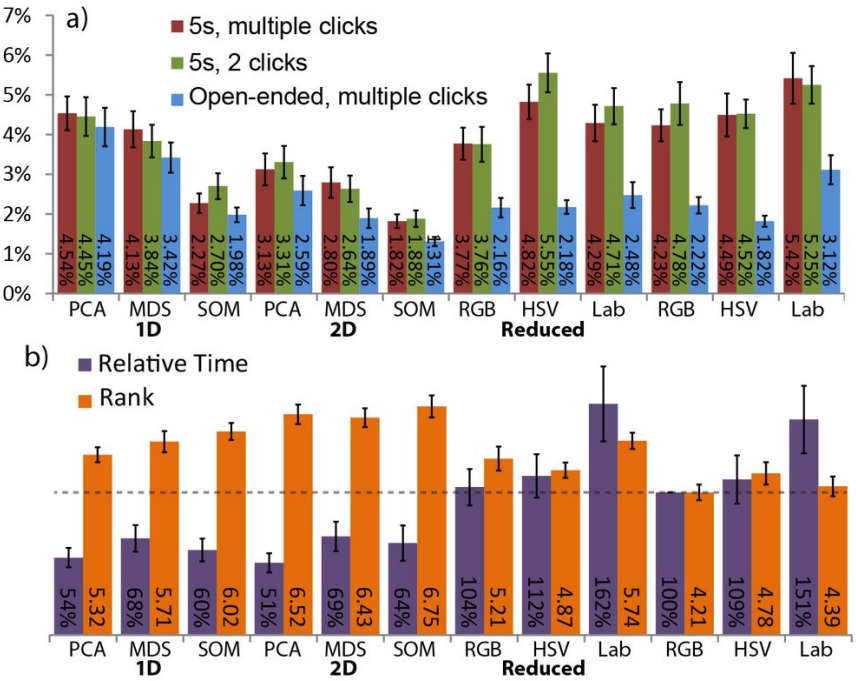

Figure 7.20: User study results: *(a)* Adjustment task: Normalized CIEDE2000 error of several adjustment UIs and their standard error of mean. Lower numbers are better (less error). *(b)* Exploration task: Mean time, relative to common RGB *(dotted line)* and mean rank accompanied by their standard error of mean. Lower time is better (faster). A higher rank indicates a better rating from the subjects. Different experimental conditions for the same UI are encoded as different colors.

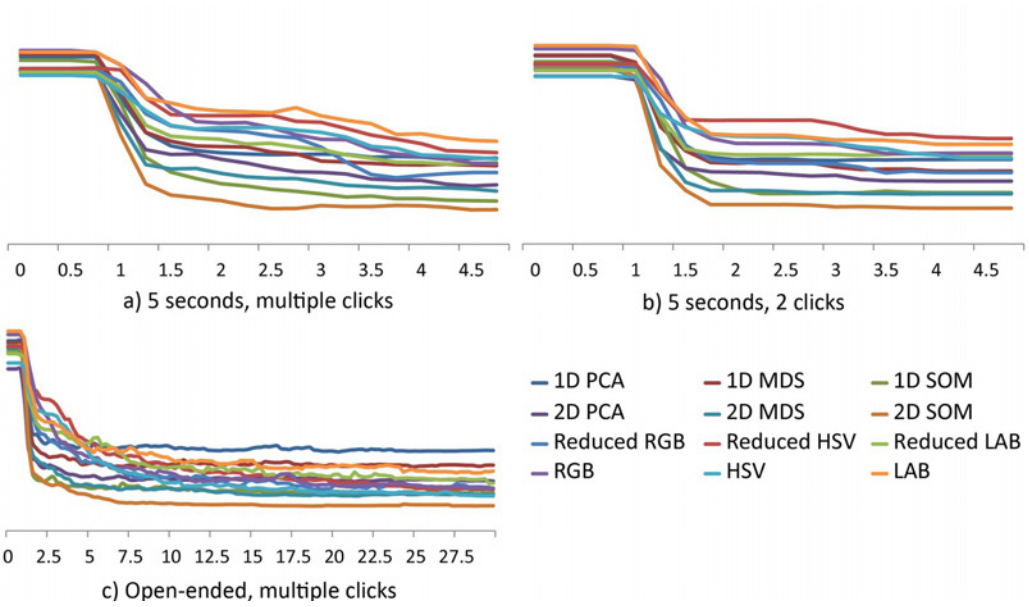

Figure 7.21: CIEDE2000 color difference of different color pickers over the editing time in different study settings: a) 5 seconds, multiple clicks b) 5 seconds, two clicks c) open-ended, multiple clicks.

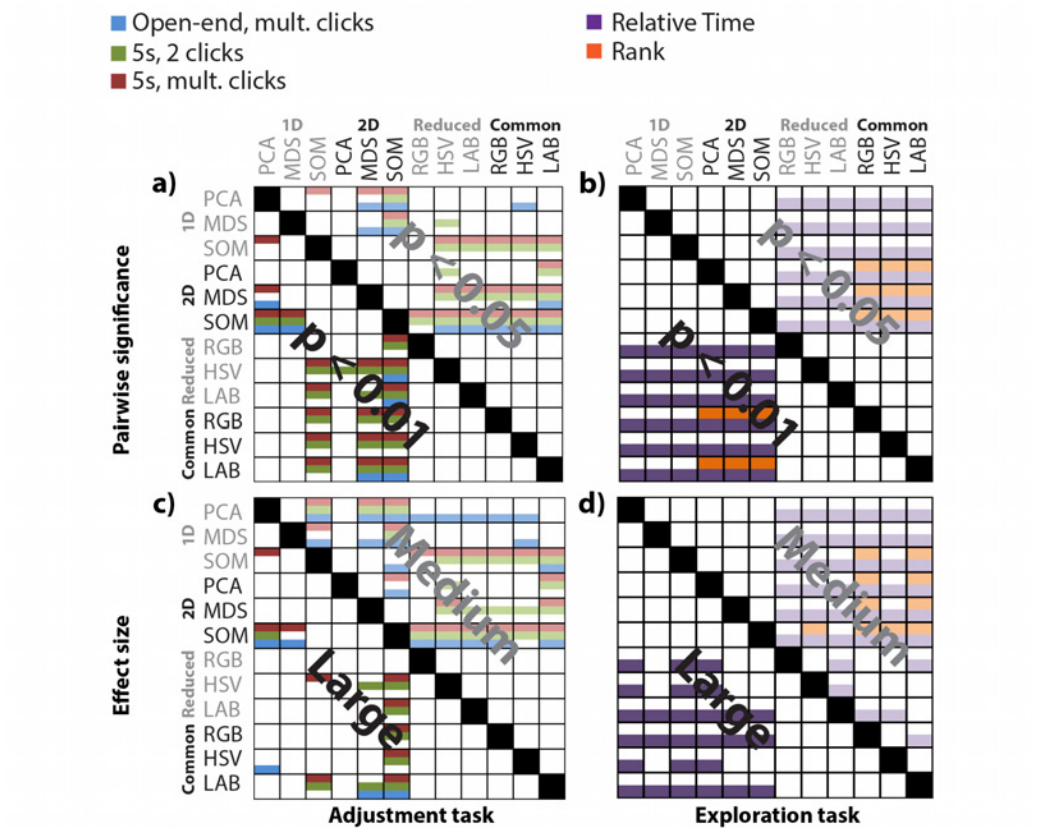

Figure 7.22: Pairwise statistical significance and effect size of different interfaces. The top-right of *a* and *b* show pairwise statistical significance with $p < .05$, while the bottom-left shows $p < .01$. The top-right of *c* and *d* show a medium while the bottom left shows a large effect. A colored cell i, j indicates a significance or an effect between the i-th and the j-th color selection interface. Different experimental conditions are encoded as colors.

of color manifolds was only due to excluding irrelevant colors (e. g., "skin" does not include blue). The task had to be finished either within a 5 second time budget with multiple mouse clicks, by using two clicks or by multiple clicks in an open-ended setting. We measured user performance as the average CIEDE2000 color difference of all pixels and all subjects denoted in percentage of the maximal error. Figure 7.20a shows the performance of different interfaces. Figure 7.21 shows the mean CIEDE2000 color difference over all pixels and all subjects, denoted in percentage of the maximal error over editing time in 3 different cases: 5 second - multiple clicks, 5 seconds - 2 clicks and open-ended timing - multiple clicks per trial. As the mean difference distribution was non-Gaussian (D'Agostino-Pearson), we used the Kruskal-Wallis non-parametric test instead of ANOVA [Siegel and Castellan 1988]. Dunn's multiple comparisons test was used to calculate the pairwise statistical significance between different interfaces. Figure 7.22a shows that both our 1D/2D SOM manifolds outperform classical color pickers and reduced color pickers with statistical significance $p < 0.05$. Figure 7.22c shows pairwise Cohen's effect size of the adjustment task. Table 7.2 shows the average CIEDE2000 color difference of the color adjustment tasks.

The color exploration task was performed in two phases where some participants created im-

Setup	1D PCA	1D MDS	1D SOM	2D PCA	2D MDS	2D SOM	Re. RGB	Re. HSV	Re. LAB	RGB	HSV	LAB
5s - (M)	.045 .004	.041 .005	.023 .002	.031 .004	.028 .004	.018 .002	.038 .004	.048 .004	.043 .005	.042 .004	.045 .005	.054 .006
5s - (2)	.045 .005	.038 .004	.027 .003	.033 .004	.026 .003	.019 .002	.038 .004	.056 .005	.047 .005	.048 .005	.045 .004	.053 .005
Open - (M)	.042 .005	.034 .004	.020 .002	.026 .004	.019 .002	.013 .001	.022 .002	.022 .002	.025 .003	.022 .002	.018 .001	.031 .004
Time	0.54 0.06	0.68 0.09	0.59 0.08	0.51 0.07	0.69 0.10	0.64 0.12	1.04 0.13	1.12 0.15	1.62 0.26	1.00 0.00	1.09 0.17	1.51 0.24
Rank	5.32 0.30	5.71 0.42	6.02 0.34	6.52 0.39	6.43 0.38	6.75 0.45	5.21 0.48	4.87 0.31	5.74 0.33	4.21 0.32	4.78 0.45	4.39 0.40

Table 7.2: Table of statistics: From top to bottom are the results of the color adjustment tasks: 5 seconds - multiple clicks (5s-(M)), 5 seconds - two clicks (5s-(2)) and open-ended - multiple clicks (Open-(M)). For the color adjustment tasks, we report the normalized-CIEDE2000 color difference of different interfaces. Next are the results of the color exploration tasks: the relative finishing time of the creation phase and the ranking of images generated using different interfaces in the ranking phase. For every study, the first row shows the mean value and the second row shows the standard error of the mean.

ages using different interfaces and other subjects ranked their images later on. In the *creation* phase, subjects were asked to adjust the color of an image using different interfaces until satisfied. Figure 7.19 shows the set images used for our study. Four expert users (computer graphics hobbyists naïve to the purpose of the study) participated in this step performing 60 trials each. In every trial, participants were presented one out of five different images where some parts needed to be colored using one out of 12 different color pickers from the first experiment. Figure 7.20b shows the average finishing time of different color pickers relative to using an RGB color picker for a given set and the average ranking for different color pickers. The resulting images were used in a second *ranking* phase where we randomly chose 10 different sets of images. Every set contained 12 images edited by one specific expert user on the same input image using 12 different interfaces. 10 other subjects (3 males and 7 females, who did not participate in the creation phase) participated in the study. Every subject was sequentially presented 10 Powerpoint slides where every slide contained a different set of images showed in random layout. For every slide, the subject was asked to sort the images from left to right in increasing order of color appearance (Figure 7.18). An example set is showed in Figure 7.18. Figure 7.19 shows the average ranking of some images used in the study. Similar to the adjustment study, we used a Kruskal-Wallis test and Dunn's post-hoc. Figure 7.22b shows that our 1D and 2D SOM allows for significantly better exploration performance. Figure 7.22d shows pairwise Cohen's effect size of the exploration task. Table 7.2 shows the average relative timing of the creation phase and the average ranking of the ranking phase.

Conclusion Our studies re-confirm that the choice of classical color pickers has only little impact on the performance of a color selection interface. Our studies show that encoding "class" information into classical color pickers, such as done in our "reduced" pickers, does not improve the performance in both color adjustment and color exploration task.

Our data-driven 1D or 2D manifolds outperform classical color pickers and reduced color pickers in both color adjustment and exploration task. Furthermore, SOM is the best choice compared to PCA and MDS. Unless stated otherwise, weakly supervised SOM was used for further results shown in this chapter.

7.4 Results

7.4.1 Manifolds

This section shows 1D and 2D manifolds produced automatically for a certain class, for classes with additional semantic parameters or by classes interactively composed by a user.

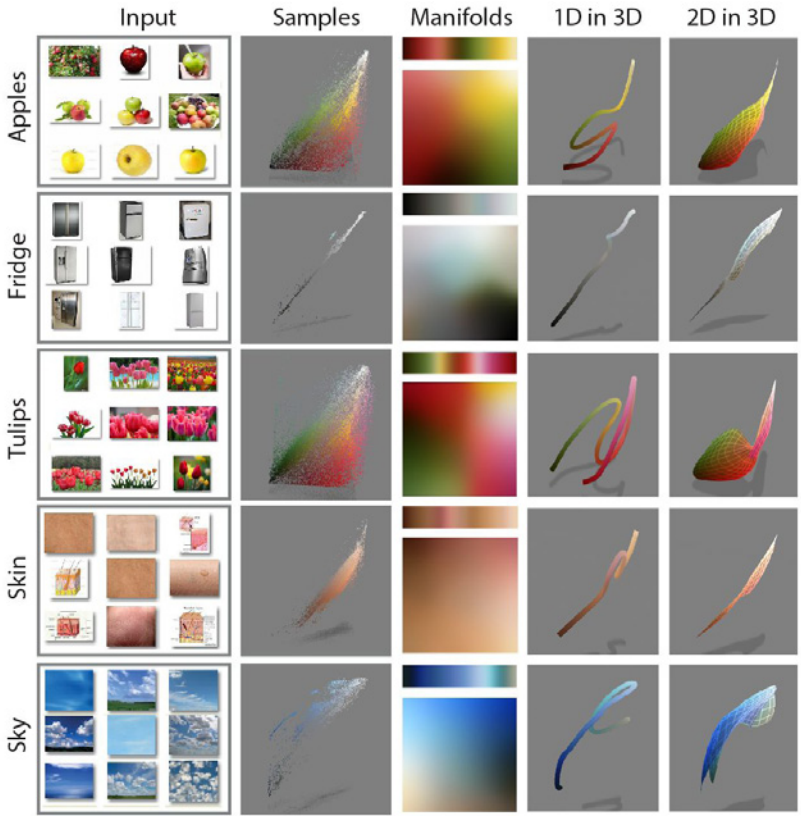

Figure 7.23: Input images returned by Google Image Search *(1st col., only subset shown)*, color distribution in 3D RGB space *(2nd col.)* and 1D as well as 2D manifolds *(3rd col.)* produced by our weakly supervised SOM for different classes *(rows)*. Finally, we show these 1D manifolds *(4th col.)* and 2D manifolds *(5th col.)* as paths and patches in 3D RGB space.

1D and 2D Manifolds Manifolds generated for different classes are shown in Figure 7.23. As images contain shading, true reflectance data as extracted from Internet 3D model repositories can be superior (Figure 7.24). If the class is not unique inside the image, we allow user annotation (Figure 7.25) or make use of already annotated data such as the SUN database

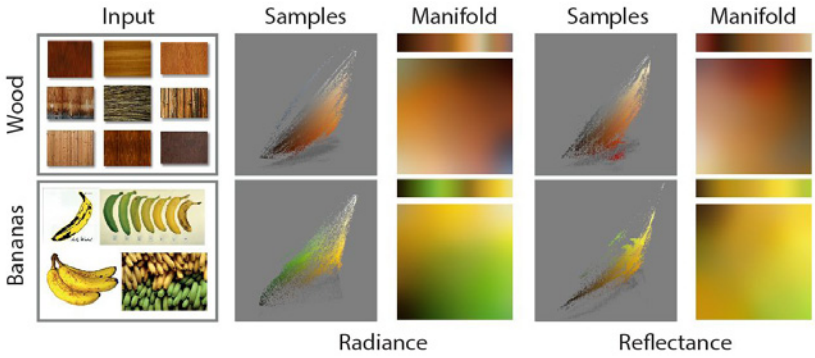

Figure 7.24: Comparison of manifolds constructed from 2D images with shading returned by Google Image Search *(3rd col.)* and reflectance from textures of 3D meshes acquired from Google Sketchup *(5th col.)*. The color distribution of the 2D images in 3D RGB space and the reflectance from the textures are shown in the second and forth column, respectively.

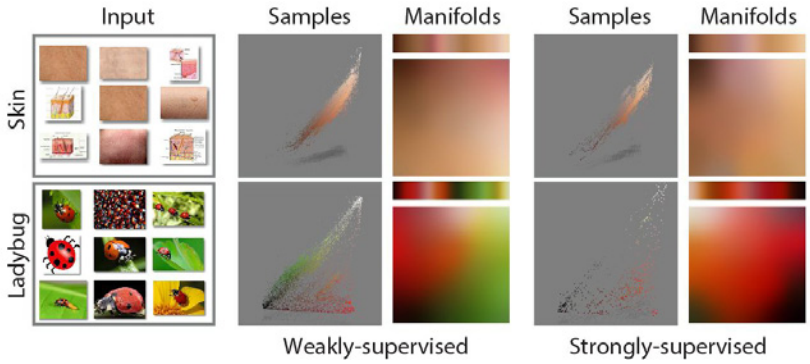

Figure 7.25: Comparison of weakly and strongly supervised manifolds. Weakly supervised manifolds *(3rd col.)* are constructed using input images returned by Google Image Search *(1st col., only subset shown)*. Next, image pixels that do not belong to the class are manually excluded and the refined images are then used to construct the strongly supervised manifolds *(5th col.)*.

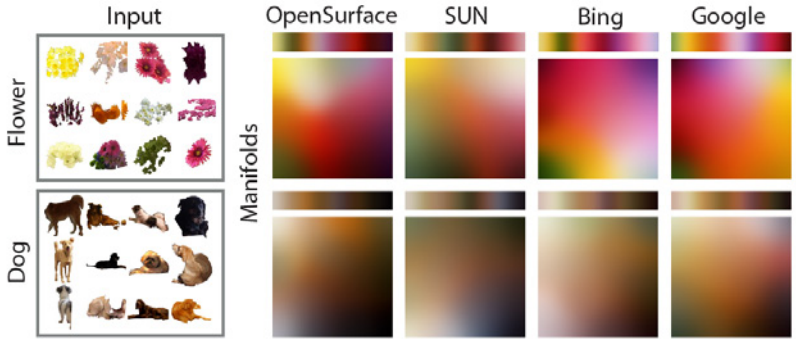

Figure 7.26: Comparison of manifolds generated using different databases. The first column shows a subset of the class from the OpenSurface database [Bell et. al. 2013]. Next are the manifolds generated using the input images from the annotated database OpenSurface *(2nd col.)*, Sun [Xiao et al. 2010] *(3rd col.)*, or weakly supervised images returned by Bing *(4th col.)* and Google *(5th col.)* Image Search using the same keywords.

[Xiao et al. 2010] or the OpenSurface database [Bell et. al. 2013]. Figure 7.26 shows different manifolds generated using different databases. Figure 7.27 shows classes of our 1D and 2D manifolds compared to the classical color pickers. Manifolds are low-resolution images of only a few kilobyte that are ready to be used in any application just by using any smooth reconstruction filter. Figure 7.38 shows manifolds generated by weakly supervised images with different dimensionality reduction algorithms.

Parametrized Families of Manifolds The range of colors found for a certain class sometimes depends on external conditions, such as the colors of a forest follow the seasons or the sky's color changes over the course of a day. Our approach naturally supports this observation by extending to families of manifolds $g_{\mathbf{t}}(\mathbf{x})$, *parametrized* by a k-dimensional vector $\mathbf{t} \in \mathbb{R}^k$. To this end, we acquire pairs $(\mathbf{a}_i, \mathbf{t}_i)$ of image colors and parameter values, such as when or where the image was taken. Then, we construct our manifold for every value of the parametric domain,e. g., one 1D MDS manifold for every time of the day as in Figure 7.28.

Interactive Manifold Compositing The efficient implementation of the manifold creation allows to interactively specify a set of images and observe the resulting manifold (Figure 7.30). Interactive manifold creation could be used to summarize or further refine the color organization of a set of images produced on-demand, e. g., by an image search.

Figure 7.27: Several manifolds generated using weakly supervised data from the Internet images. The final two sets on the bottom row (denoted as (S)) show the manifolds of two categories from the SUN database.

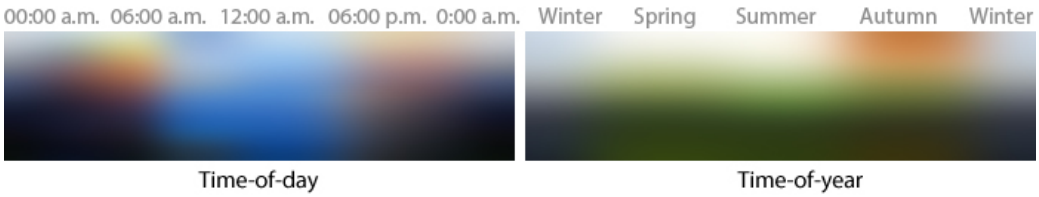

Figure 7.28: Two 1D parametric families of 1D manifolds. Each vertical slice is a manifold in color space that changes along the horizontal dimension depending on a (semantic) parameter. Figure 7.29 shows a subset of the images used to construct the manifolds.

Figure 7.29: The images (subset shown) used for the parameterized manifolds shown in Figure 7.28. The top row shows the datasets of forest in different seasons. The second row shows the datasets of sky over different time in a day.

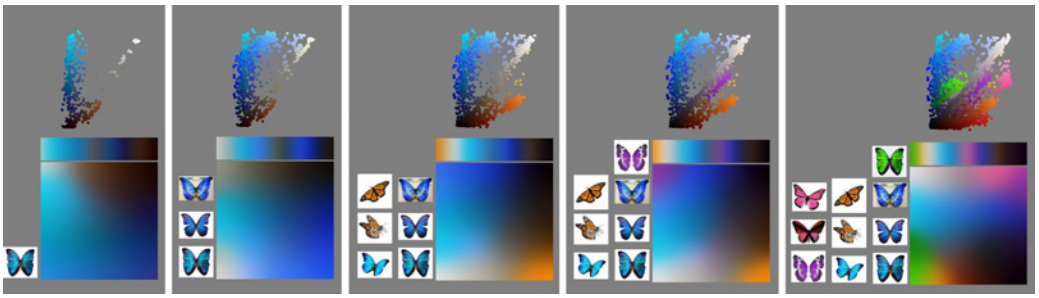

Figure 7.30: Interactive manifold creation in five steps. Starting from a single input image *(1st col.)*, a user interactively adds more images *(2nd to 5th col.)* to refine the "Butterfly" manifold. In every screenshot, the left column shows the set of input images, the top right shows the color point cloud in 3D and the bottom right shows the 1D and 2D manifolds.

7.4.2 Applications

Our manifolds allow for a range of applications, such as color editing, palettes, stylization, compression and white balancing.

User Interaction Our primary application of color manifolds is continuous 1D or 2D color slider (Figure 7.1). Instead of continuous variation of color, a discrete sampling leads to meaningful recoloring suggestions (Figure 7.32) or when directly used as discrete palettes (Figure 7.31). Compared to simpler alternatives like clustering, the discrete elements produced by our approach have a meaningful one- or two-dimensional ordering and can be refined if required.

Color Stylization Our approach can be used to permute the hue value in an image, while staying inside the manifold of plausible images (Figure 7.33). To this end, the image is

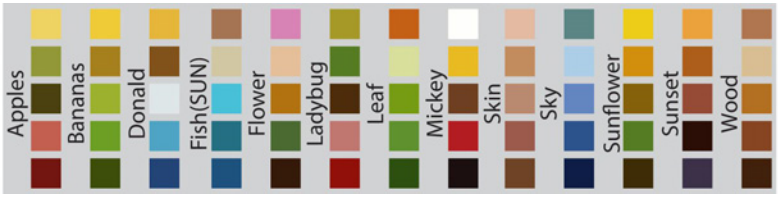

Figure 7.31: Palettes produced by equally-spaced sampling of a 1D manifold. The pallettes cover the space well and in a meaningful order.

Figure 7.32: Re-coloring suggestion galleries using the "Apples" *(1st row)*, "Skin" *(2nd row)* and "Sunset" *(3rd row)* manifold.

Figure 7.33: An original image *(1st col.)* and its hue permutations *(2nd to 4th col.)* using the "Apples" manifold.

segmented manually, a new random hue from the manifold is assigned to every segment.

Compression Our approach can be used to compress color information. First, the image is converted to CIE Lab. The luminance L is not compressed and the chrominance is processed further. Second, a color manifold is chosen, and all colors are transformed into the new space and compressed. Quantization of colors to a low number of bits is used to

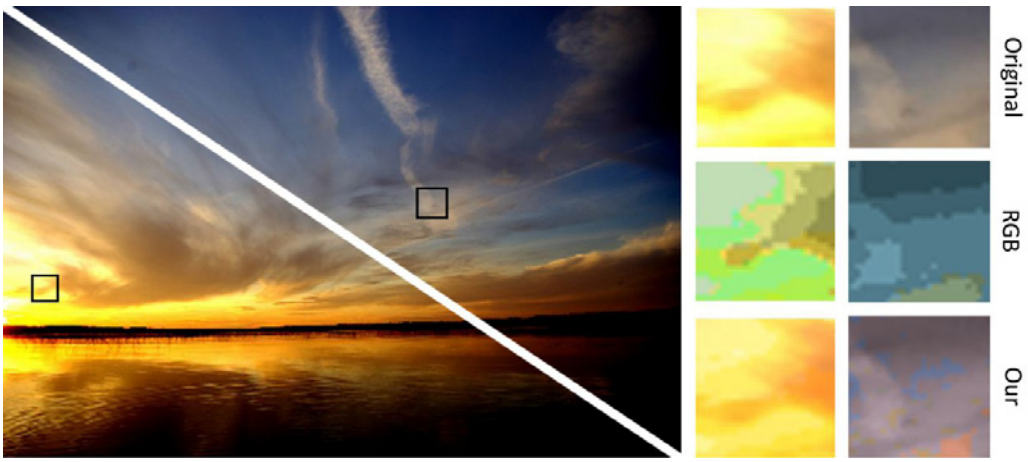

Figure 7.34: Original *(bottom left)* and compression using our "Sunset" manifold *(top right)* and using the RGB space *(insets)*.

compress colors in our example, but other operations such as DCT or wavelets would be possible. Additionally, we can quantize less important coordinates with 0 bits, skipping this dimension, i. e., reducing a 2D color to a 1D color. For decompression they are transformed back and combined with luminance (Figure 7.34). Our technique is especially useful to compress a set of images from the same class.

White Balance White balance seeks to disambiguate the product $\mathbf{a} = L\mathbf{r}$ of a spatially-invariant, unknown scalar RGB illuminant L and an unknown spatially-varying diffuse RGB reflectance $\mathbf{r}$. Humans are known to exploit knowledge of familiar i. e., plausible, reflectance to disambiguate this product [Olkkonen, Hansen and Gegenfurtner 2008]. Our manifolds encode such knowledge and can extend the popular "grey world assumption" to an "on-manifold assumption". This assumption defines the illuminant as the ratio of the average color in a k pixel-image and grey $L_{\text{grey}} = \sum \mathbf{a}_i / k$. In this framework, observing a slightly pink image would result in a pink illuminant. If we, however, know that parts of the image belong to the "human skin" class, pink might not be the illuminant, but the reflectance and the illuminant is white. Along those lines, our manifolds can be used to regularize plausible values for $\mathbf{r}$ as follows. First, the set of image pixels $\mathcal{M}$ that belong to a known class are selected. Next, we solve for the best RGB illuminant L that minimizes the cost function $\sum_{i \in \mathcal{M}} \left\| \alpha_f(\frac{\mathbf{a}_i}{L}) - \frac{\mathbf{a}_i}{L} \right\|^2$ where $\alpha_f(\mathbf{x})$ is the nearest color to $\mathbf{x} \in \mathbb{R}^m$ on the manifold f. The cost function optimizes for the manifold constraint that pixels in $\mathcal{M}$ after white balancing should stay inside the manifold (Figure 7.35). While the cost is non-linear due to the projection operation α_f, the space is small enough (all possible colors) to be enumerate exhaustively using a GPU-solver employing a 3D look-up table for α_f.

The approach is limited in that it cannot disambiguate situations, where the product of reflectance and illuminant, as well as the reflectance itself are close to the manifold: a slight blue tint in a human face which is likely not reflectance, but illuminant; a brown shift might be the reflectance (tanned skin) or it might be a brownish illuminant. The latter cannot be disambiguated.

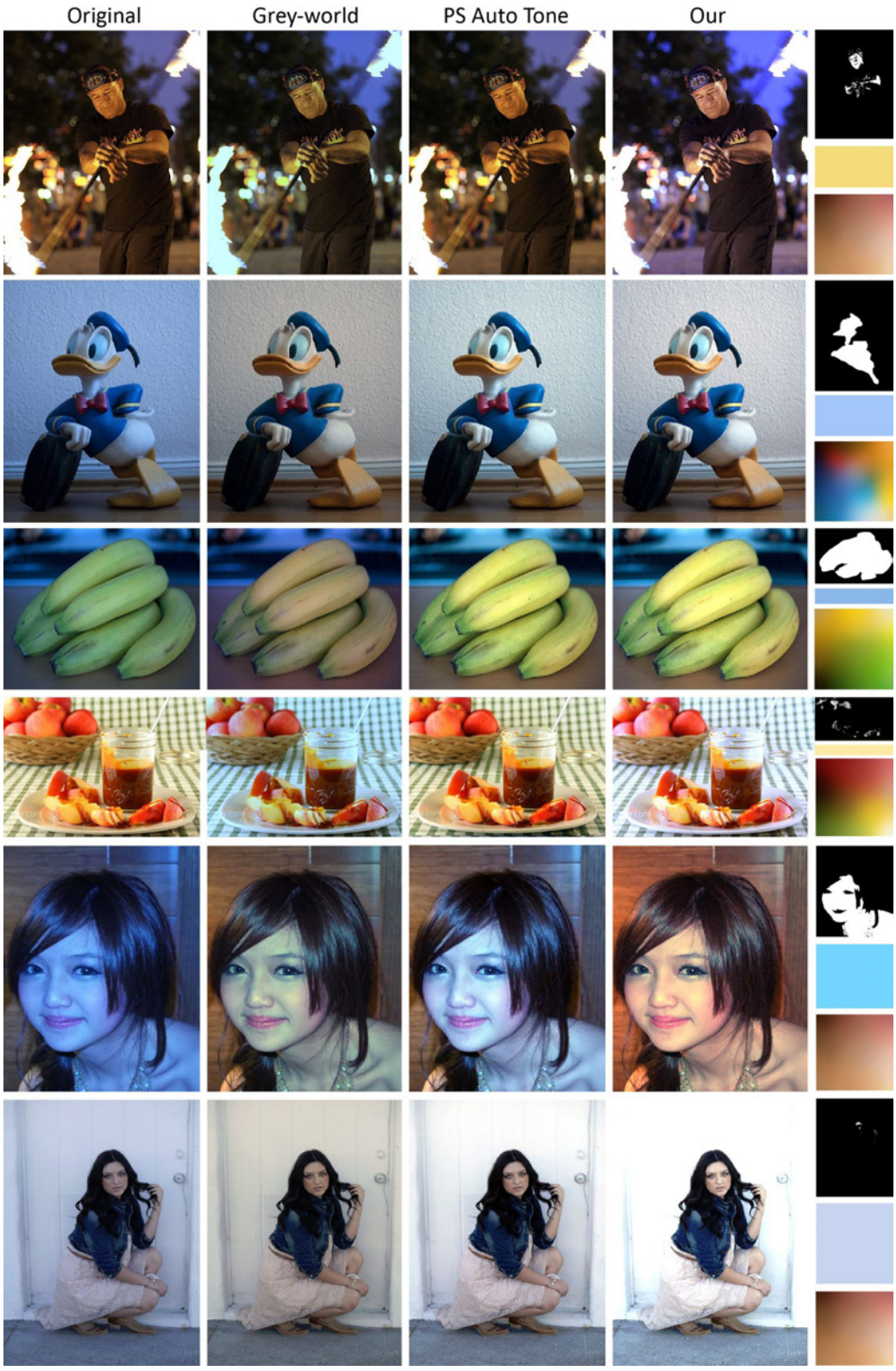

Figure 7.35: White balancing *(left to right):* Original, grey-world assumption, Adobe Photoshop CS6 Auto Tone, our approach, finally the mask and our illuminant. The manifolds used are: “Skin”, “Donald”, “Bananas” and “Apples".

7.5 Discussion and Limitations

Discussion Our pipeline is specifically designed for high performance and each step could be replaced to improve the quality of the constructed manifolds. During density estimation (Section 7.2.2), colors are quantized into l bins instead of using a continuous estimation over k colors such as RBF or moving least squares. This allows for higher performance and provides a unified input for different dimensionality reduction methods (Section 7.2.3), furthermore discrete histograms can be adapted easily to a simple outlier removal step by thresholding the bins. While robust statistics [Maronna, Martin and Yohai 2006] in a continuous estimation will likely improve the quality of outlier removal, it would further complicate our pipeline.

Limitations The main limitation of our work is the requirement of reliable classification, both to construct the manifold and to use it. For construction, we found the quality of typical search engine results to be sufficient, as indicated by our results. Typical failure modes are ambiguous queries such as "apple", which might refer to a fruit as well as to a brand of computers. For weakly supervised manifolds, we do not perform any preprocessing on the input image set. As images returned from Google Image Search might contain improper camera calibration, color distortions or the inter-reflection between objects, they will likely produce less sharp manifolds. Our simple background removal (Section 7.2.1) might fail to remove background colors from several images in the set due to complex backgrounds or objects with holes, etc. These parts normally occupy a small part of the images which could be rectified by density thresholding (Section 7.2.2). In some cases, colors that do not belong to the class such as green in the "ladybug" (Figure 7.25) are dominant. Even though these outliers (colors) might greatly reduce the usability of our manifold, they are closely related to the class semantically (leaves in the "ladybug") and we believe that they can act as supplementary colors for a specific class. While manifold construction faces all those challenges, once the manifolds for a class are made available, they don't need to be acquired again. For manifold usage, the ideal classifier would also detect the object class for a location inside the image or identify different classes if multiple classes are present in one image [Jia 2013]. Such classifiers are an active area of research in computer vision but not yet readily available to computer graphics applications. If the entire image belongs to a single class or the class is known a priori our manifolds are ready to use. For some classes, the dimensionality reduction is not always possible in a satisfying manner. We are able to find potentially non-linear lower-dimensional structures, but only if they are present. Classes that have a color distribution that roughly follows a dominant line are well-described by 1D manifolds, classes that are roughly distributed around a surface can be represented using 2D manifolds. However it is futile and counterproductive, to organize an inherently 2D distributions of colors into a 1D curve. Figure 7.36 contains more results of man-made objects or objects with rich textures. Our method might fail to produce the manifolds for these objects as they could have multiple dominant colors over the surface and thus form multiple manifolds in color space. Figure 7.37 shows additional results of man-made and textured objects generated using weakly-supervised images downloaded from Google Image Search. Finally, perceptual organization of colors into lightness, saturation, or the periodicity of hue would require additional considerations.

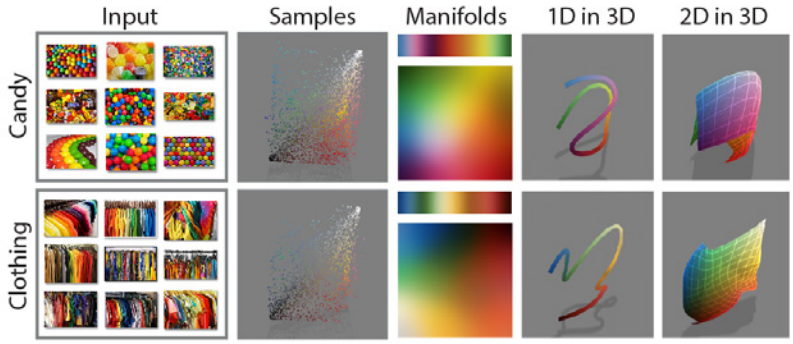

Figure 7.36: Limitations: Colors of some man-made and textured objects that virtually exists in all hues and shades and consequently should be used in 3D. The dimensionality reduction step would try hard to find the best traversal in 3D, resulting in a desperate zig-zag enumeration of the entire higher-dimensional space. Detailed explanations of this figure can be found in the caption of Figure 7.23.

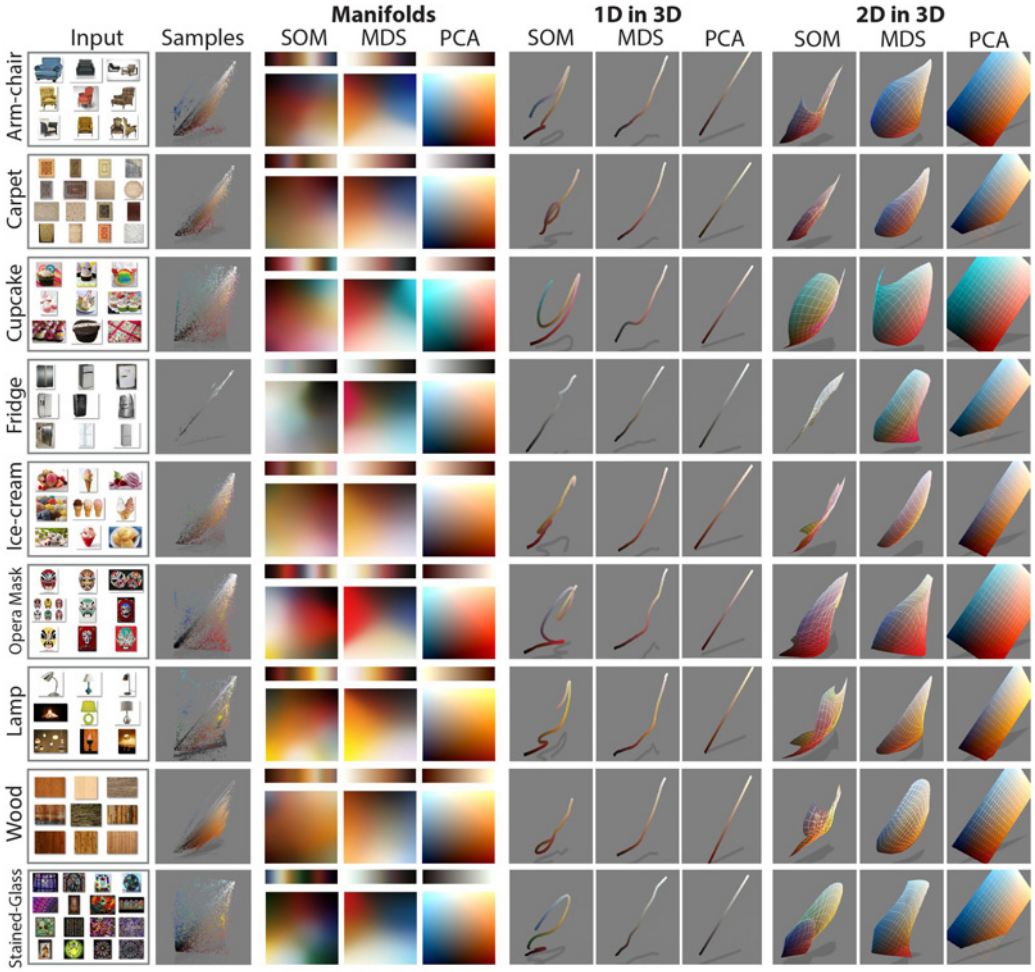

Figure 7.37: Manifolds of man-made or textured objects constructed using different dimensionality reduction approaches. Detailed explanations of this figure can be found in the caption of Figure 7.7.

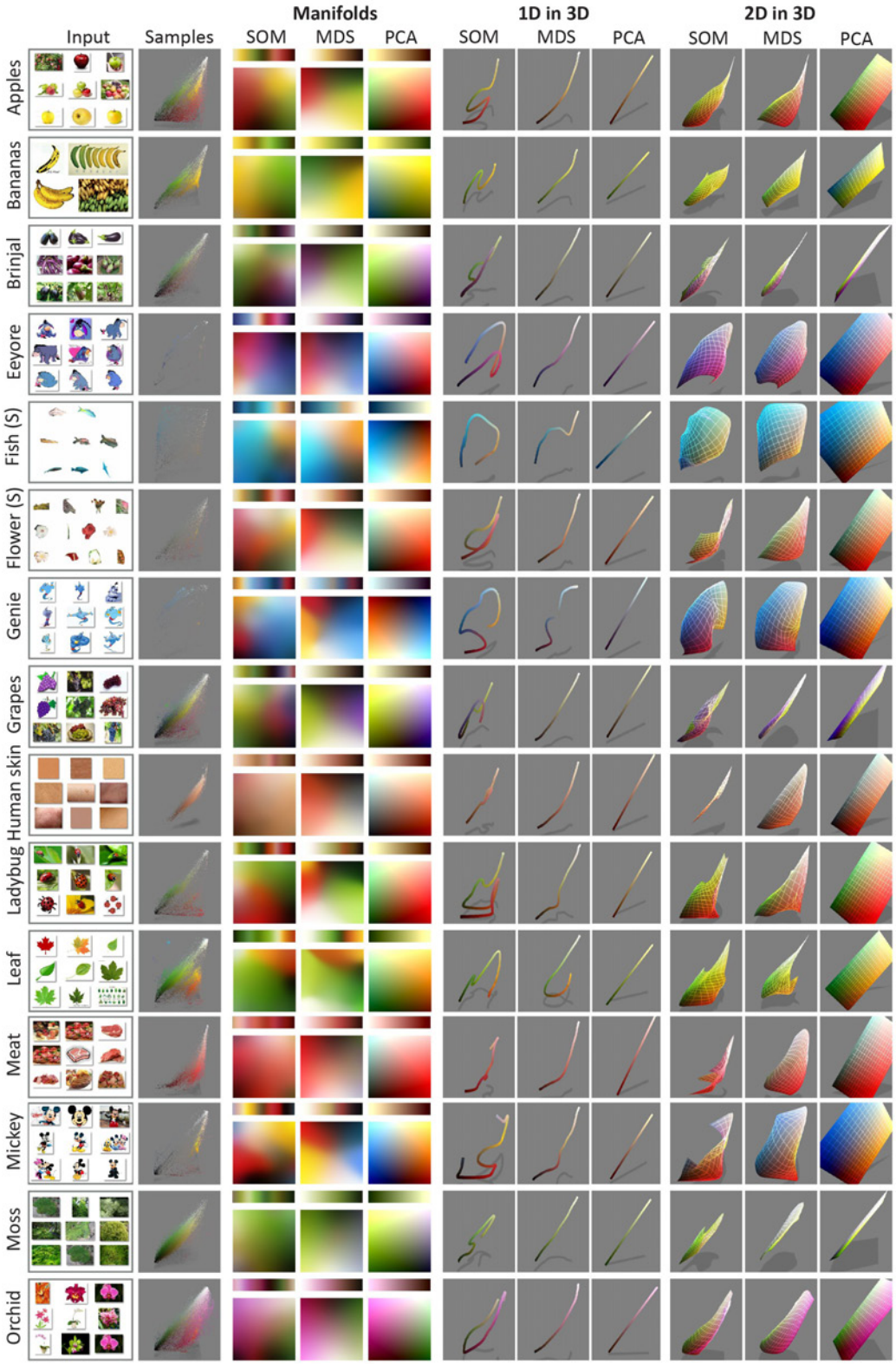

Figure 7.38: Manifolds constructed using different dimensionality reduction approaches. (S) denotes annotated images taken directly from the SUN database. Detailed explanations of this figure can be found in the caption of Figure 7.7.

8

Conclusion

We summarize the major contributions following the five main tracks: Preconvolved Radiance Caching, Surface Light Field Manipulation, Material Style Transfer, Shape-Color Subspaces, and Data-driven Color Manifolds in Section 8.1. In Section 8.2, we discuss possible future work.

8.1 Closing Remarks

Appearance editing is a time-intensive process that requires a lot of effort from artists due to the lack of tools and the long design cycle where they repeatedly tweak the parameters and wait for the visualization until the desired look is achieved. The techniques proposed in this thesis contribute to different blocks in the artist's design pipeline ranging from rendering over editing techniques to user interfaces. Preconvolved Radiance Caching (Chapter 3) is a fast global illumination rendering technique used for interactive visualization of synthetic virtual 3D scenes in Chapter 4 and Chapter 5. Surface Light Field Manipulation (Chapter 4), Material Style Transfer (Chapter 5) and Shape Color Subspaces (Chapter 6) are three techniques proposed for different appearance editing tasks. Finally, Data-driven Color Manifolds (Chapter 7) are new user interfaces, and can be incorporated easily into existing editing software packages.

A rendered 3D scene usually conveys a certain mood that the artists want to express. Material Style Transfer, an example-based material assignment approach where the 3D models of a virtual scene can be materialized simply by giving a guidance source (image/video), can serve as an initial setting to achieve the desired mood. Results of our system can be integrated easily into common design pipelines where further modification such as texture mapping or material appearance editing could be performed.

While Material Style Transfer is a fast and fully automatic approach, direct control by the artists is often mandatory, and usually further edits are required to converge to the final solution. To better support fine tune editing of the rendered images, Surface Light Field Manipulation allows artist to "paint" directly onto a virtual 2D canvas and the system finds the best reflectance to produce the desired appearance. Different from classic approaches where

artists need to tweak the non-intuitive parameters, our painting interface allows direct control over the final appearance where understanding of the underlying models is not necessary.

Tuning for a desired appearance, however, is not always an obvious task e. g., adjusting the skin color of human faces (either in images or virtual 3D scenes) to make it look real requires a lot of effort. To improve performance and quality of appearance editing in a design session, Data-driven Color Manifolds can be used as an alternative to the classic color pickers, e. g., the "human skin" manifold, represented as 1D or 2D slider, captures only possible colors of human skin should be used to adjust the skin color.

While Data-driven Color Manifolds encode most colors of a specific object, they do not capture the spatially varying nature of objects' colors, e. g., the comb of "chicken" is red, the feet are yellow and the tail is usually black. To alleviate this limitation, a morphable model of an object is constructed from an image collection. This model can be used to restrict the manipulation of shape and color in an image to a valid subspace.

In Chapter 3 we proposed an improvement of Radiance Caching that substantially reduces the linear cost of per-pixel reflection to a constant number of lookups (e. g., two for Phong) when interpolating caches. Correct querying of cache items [Szirmay-Kalos et al. 2005] has shown to be essential to our approach. This allows for reproduction of specular and normal map details between caches, even in interactive applications that previously were restricted to the (edge-aware) interpolation of scalar irradiance. The technique is used as the core framework for our Surface Light Field Manipulation proposed in Chapter 4 and can be used for interactive visualization of globally illuminated virtual 3D scenes in Chapter 5.

In Chapter 4 we proposed a system to manipulate a SLF by painting it from different views and a solver that infers valid reflectance from this input. Our approach does not expose the shading model and its parameters to the user at all and uses scribbles to infer changes in reflectance. Different from previous work, we exploit a fast GPU-based approach using pre-computed visibility and pre-convolved lumitexels on a point-based representation of the 3D scene (Chapter 3), allowing to freely move the view point, arbitrary spatially varying BRDFs and editing with fast pre-computation times.

Chapter 5 proposed a technique to transfer material style from a guide source to a target 3D scene. The problem was stated as a combinatorial optimization problem of assigning discrete materials (approximately extracted from the guide source) to discrete objects in the target according to an image and geometry cost function. We transfer materials that change their impact on the final result depending on their spatial context. The guide source does not only provide approximate material information, but the target 3D scene also provides a similar gist even when it is seen from several different views that allows our assignment to be more reliable than it could be when treating them as images only. Our system can effectively capture materials in terms of highlights and diffuse detail textures and efficiently apply them to a target scene or even whole databases in a meaningful and automatic way.

Chapter 6 proposed a system to restrict the manipulation of shape and color in an image to the subspace of valid changes which we learn from a collection of exemplar images. Different from previous approaches, we use an automatic alignment of exemplars to create the subspace from sample images. We hope that our subspaces will substantially increase the performance and quality of image editing tasks (as shown in our user study).

Chapter 7 proposed content-dependent color manifolds as a replacement for established general color spaces by embedding a curve or surface into a 3D color space. This allows for improved color selection, as validated by perceptual studies, color stylization, compression and white balancing. We hope that our intuitive 1D or 2D manifolds will substantially ease the navigation of color spaces (as shown in color adjustment study) and improve color editing quality in general (as shown in color exploration study).

8.2 Future Works

In this section, possible future research avenues of each component are discussed (Section 8.2.1), followed by their combinations (Section 8.2.2) and a general outlook to interesting open problems (Section 8.2.3).

8.2.1 Individuals

In this section, we will discuss possible future work of each component in detail.

Preconvolved Radiance Caching Future work could investigate the effect of (adaptive) cache item placement in 2D or 3D and its temporal coherence on rendering quality. While we use simple MIP mapping for pre-convolution, advanced methods with different pre-filtering can be used to approximate more complex BRDFs [Kautz et al. 2000], e. g., such BRDFs that depend on more than a single (gloss) parameter. Handling many *different classes* of BRDFs that all require different pre-convolution is an avenue of further investigation not just for our approach but for pre-convolution in general. While we used pre-convolution in combination with splatting (which does not fit modern GPUs well), it should also be applicable to gathering-type upsampling [Laine et al. 2007] in future work. Furthermore, we would like to generalize our idea to other phenomena that require integration, such as antialiasing, motion blur or depth of field, by placing cache items in the spatial, temporal or lens domain, performing pre-convolution and proper reconstruction.

Surface Light Field Manipulation Our system is only one instance from a class of approaches where users loosely manipulate rendered 3D images (e. g., using strokes) and the system infers sparse and physically meaningful parameter changes. Extensions to other physically-based rendering, such as depth-of-field, motion blur, participating media or binocular stereo are exciting avenues of future research. Here, the laws of physics are considered not for the sake of accuracy, but act as a regularizer to make the users changes work together and behave in a consistent and plausible way. Further research will be required to allow editing of shading models with multiple non-linear parameters. Currently, manipulated SLFs are inferred from the user's strokes, it would be interesting to support scanned SLFs as an alternative input. Finally, a user study comparing our approach to other alternatives, in the spirit of [Kerr and Pellacini 2010], is mandatory to assess if the proposed interface is indeed intuitive.

Material Style Transfer Recently, stock 3D models were used to guide the manipulation of images [Kholgade et. al. 2014], incorporating such semantics to constrain material properties of 3D objects would certainly improve the quality of material transfer. User-annotation (such as labels) and structure information (such as symmetry) could be included in future work. Support of more complex materials such as subsurface scattering, spatially varying BRDF and textured objects would certainly improve the realism of the materialized scenes. Finally, to simplify the optimization, lighting conditions are being fixed beforehand, it is interesting to relieve this constraint by incorporating an optimization for lighting properties, such as choosing a light fixture model from a light library, or different environment maps.

Shape and Color Subspaces Our alignment is currently limited to images that contain a single object with similar perspective, this prevents us from using Internet images that are downloaded directly from Google with certain keywords, such as "Horse". Relieving this constraint would certainly improve the practicality of our subspaces. Solving for the alignment problem in natural images, however, is a very challenging task. One possible direction is to perform "co-alignment" in the spirit of image co-segmentation [Joulin, Bach and Ponce 2010]. Furthermore, it would be interesting to generalize the approach to other domains such as 3D meshes and video to cover the mutual relation of artistic style, texture, reflectance, or other properties. Finally, while we preserve details of the target instance, we would also like to use our alignment to construct non-linear subspaces to capture more detailed variations in the subspaces.

Data-driven Color Manifolds In future work, we would like to extend this idea to very high or infinite-dimensional input spaces such as spectral colors or BDRFs while keeping smoothness, density and distance-preservation of our embedding. Another generalization could address texture variation or materials. Besides, comparison to different elaborate interfaces such as image galleries [Shapira, Shamir and Cohen-Or 2009] or color template [ODonovan et. al. 2011] in color adjustment and exploration tasks is an interesting future direction. Furthermore, to simplify the study, a detailed, soft selection is assumed to exist and subjects were asked to modify the images by picking a single color. This can be generalized in the spirit of “How do humans colorize images?” by allowing multiple color selections, in a manner similar to Cole et al. [2008].

8.2.2 Combinations

Interactive global illumination (Preconvolved Radiance Caching), appearance editing techniques (Surface Light Field Manipulation, Material Style Transfer, Shape Color Subspaces) and user interfaces (Data-driven Color Manifolds) are orthogonal and form building blocks in a design pipeline. Now we will discuss certain possible combinations in detail.

Inter-block Combinations Data-driven Color Manifolds could be used as a regularizer for Material Style Transfer and Surface Light Field Manipulation. To this end, objects with specified semantics are constrained to appearances of the relevant Data-driven Color Man-

ifolds, e. g., the "Apples" manifold. This would certainly improve the realism of Material Style Transfer and the quality of Surface Light Field Manipulation.

Data-driven Color Manifolds and Shape Color Subspaces are highly related. While the Shape Color Subspaces account for the spatial arrangement of colors (e. g., different parts of a chicken have different colors) that Color Manifolds cannot preserve , subspaces are less intuitive to explore compared to Color Manifolds. To alleviate this difficulty, subspace-aware edit propagation was proposed (Chapter 6). It would be interesting to investigate more intuitive interfaces, such as perceptually motivated parameter spaces (in the spirit of our 1D or 2D Data-driven Color Manifolds) and better parameter navigation schemes (in the spirit of image galleries [Shapira, Shamir and Cohen-Or 2009] or design galleries [Marks et. al. 1997]).

Intra-block Combinations The techniques proposed for different appearance editing tasks are closely related. As discussed earlier, Material Style Transfer can served as an initial solution that is later used for Surface Light Field Manipulations. Surface Light Field Manipulation can be complemented by Shape Color Subspaces to improve the editing quality. Combinations of these techniques are also possible.

Recently, Material Memex [Jain et al. 2012] models the correlation of materials found on 3D objects and their parts by learning from a database of 3D objects and their materials. This model can then be used to assign plausible materials to an object automatically or to provide material suggestions. Such databases of 3D objects are, however, limited and not easily accessible. In a similar manner, it is interesting to extract the correlation of shapes and colors in the easier accessible 2D Internet image collections, such as done for Shape Color Subspaces construction, and use them as a regularizer for Material Style Transfer.

8.2.3 General Outlook

Aside from the specific and combined directions for future research proposed in the preceding sections, the presented techniques developed in this thesis open up interesting general directions that need further investigation.

In the spirit of our color manifolds, intuitive appearance editing would benefit from projecting the high-dimensional space of content creation/editing algorithm parameters into a perceptually motivated lower dimensional space, e. g., 1D or 2D slider. One particular example is the recent work that aims for an intuitive exploration of fonts [ODonovan et. al. 2014]. We expect that the derivation of such models, if possible, requires robust regression from large sampled data collections combined with user experience.

From the user interfaces point of view, different software provides different interfaces for different editing purposes, e. g., Photoshop for 2D images and Maya or Blender for modeling of 3D scenes. For some editing tasks, certain software packages are more intuitive and easier to operate compared to others e. g., color editing of 2D images in Photoshop is more intuitive than material editing of 3D scenes in Blender where artists need to tweak more parameters and require understanding of material and lighting models; on the other hand, controlling the shape of specular highlights in 3D scenes is easier in Blender. Our Surface Light Field Manipulation is the first of its kind for material editing in 3D scenes that

supports an intuitive painting interface in a Photoshop-like manner, for future work, it would be interesting to adapt other intuitive editing interfaces from 2D images to 3D scenes. On the other hand, some editing tasks are easier to perform in 3D, recovering 3D information from RGB or RGB-D images that can later be used for image editing is an active research area [Khan et al. 2006; Karsch et al. 2011; Kholgade et. al. 2014].

From the data-collection point of view, automatic or semi-automatic annotated data, such as semantics of segmented objects in image collections, would further improve our techniques. This is an active research area in computer vision. Recently, data crowd sourcing has been applied to improve automatic intrinsic image decomposition [Bell et. al. 2014], it would be interesting to investigate whether a crowd sourcing approach would help in data annotation.

Finally, in the scope of this thesis, we only extracted plausible materials, colors and their spatial distributions from Internet images, more high level information, such as the correlation of shape and texture, are currently ignored. Such information, once successfully extracted, would likely spark more interesting applications.

8.3 Message

This thesis argued, how appearance editing of digital content can benefit from more intuitive approaches. Such approaches require further development of different components in a design pipeline ranging from rendering over editing techniques to user interfaces. Most applications of this work are targeting for novice users, who seek to edit their digital content. Possible directions are to adopt more intuitive approaches from cross-domain software packages or to exploit data-driven methodology, as the thesis pursued. Unlike most data-driven approaches where data need to be acquired in a fully calibrated environment, in this thesis, we opted to use Internet data in a semi-supervised manner that increases the practicability of our approaches. We foresee that many other tools that benefit from using Internet data will be developed in future work.

Bibliography (Own work)

NGUYEN, C. H., KYUNG, M.-H., LEE, J.-H. AND NAM, S.-W. (2010): A PCA Decomposition for Real-time BRDF Editing and Relighting with Global Illumination. *Comp. Graph. Forum (Proc. EGSR)*, 4 (29), 1469–1478 18

NGUYEN, C. H., NALBACH, O., RITSCHEL, T. AND SEIDEL, H.-P. (2015b): Guiding Image Manipulations using Shape-appearance Subspaces from Co-alignment of Image Collections. *Comp. Graph. Forum (Proc. EUROGRAPHICS)* 2 (34) 4

NGUYEN, C. H., RITSCHEL, T., EISEMANN, E., MYSZKOWSKI, K. AND SEIDEL, H.-P. (2012): 3D Material Style Transfer. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 4 (29), 1469–1478 4

NGUYEN, C. H., RITSCHEL, T. AND SEIDEL, H.-P. (2015): Data-driven Color Manifolds. *ACM Trans. Graphs. (Presented at SIGGRAPH 2015)* 2 (34) 4, 5

NGUYEN, C. H., SCHERZER, D., RITSCHEL, T. AND SEIDEL, H.-P. (2013): Material Editing in Complex Scenes by Surface Light Field Manipulation and Reflectance Optimization. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 2 (32), 185–194 4

SCHERZER, D., NGUYEN, C. H., RITSCHEL, T. AND SEIDEL, H.-P. (2012): Pre-convolved Radiance Caching. *Comp. Graph. Forum (Proc. EGSR 2012)*, 4 (31), 1391–1397 4, 51

Bibliography

ADOBE SYSTEMS INCORPORATED (2014a): Adobe Color. ⟨URL: `https://color.adobe.com/create/color-wheel/`⟩ 17, 18

ADOBE SYSTEMS INCORPORATED (2014b): Adobe Photoshop. ⟨URL: `http://www.adobe.com/products/photoshop.html`⟩ 2

ALEXA, M., COHEN-OR, D. AND LEVIN, D. (2000): As-rigid-as-possible shape interpolation. In *Proc. SIGGRAPH*, 157–164 19

ALLEN, B., CURLESS, B. AND POPOVIĆ, Z. (2003): The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph.* 22 (3), 587–594 20, 76

AN, X. AND PELLACINI, F. (2008): AppProp: All-pairs appearance-space edit propagation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 3, 40:1–40:9 2, 19, 90

ANJYO, K.-I. AND HIRAMITSU, K. (2003): Stylized Highlights for Cartoon Rendering and Animation. *IEEE Comput. Graph. Appl.* 23 (4), 54–61 18, 19

ARVO, J., TORRANCE, K. AND SMITS, B. (1994): A framework for the analysis of error in global illumination algorithms. In *Proc. SIGGRAPH*, 75–84 12

ASHIKMIN, M. AND SHIRLEY, P. (2000): An anisotropic Phong light reflection model. *Journal of Graphics Tools*, 5, 25–32 10

AUTODESK INC (2014): AutoDesk Maya. ⟨URL: `http://www.autodesk.com/products/maya/overview`⟩ 2

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A. AND GOLDMAN, D. B. (2009): PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 28 (3), 24:1–24:11 20, 82

BELL, S., BALA, K. AND SNAVELY, N. (2014): Intrinsic Images in the Wild. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33 (4), 159:1–159:12 134

BELL, S., UPCHURCH, P., SNAVELY, N. AND BALA, K. (2013): OpenSurfaces: A Richly Annotated Catalog of Surface Appearance. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 32 (4), 111:1–111:17 119, 120

BELONGIE, S., MALIK, J. AND PUZICHA, J. (2000): Shape context: A new descriptor for shape matching and object recognition. In *NIPS* Volume 2,, 3 20, 82, 85

BEN-ARTZI, A., OVERBECK, R. AND RAMAMOORTHI, R. (2006): Real-time BRDF editing in complex lighting. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 25 (3), 945–954 2, 3, 18

BEN-ARTZI, A., EGAN, K., DURAND, F. AND RAMAMOORTHI, R. (2008): A precomputed polynomial representation for interactive BRDF editing with global illumination. *ACM Trans. Graph.* 27 (2), 13:1–13:13 18

BERG, E. A. (1948): A simple objective technique for measuring flexibility in thinking. *J General Psychology*, 39, 15–22 21

BLANZ, V. AND VETTER, T. (1999): A morphable model for the synthesis of 3D faces. In *Proc. SIGGRAPH*, 187–194 3, 19, 20, 76, 85

BLENDER FOUNDATION (2014): Blender. ⟨URL: `http://www.blender.org`⟩ 2

BLINN, J. F. (1977): Models of Light Reflection for Computer Synthesized Pictures. *SIGGRAPH Comput. Graph.* 11 (2), 192–198 2, 10

BOBER, M. (2001): MPEG-7 visual shape descriptors. *IEEE Circuits and Sys. Vid. Tech.* 11 (6), 716–719 66

BOOKSTEIN, F. L. (1989): Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE PAMI*, 11 (6), 567–585 19

CAETANO, T., MCAULEY, J., CHENG, L., LE, Q. V. AND SMOLA, A. (2009): Learning graph matching. *IEEE PAMI*, 31 (6), 1048–1058 20

CAMPBELL, N. D. F. AND KAUTZ, J. (2014): Learning a Manifold of Fonts. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33 (4), 91:1–91:11 22

CASHMAN, T. J. AND FITZGIBBON, A. W. (2013): What Shape Are Dolphins? Building 3D Morphable Models from 2D Images. *PAMI*, 35 (1), 232–244 20, 76

CASTELHANO, M. S. AND HENDERSON, J. M. (2010): The influence of color on perception of scene gist. *J Vision*, 5 (8), 68–78 21

CHAJDAS, M. G., LEFEBVRE, S. AND STAMMINGER, M. (2010): Assisted texture assignment. In *Proc. I3D*, 173 21

CHANDRASEKHAR, S. (1960): Radiative Transfer., Dover Books on Intermediate and Advanced Mathematics 2

CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T. AND OUHYOUNG, M. (2003): On Visual Similarity Based 3D Model Retrieval. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 22 (3), 223–232 66, 67

CHENG, M.-M., ZHANG, F.-L., MITRA, N. J., HUANG, X. AND HU, S.-M. (2010): RepFinder: Finding Approximately Repeated Scene Elements for Image Editing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 29 (4), 83:1–83:8 20, 82

CHESLACK-POSTAVA, E., WANG, R., AKERLUND, O. AND PELLACINI, F. (2008): Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 27 (5), 128:1–128:10 18

CHRISTENSEN, P. (2008): Point-based approximate color bleeding. Pixar (08-01). – Technical report 16, 31

CIE (1931): Commission Internationale de l'Eclairage Proceedings, 1931. 9

CLAUSEN, C. AND WECHSLER, H. (2000): Color image compression using PCA and backpropagation learning. *Pattern Recognition*, 33 (9), 1555–60 9

COHEN, J. (1988): Statistical power analysis for the behavioral sciences. 24, 26

COHEN-OR, D., SORKINE, O., GAL, R., LEYVAND, T. AND XU, Y. (2006): Color harmonization. In *ACM Trans. Graph. (TOG)* Volume 25,, 624–630 18

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T. AND RUSINKIEWICZ, S. (2008): Where Do People Draw Lines? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 27 (3), 88:1–88:11 132

COMANICIU, D. AND MEER, P. (2002): Mean shift: a robust approach toward feature space analysis. *IEEE PAMI*, 24 (5), 603–19 67

COOK, R. L. AND TORRANCE, K. E. (1982): A Reflectance Model for Computer Graphics. *ACM Trans. Graph.* 1 (1), 7–24 2, 10

COOTES, T. F., EDWARDS, G. J. AND TAYLOR, C. J. (2001): Active appearance models. *IEEE PAMI*, 23 (6), 681–85 20, 76

COX, T. AND COX, M. (2000): Multidimensional scaling. Volume 88, 104

CRASSIN, C., NEYRET, F., LEFEBVRE, S. AND EISEMANN, E. (2009): GigaVoxels: Ray-guided Streaming for Efficient and Detailed Voxel Rendering. In *Proc. I3D*, 15–22 16

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S. AND EISEMANN, E. (2011): Interactive Indirect Illumination Using Voxel Cone Tracing. In *Proc. PG* Volume 30,, 1921–1930 16

CUNO, A., ESPERANÇA, C., OLIVEIRA, A. AND CAVALCANTI, P. R. (2007): 3D as-rigid-as-possible deformations using MLS. In *Proc. CGI*, 115–122 19

DANIEL, W. W. (1990): Applied nonparametric statistics. 25

DOERSCHNER, K., MALONEY, L. T. AND BOYACI, H. (2010): Perceived glossiness in high dynamic range scenes. *J. Vision*, 10 (9), 11 ⟨URL: `http://www.ncbi.nlm.nih.gov/pubmed/20936748`⟩ 11

DONG, Y., TONG, X., PELLACINI, F. AND GUO, B. (2011): AppGen: Interactive Material Modeling from a Single Image. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 30 (6), 146:1–146:10 11

DONG, Z., CHEN, W., BAO, H., ZHANG, H. AND PENG, Q. (2004): Real-time voxelization for complex polygonal models. In *Proc. PG* IEEE, 43–50 16

DONOHO, D. (2006): Compressed sensing. *Inf. Theory, IEEE Trans.* 52 (4), 1289–1306 48

DORSEY, J., SILLION, F. X. AND GREENBERG, D. P. (1991): Design and simulation of opera lighting and projection effects. In *Proc. SIGGRAPH*, 41–50 2

DOUGLAS, S. AND KIRKPATRICK, A. (1999): Model and representation: the effect of visual feedback on human performance in a color picker interface. *ACM Trans. Graph.* 18 (2), 96–127 17, 99

DURAND, F. AND DORSEY, J. (2002): Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21 (3), 257–266 63

EISEMANN, E. AND DÉCORET, X. (2008): Single-pass GPU solid voxelization for real-time applications. In *Proc. Graphics Interface* Canadian Information Processing Society, 73–80 16

FAIRCHILD, M. (2005): Color appearance models. 7

FATTAL, R., AGRAWALA, M. AND RUSINKIEWICZ, S. (2007): Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26 (3), 51–60 64

FEISNER, E. AND REED, R. (2013): Color Studies. 2

FENG, W.-W., KIM, B.-U. AND YU, Y. (2008): Real-time data driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 27 (3), 91 20

FLEMING, R., DROR, R. AND ADELSON, E. (2003): Real-world illumination and the perception of surface reflectance properties. *J. Vision*, 3 (5), 1–10 ⟨URL: `http://www.journalofvision.org/content/3/5/3.short`⟩ 11

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S. AND DOBKIN, D. (2004): Modeling by Example. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23 (3), 652–663 3

GASTAL, E. S. L. AND OLIVEIRA, M. M. (2011): Domain Transform for Edge-aware Image and Video Processing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30 (4), 69:1–69:12 19, 88, 90, 91, 93, 114

GAUTRON, P., KŘIVÁNEK, J., BOUATOUCH, K. AND PATTANAIK, S. (2008): Radiance cache splatting: A GPU-friendly global illumination algorithm. In *ACM SIGGRAPH 2008 Classes* ACM, 78–88 51

GAUTRON, P., KŘIVÁNEK, J., PATTANAIK, S. AND BOUATOUCH, K. (2004): A novel hemispherical basis for accurate and efficient rendering. In *Proc. EGSR*, 55–64 15, 32

GAUTRON, P., KŘIVÁNEK, J., BOUATOUCH, K. AND PATTANAIK, S. N. (2005): Radiance Cache Splatting: A GPU-Friendly Global Illumination Algorithm. In *Proc. EGSR*, 55–64 15

GINGOLD, Y. AND ZORIN, D. (2008): Shading-based Surface Editing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 27 (3), 95:1–95:9 18

GOLDBERG, C., CHEN, T., ZHANG, F.-L., SHAMIR, A. AND HU, S.-M. (2012): Data-Driven Object Manipulation in Images. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 31 (2pt1), 265–274 20, 85

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R. AND COHEN, M. F. (1996): The Lumigraph. In *Proc. SIGGRAPH*, 43–54 12

GRAPHPAD (2014): GraphPad Statistics Guide. ⟨URL: `http://www.graphpad.com/guides/prism/6/statistics/`⟩ – visited on 03-10-2014 24

GRAVETTER, F. AND WALLNAU, L. (2013): Statistics for the behavioral sciences. 22, 24

GREENE, N. (1986): Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6 (11), 21–29 15

GREGER, G., SHIRLEY, P., HUBBARD, P. AND GREENBERG, D. (1998): The irradiance volume. *IEEE Computer Graphics and Applications*, 18 (2), 32–43 13

HACOHEN, Y., SHECHTMAN, E., GOLDMAN, D. B. AND LISCHINSKI, D. (2011): Non-rigid Dense Correspondence with Applications for Image Enhancement. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30 (4), 70:1–70:10 20, 82

HAUMONT, D. AND WARZÉE, N. (2002): Complete polygonal scene voxelization. *Journal of Graphics Tools*, 7 (3), 27–41 16

HAŠAN, M., PELLACINI, F. AND BALA, K. (2006): Direct-to-indirect Transfer for Cinematic Relighting. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 25 (3), 1089–1097 18

HE, K., SUN, J. AND TANG, X. (2013): Guided Image Filtering. *IEEE PAMI*, 35 (6), 1397–1409 90

HEATH, K., GELFAND, N., OVSJANIKOV, M., AANJANEYA, M. AND GUIBAS, L. (2010): Image webs: Computing and exploiting connectivity in image collections. In *Proc. CVPR*, 3432–3439 20

HEEGER, D. J. AND BERGEN, J. R. (1995): Pyramid-based texture analysis and synthesis. In *Proc. SIGGRAPH*, 229–238 21

HEIDRICH, W. AND SEIDEL, H. (1999): Realistic, hardware-accelerated shading and lighting. In *Proc. SIGGRAPH*, 171–178 15, 29, 30, 51

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B. AND SALESIN, D. H. (2001): Image analogies. In *Proc. SIGGRAPH*, 327–340 21

HOFF III, K., KEYSER, J., LIN, M., MANOCHA, D. AND CULVER, T. (1999): Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proc. SIGGRAPH*, 277–286 50

HORN, B. K. (1987): Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4 (4), 629–642 84

HORN, D. AND CHEN, B. (2007): LightShop: Interactive light field manipulation and rendering. In *Proc. I3D* 13

HSU, R.-L., ABDEL-MOTTALEB, M. AND JAIN, A. (2002): Face detection in color images. *IEEE PAMI*, 24 (5), 696 –706 9, 17

HUANG, Q.-X., ZHANG, G.-X., GAO, L., HU, S.-M., BUTSCHER, A. AND GUIBAS, L. (2012): An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 31 (6), 167 80

HUBER, D. F. (2002): Automatic three-dimensional modeling from reality. Ph. D thesis, Carnegie Mellon U, Pittsburgh 80

IGARASHI, T., MOSCOVICH, T. AND HUGHES, J. F. (2005): As-rigid-as-possible shape manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24 (3), 1134–41 19

IMMEL, D., COHEN, M. AND GREENBERG, D. (1986): A radiosity method for non-diffuse environments. *Proc. SIGGRAPH*, 20 (4), 133–142 45

IRONY, R., COHEN-OR, D. AND LISCHINSKI, D. (2005): Colorization by Example. *Proc. EGSR*, 1, 201–210 21

JAIN, A., THORMÄHLEN, T., RITSCHEL, T. AND SEIDEL, H.-P. (2012): Material Memex: Automatic Material Suggestions for 3D Objects. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2012)* 31 (5) 133

JIA, Y. (2013): Caffe: An open source convolutional architecture for fast feature embedding. 125

JOHNSON, M. K., DALE, K., AVIDAN, S., PFISTER, H., FREEMAN, W. T. AND MATUSIK, W. (2010): CG2Real: Improving the Realism of Computer Generated Images using a Large Collection of Photographs. *IEEE TVCG*, 17, 1273–1285 21

JOULIN, A., BACH, F. AND PONCE, J. (2010): Discriminative clustering for image co-segmentation. In *Proc. CVPR* IEEE, 1943–1950 132

KAJIYA, J. T. (1986): The Rendering Equation. *Comput. Graph. (Proc. SIGGRAPH)*, 20 (4), 143–150 2, 11

KARSCH, K., HEDAU, V., FORSYTH, D. AND HOIEM, D. (2011): Rendering Synthetic Objects into Legacy Photographs. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 30 (6), 157:1–157:12 134

KAUTZ, J., VÁZQUEZ, P., HEIDRICH, W. AND SEIDEL, H. (2000): A unified approach to prefiltered environment maps. In *Proc. EGWR* Volume 6, 16, 30, 131

KAWAI, J. K., PAINTER, J. S. AND COHEN, M. F. (1993): Radioptimization: goal based rendering. In *Proc. SIGGRAPH*, 147–154 18

KELLER, A. (1997): Instant radiosity. In *Proc. SIGGRAPH*, 49–56 16, 31, 68

KERR, W. B. AND PELLACINI, F. (2010): Toward evaluating material design interface paradigms for noviceusers. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 29 (3), 35:1–35:10 3, 19, 39, 131

KHAN, E. A., REINHARD, E., FLEMING, R. W. AND BÜLTHOFF, H. H. (2006): Image-based material editing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 25 (3), 654–662 2, 134

KHOLGADE, N., SIMON, T., EFROS, A. AND SHEIKH, Y. (2014): 3D Object Manipulation in a Single Photograph Using Stock 3D Models. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33 (4), 127:1–127:12 132, 134

KIRKPATRICK, S., JR, C. D. G. AND VECCHI, M. P. (1983): Optimization by simmulated annealing. *Science*, 220 (4598), 671–680, ISSN 10959203 67

KOHONEN, T. (1990): The self-organizing map. *Proceedings of the IEEE*, 78 (9), 1464–1480 105

KOKKINOS, I. AND YUILLE, A. (2007): Unsupervised learning of object deformation models. In *Proc. CVPR*, 1–8 20

KŘIVÁNEK, J., GAUTRON, P., PATTANAIK, S. AND BOUATOUCH, K. (2005): Radiance caching for efficient global illumination computation. *IEEE Trans. Vis. Comp. Graph.* 11 (5), 550–61 13, 15, 28

LAFFONT, P.-Y., REN, Z., TAO, X., QIAN, C. AND HAYS, J. (2014): Transient Attributes for High-level Understanding and Editing of Outdoor Scenes. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33 (4), 149:1–149:11 21

LAFORTUNE, E. P. AND WILLEMS, Y. D. (1993): Bi-Directional Path Tracing. In *Proc. Computational Graphics and Visualization Techniques*, 145–153 13

LAGAE, A., VANGORP, P., LENAERTS, T. AND DUTRÉ, P. (2010): Procedural isotropic stochastic textures by example. *Computers & Graphics*, 34 (4), 312–321 21

LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J. AND AILA, T. (2007): Incremental instant radiosity for real-time indirect illumination. In *Proc. EGSR*, 277–286 16, 131

LALONDE, J.-F. AND EFROS, A. A. (2007): Using color compatibility for assessing image realism. In *Proc. ICCV*, 1–8 21

LAND, E. H. AND MCCANN, J. J. (1971): Lightness and retinex theory. *J OSA*, 61 (1), 1–11 11

LANG, M., WANG, O., AYDIN, T., SMOLIC, A. AND GROSS, M. (2012): Practical Temporal Consistency for Image-based Graphics Applications. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31 (4), 34:1–34:8 20, 84, 85

LANGER, M. S. (1999): When shadows become interreflections. *J Comp. Vis.* 34 (2), 193–204 11

LAVANYA SHARAN, R. R. . E. H. A. (2008): Rapid visual perception of material properties. In *Proc. Workshop on Object Perception, Attention & Memory* 21

LAWRENCE, R. D., ALMASI, G. S. AND RUSHMEIER, H. E. (1999): A Scalable Parallel Algorithm for Self-Organizing Maps with Applicationsto Sparse Data Mining Problems. *Data Min. Knowl. Discov.* 3 (2), 171–195, ISSN 1384–5810 106

LAZEBNIK, S., SCHMID, C. AND PONCE, J. (2006): Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR* Volume 2,, 2169–78 100

LENSCH, H. P. A., KAUTZ, J., GOESELE, M., HEIDRICH, W. AND SEIDEL, H.-P. (2003): Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph.* 22 (2), 234–257 11

LEVIN, A., LISCHINSKI, D. AND WEISS, Y. (2004): Colorization using optimization. *ACM Trans. Graph.* 23 (3), 689–94 19, 90, 95

LEVOY, M. AND HANRAHAN, P. (1996): Light field rendering. In *Proc. SIGGRAPH* 12

LI, Y., JU, T. AND HU, S.-M. (2010): Instant Propagation of Sparse Edits on Images and Videos. *Comp. Graph. Forum (Proc. Pacific Graphics)*, 29 (7), 2049–2054 90

LIAO, J., LIMA, R., NEHAB, D., HOPPE, H., SANDER, P. AND YU, J. (2014): Automating image morphing using structural similarity on a halfway domain. *ACM Trans. Graphics*, 33 (3), 1–14 20

LISCHINSKI, D., FARBMAN, Z., UYTTENDAELE, M. AND SZELISKI, R. (2006): Interactive local adjustment of tonal values. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 25 (3), 646–653 2, 3, 19

LISSNER, I. AND URBAN, P. (2009): How Perceptually Uniform Can a Hue Linear Color Space Be? In *Proc. IST CIC*, 97–102 10

LIU, C., YUEN, J. AND TORRALBA, A. (2011): SIFT flow: Dense correspondence across scenes and its applications. *IEEE PAMI*, 33 (5), 978–994 20, 82

LIU, X., WAN, L., QU, Y., WONG, T.-T., LIN, S., LEUNG, C.-S. AND HENG, P.-A. (2008): Intrinsic colorization. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 27 (5), 1–9 21

LUAN, Q., WEN, F., COHEN-OR, D., LIANG, L., XU, Y.-Q. AND SHUM, H.-Y. (2007): Natural Image Colorization. In *Proc. EGSR* 21

LUCAS, B. D. AND KANADE, T. (1981): An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proc. IJCAI*, 674–679 20, 82

MACQUEEN, J. (1967): Some methods for classification and analysis of multivariate observations. In *Proc. Berkeley Symp. Math. Stat. Prob.* Volume 1,, 14 63

MALGOUYRES, R. (2002): A discrete radiosity method. In *Discrete Geometry for Computer Imagery* Springer, 428–438 16

MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUML, W., RYALL, K., SEIMS, J. AND SHIEBER, S. (1997): Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proc. SIGGRAPH*, 389–400 22, 93, 133

MARONNA, R. A., MARTIN, D. R. AND YOHAI, V. J. (2006): Robust Statistics: Theory and Methods. 125

MATSUDA, Y. (1995): Color design. *Asakura Shoten*, 2 (4), 1–23 18

MATTAUSCH, O., IGARASHI, T. AND WIMMER, M. (2013): Freeform Shadow Boundary Editing. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 32, 175–184, ISSN 1467–8659 18

MATUSIK, W., PFISTER, H., BRAND, M. AND MCMILLAN, L. (2003): A Data-Driven Reflectance Model. In *Proc. SIGGRAPH*, 759–768 11, 22

MAXWELL, S. E. AND DELANEY, H. D. (2004): Designing experiments and analyzing data: A model comparison perspective. Volume 1, 25

MERRELL, P., SCHKUFZA, E. AND KOLTUN, V. (2010): Computer-generated Residential Building Layouts. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 29 (6), 181:1–181:12 3

MERRELL, P., SCHKUFZA, E., LI, Z., AGRAWALA, M. AND KOLTUN, V. (2011): Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30 (4), 87–97 3

MERTENS, T., KAUTZ, J., CHEN, J. AND DURAND, F. (2007): Texture Transfer Using Geometry Correlation. In *Proc. EGSR* 21

MIGUEL, E., BRADLEY, D., THOMASZEWSKI, B., BICKEL, B., MATUSIK, W., OTADUY, M. A. AND MARSCHNER, S. (2012): Data-Driven Estimation of Cloth Simulation Models. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 31 (2), 519–528 3

MILLER, G., RUBIN, S. AND PONCELEON, D. (1998): Lazy decompression of surface light fields for precomputed global illumination. In *Proc. EGWR*, 281–292 13

MILLER, R. G. (1966): Simultaneous statistical inference. 24

MORONEY, N., FAIRCHILD, M. D., HUNT, R. W., LI, C., LUO, M. R. AND NEWMAN, T. (2002): The CIECAM02 color appearance model. In *Color and Imaging Conference* Volume 2002, Society for Imaging Science and Technology, 23–27 9

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M. AND GROSS, M. (2005): Meshless deformations based on shape matching. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24 (3), 471–478 2, 89

NG, R., RAMAMOORTHI, R. AND HANRAHAN, P. (2003): All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22 (3), 376–381 3, 16, 18, 28

NG, R., RAMAMOORTHI, R. AND HANRAHAN, P. (2004): Triple Product Wavelet Integrals for All-frequency Relighting. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23 (3), 477–487 18

NISHIDA, S. AND SHINYA, M. (1998): Use of image-based information in judgments of surface-reflectance properties. *JOSA A*, 15 (12), 2951–65 11

OBEIN, G., KNOBLAUCH, K. AND VIÉOT, F. (2004): Difference scaling of gloss: Nonlinearity, binocularity, and constancy. *J Vision*, 4 (9), 711–720 11

O'DONOVAN, P., AGARWALA, A. AND HERTZMANN, A. (2011): Color compatibility from large datasets. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30 (4), 1–11 18, 64, 99, 132

O'DONOVAN, P., LIBEKS, J., AGARWALA, A. AND HERTZMANN, A. (2014): Exploratory Font Selection Using Crowdsourced Attributes. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33 (4), 1–9 133

OLIVA, A. AND TORRALBA, A. (2001): Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *Int. J. Computer Vision*, 42 (3), 145–175 21

OLKKONEN, M., HANSEN, T. AND GEGENFURTNER, K. R. (2008): Color appearance of familiar objects: Effects of object shape, texture, and illumination changes. *J Vision*, 8 (5), 1–16 123

OMER, I. AND WERMAN, M. (2004): Color lines: Image specific color representation. In *Proc. CVPR* Volume 2,, II–946 10, 17

OSKAM, T., HORNUNG, A., SUMNER, R. AND GROSS, M. (2012): Fast and Stable Color Balancing for Images and Augmented Reality. In *Proc. 3DIMPVT*, 49–56 22

PARZEN, E. (1962): On estimation of a probability density function and mode. *Ann. Math. Statist*, 33, 1065–1076 66

PELLACINI, F., TOLE, P. AND GREENBERG, D. P. (2002): A user interface for interactive cinematic shadow design. *ACM Trans. Graph.* 21 (3), 563–566 18

PELLACINI, F., FERWERDA, J. A. AND GREENBERG, D. P. (2000): Toward a psychophysically-based light reflection model for image synthesis. In *Proc. SIGGRAPH*, 55–64 2, 11, 47

PELLACINI, F. AND LAWRENCE, J. (2007): AppWand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26 (3), 54 2, 19, 68

PELLACINI, F., VIDIMČE, K., LEFOHN, A., MOHR, A., LEONE, M. AND WARREN, J. (2005): Lpics: A Hybrid Hardware-accelerated Relighting Engine for Computer Cinematography. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24 (3), 464–470 3, 18

PERLIN, K. (1985): An image synthesizer. *Computer Graphics (Proc. SIGGRAPH)*, 19 (3), 287–296 63

PHONG, B. (1975): Illumination for computer generated pictures. *Communications of the ACM*, 18 (6), 311–317 2, 10, 29, 63

POULIN, P. AND FOURNIER, A. (1995): Painting surface characteristics. In *Proc. EGWR*, 160–9 18

RAMAMOORTHI, R. AND HANRAHAN, P. (2001): An efficient representation for irradiance environment maps. In *Proc. SIGRAPH*, 497–500 16, 28, 30, 32

REINHARD, E., ASHIKHMIN, M., GOOCH, B. AND SHIRLEY, P. (2001): Color Transfer between Images. *IEEE Computer Graphics and Applications*, 21 (5), 34–41 2, 3, 21

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H., KAUTZ, J. AND DACHSBACHER, C. (2009a): Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 28 (5), 132–141 16, 31

RITSCHEL, T., GROSCH, T., KIM, M., SEIDEL, H., DACHSBACHER, C. AND KAUTZ, J. (2008): Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 27 (5), 129–137 16

RITSCHEL, T., OKABE, M., THORMÄHLEN, T. AND SEIDEL, H.-P. (2009b): Interactive Reflection Editing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 28 (5), 129:1–129:7 19

RITSCHEL, T., THORMÄHLEN, T., DACHSBACHER, C., KAUTZ, J. AND SEIDEL, H.-P. (2010): Interactive On-surface Signal Deformation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 29 (4), 36:1–36:8 18

RUSSELL, B., FREEMAN, W., EFROS, A., SIVIC, J. AND ZISSERMAN, A. (2006): Using Multiple Segmentations to Discover Objects and their Extent in Image Collections. In *Proc. CVPR*, 1605–1614 67

SADEGHI, I., PRITCHETT, H., JENSEN, H. W. AND TAMSTORF, R. (2010): An Artist Friendly Hair Shading System. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 29 (4), 56:1–56:10 2, 11

SAITO, T. AND TAKAHASHI, T. (1990): Comprehensible rendering of 3-D shapes. *Computer Graphics (Proc. SIGGRAPH)*, 24 (4), 197–206 68

SCHAEFER, S., MCPHAIL, T. AND WARREN, J. (2006): Image deformation using moving least squares. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 25 (3), 533–40 2, 3, 19, 84, 88, 89, 91

SCHLICK, C. (1994): An Inexpensive BRDF Model for Physically-based Rendering. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 13 (3), 233–246 10

SCHMIDT, T.-W., PELLACINI, F., NOWROUZEZAHRAI, D., JAROSZ, W. AND DACHSBACHER, C. (2014): State of the Art in Artistic Editing of Appearance, Lighting, and Material. In *Eurographics 2014 - State of the Art Reports* 19

SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J. AND GREENBERG, D. (1993): Painting with light. In *Proc. SIGGRAPH*, 143–146 18

SCHWARZ, M. W., COWAN, W. B. AND BEATTY, J. C. (1987): An experimental comparison of RGB, YIQ, LAB, HSV, and opponent color models. *ACM Trans. Graph.* 6 (2), 123–158 17, 99

SEOL, Y., LEWIS, J. P., SEO, J., CHOI, B., ANJYO, K. AND NOH, J. (2012): Spacetime expression cloning for blendshapes. *ACM Trans. Graph.* 31 (2), 14 87, 89

SERRE, T., WOLF, L., BILESCHI, S., RIESENHUBER, M. AND POGGIO, T. (2007): Robust object recognition with cortex-like mechanisms. *IEEE PAMI*, 29 (3), 411–426 80

SEUNG, H. S. AND LEE, D. D. (2000): The manifold ways of perception. *Science*, 290 (5500), 2268–69 76

SHAPIRA, L., SHAMIR, A. AND COHEN-OR, D. (2009): Image Appearance Exploration by Model-Based Navigation. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 28 (2), 629–38 17, 132, 133

SHARMA, G., WU, W. AND DALAL, E. N. (2005): The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30 (1), 21–30 63

SHIH, Y., PARIS, S., DURAND, F. AND FREEMAN, W. T. (2013): Data-driven Hallucination of Different Times of Day from a Single Outdoor Photo. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 32, 200:1–200:11 89

SIEGEL, S. AND CASTELLAN, N. J. (1988): Nonparametric Statistics for The Behavioral Sciences. 24, 116

SLOAN, P., GOVINDARAJU, N., NOWROUZEZAHRAI, D. AND SNYDER, J. (2007): Image-based proxy accumulation for real-time soft global illumination. In *Proc. Pacific Graphics*, 97–105 16

SLOAN, P., KAUTZ, J. AND SNYDER, J. (2002): Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Trans. Graph. (Proc. SIGGRAPH)* Volume 21,, 527–536 3, 15, 16, 18

STROOP, J. R. (1935): Studies of interference in serial verbal reactions. *J Exp. Psy.: General*, 18 (6), 643–662 21, 62

SUMNER, R. W., ZWICKER, M., GOTSMAN, C. AND POPOVIĆ, J. (2005): Mesh-based inverse kinematics. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24 (3), 488–495 20

SWAIN, M. J. AND BALLARD, D. H. (1991): Color indexing. *Int. J Computer Vision*, 7 (1), 11–32 66

SZIRMAY-KALOS, L., ASZÓDI, B., LAZÁNYI, I. AND PREMECZ, M. (2005): Approximate Ray-Tracing on the GPU with Distance Impostors. *Comp. Grap. Forum (Proc. EUROGRAPHICS)*, 24 (3), 695–704 31, 51, 130

TABELLION, E. AND LAMORLETTE, A. (2004): An approximate global illumination system for computer generated films. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23 (3), 469–476 15

TAO, M., BAI, J., KOHLI, P. AND PARIS, S. (2012): SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 31 (2pt1), 345–353 20, 82, 83, 84, 85

TENENBAUM, J., DE SILVA, V. AND LANGFORD, J. (2000): A global geometric framework for nonlinear dimensionality reduction. *Science*, 290 (5500), 2319–23 104

TKALCIC, M. AND TASIC, J. (2003): Colour spaces: perceptual, historical and applicational background. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8* Volume 1,, 304–308 vol.1 9

TOMBARI, F., MATTOCCIA, S., DI STEFANO, L. AND ADDIMANDA, E. (2008): Classification and evaluation of cost aggregation methods for stereo correspondence. In *Proc. CVPR*, 1–8 83

TOMINAGA, S. AND TANAKA, N. (2000): Estimating reflection parameters from a single color image. *IEEE Computer Graphics and Applications*, 20 (5), 58–66 10

TRIMBLE NAVIGATION (2014): SketchUp. ⟨URL: `http://www.sketchup.com`⟩ 2

TURK, M. AND PENTLAND, A. (1991): Eigenfaces for recognition. *J Cog. Neuroscience*, 3 (1), 71–86 3, 19, 76

TVERSKY, A. AND KAHNEMAN, D. (1981): The framing of decisions. *Science*, 211, 453–458 76

TZENG, D. AND BERNS, R. (2005): A review of principal component analysis and its applications to color technology. *Color Res. & App.* 30 (2), 84–98 10

VANGORP, P., LAURIJSSEN, J. AND DUTRÉ, P. (2007): The influence of shape on the perception of material reflectance. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26 (3), 77 11

VEACH, E. AND GUIBAS, L. J. (1997): Metropolis Light Transport. In *Proc. SIGGRAPH*, 65–76 13

WALKER, L. L. AND MALIK, J. (2002): When is scene recognition just texture recognition? *J Vision*, 2 (7), 255–264 21

WANG, B., YU, Y., WONG, T.-T., CHEN, C. AND XU, Y.-Q. (2010): Data-driven image color theme enhancement. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 3, 146:1–146:10 3, 18, 21

WANG, B., YU, Y. AND XU, Y.-Q. (2011): Example-based image color and tone style enhancement. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 3, 64:1–64:12 3, 18

WANG, H., O'BRIEN, J. F. AND RAMAMOORTHI, R. (2011): Data-Driven Elastic Models for Cloth: Modeling and Measurement. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 3, 71:1–71:11 3

WANG, J., TONG, X., LIN, S., PAN, M., WANG, C., BAO, H., GUO, B. AND SHUM, H.-Y. (2006): Appearance manifolds for modeling time-variant appearance of materials. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 3, 754–61 22

WANG, Z., BOVIK, A. C., SHEIKH, H. R. AND SIMONCELLI, E. P. (2004): Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 13 (4), 600–612 80

WARD, G. AND HECKBERT, P. (1992): Irradiance gradients. In *Proc. EGWR* 10, 13, 31

WARD, G., RUBINSTEIN, F. AND CLEAR, R. (1988): A ray tracing solution for diffuse interreflection. In *Proc. SIGGRAPH* Volume 22,, 85–92 13

WILKINSON, C., WOODRUFF, S. D., BROHAN, P., CLAESSON, S., FREEMAN, E., KOEK, F., LUBKER, S. J., MARZIN, C. AND WHEELER, D. (2011): Recovery of logbooks and international marine data: the RECLAIM project. *International Journal of Climatology*, 31 (7), 968–979 3

WILLIAMS, L. (1983): Pyramidal parametrics. *Proc. SIGGRAPH*, 17 (3), 1–11 15, 32, 51

WILLS, J., AGARWAL, S., KRIEGMAN, D. AND BELONGIE, S. (2009): Toward a perceptual space for gloss. *ACM Trans. Graph.* 28 (4), 103 11

WOOD, D., AZUMA, D., ALDINGER, K., CURLESS, B., DUCHAMP, T., SALESIN, D. AND STUETZLE, W. (2000): Surface light fields for 3D photography. In *Proc. SIGGRAPH* 13

XIAO, J., HAYS, J., EHINGER, K., OLIVA, A. AND TORRALBA, A. (2010): SUN database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, 3485–92 119, 120

XU, L., YAN, Q. AND JIA, J. (2013): A Sparse Control Model for Image and Video Editing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 32 (6), 197:1–197:10 19, 90

XUE, S., WANG, J., TONG, X., DAI, Q. AND GUO, B. (2008): Image-based Material Weathering. *Comp. Graph. Forum (Proc. EUROGRAPHICS)*, 27 (2), 617–626 22

YANG, Q., WANG, S. AND AHUJA, N. (2010): Real-time Specular Highlight Removal Using Bilateral Filtering. In *Proc. ECCV*, 87–100 63

YU, L.-F., YEUNG, S.-K., TANG, C.-K., TERZOPOULOS, D., CHAN, T. F. AND OSHER, S. J. (2011): Make it Home: Automatic Optimization of Furniture Arrangement. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30 (4), 1–10 3

ZHOU, S., FU, H., LIU, L., COHEN-OR, D. AND HAN, X. (2010): Parametric Reshaping of Human Bodies in Images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 29 (3), 126:1–126:10 3