EXPLORING APPEARANCE AND STYLE IN HETEROGENEOUS VISUAL CONTENT

ELENA GARCES

SUPERVISOR: DIEGO GUTIERREZ

Tesis Doctoral - Ingeniería Informática
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

September 2016

To the memory of my grandfather, Telesforo,
and my grandmother, Rosario.

ABSTRACT

There are multiple ways to capture and represent the visual world; a drawing, a photograph, or a video are a few examples of visual data that are very frequent nowadays. Despite the different nature of each domain, there is a common need to process and edit these data after its production for different purposes. For example, we might want to modify the materials and the illumination of an object in a photograph, or we might want to explore a huge collection of non labeled images. The solutions to these problems mainly depend on the amount of information we have as input: it is not the same to process a plain set of colored pixels, like a photograph, than a scene captured with a 3D laser scan and multiple cameras. Thus, the nature of the visual data will also determine the complexity of the model we can use for processing.

In this thesis, we focus on creating alternative representations of the visual content which will facilitate posterior editing and exploration tasks. In particular, we will focus on conventional visual data like pictures, video sequences , and light fields; and we will explore two different aspects or these data, the *appearance* in real scenes and the *style* in artistic scenes.

In the first part of the thesis we focus on the problem of exploring *appearance* in real scenes, represented by single images, video sequences, and light fields. We leverage the intrinsic decomposition model, which represents a scene as the product of two layers: reflectance and shading. The reflectance layer defines materials and color properties, while the shading contains illumination and geometry information. This problem is highly ill-posed as for each known value of the scene, we need to recover two unknowns. First, we present our approach to the problem for single images. Our solution is based on a two level clustering strategy, first in color space and then in image space, which allows to incorporate global and local constraints, respectively. Second, we extend the problem to the temporal domain, where the challenge is to preserve temporal consistency and keep memory consumption to a minimum. We present a solution based on an iterative workflow of reflectance propagation and completion which produces compelling results. Finally, we further extend the problem in the angular dimension, where our main goal is to keep the global coherency and leverage structural information of the light field volume to make the problem more constrained.

In the second part of the thesis we focus on representing *style* in artistic scenes, in particular those created with vector art. We devise a feature-based representation of style which is learnt via crowdsourcing from the human perception of style similarity. Thanks to this novel representation we can successfully perform operations of search by style and create mash-up compositions. Additionally, the continuous growing of online datasets, makes it necessary to develop novel tools for exploration. Thus, we propose an exploratory interface which combines information about semantic labeling of the data with the style metric to provide the user with more useful visualizations of the content of the dataset.

RESUMEN

Hay muchas formas de capturar y representar el mundo que nos rodea, por ejemplo, un dibujo, una foto, o un video, son unos pocos ejemplos de contenido visual muy frecuente actualmente. A pesar de la diferente naturaleza de cada dominio, hay una necesidad común de procesar y editar estos datos después de que son generados. Por ejemplo, podemos querer modificar los materiales y la iluminación de un objeto en una fotografía, o podemos querer explorar una gran colección de imágenes no etiquetadas. Las soluciones a estos problemas dependen principalmente de la cantidad de información que tengamos: no es lo mismo procesar el conjunto de píxeles que podemos encontrar en una fotografía, que una escena capturada con láseres 3D y múltiples cámaras. Por tanto, la naturaleza de los datos también determinará la complejidad del que modelo que podamos usar para su procesamiento.

En esta tesis nos centraremos en crear representaciones alternativas del contenido visual que facilitarán su posterior edición y exploración. En particular, no centraremos en datos que se pueden capturar con dispositivos convencionales, como puede ser dibujos, fotos, secuencias de video, y campos de luces o *light-fields*. Exploraremos dos aspectos diferentes de estos datos: la *apariencia* en escenas reales y el *estilo* en escenas artísticas.

En la primera parte de la tesis nos centraremos en explorar la apariencia en escenas reales, representadas por imágenes, secuencias de video, y light-fields. Utilizaremos el modelo de descomposición intrínseco, que representa una escena como el producto de dos capas: la reflectancia y el sombreado. La capa de reflectancia define las propiedades y color de los materiales; mientras que la capa de sombreado contiene información de iluminación y geometría. Este problema se dice que está mal formulado o *ill-posed* ya que para cada valor que conocemos de la entrada, tenemos que resolver dos incógnitas. Primero, presentamos nuestra solución tomando imágenes como entrada. Nuestra solución se basa en segmentación en dos niveles, primero en espacio de color y luego en espacio de imagen, lo cual nos permite incorporar restricciones globales y locales, respectivamente. Segundo, extendemos el problema al dominio temporal, donde el problema principal radica en mantener la consistencia temporal y el consumo de memoria al mínimo. Nuestra solución se basa en un flujo iterativo donde la capa de reflectancia es propagada desde el primer fotograma al último manteniendo la coherencia. Finalmente, extendemos el problema al domino angular, donde nuestro principal objetivo será matener la coherencia global y aprovechar la información presente en este nuevo tipo de dato, los *light-fields*.

En la segunda parte de la tesis nos centraremos en representar el *estilo* en escenas artísticas, en particular aquellas creadas con arte vectorial. Obtenemos una representación basada en características de bajo nivel que aprendemos vía *crowdsourcing* a través de nuestra percepción del estilo. Gracias a esta nueva representación podemos realizar operaciones de búsqueda por estilo y crear composiciones. Por otra parte, el continuo crecimiento de bases de datos online, hacen necesario desarrollar nuevas técnicas para su exploración. Nosotros proponemos una interfaz exploratoria que combina etiquetado semántico de los datos con nuestra métrica de estilo para proporcionar al usuario visualizaciones más útiles del contenido de estas colecciones.

ACKNOWLEDGEMENTS

This thesis would have never been possible without the help and support of many people.

Diego, for teaching me how to improve everyday as a researcher, and helping me throughout the way. For trusting my judgement and always listening my opinions.

Jorge Lopez, for passing his passion for the field on me. Because I started this path in part because him.

The people of the *Graphics and Imaging Lab*, because these years have been hard, but fun, and they made the fun part possible. I also thank them all for always sharing their knowledge and their help without expecting anything in return. In particular, I thank *Cheve*, for listening to me and supporting me from the very first moment when we started together in San Jose. *Adrian Jarabo*, for his help in the shadows. *Carlos Aliaga* and *Cris Tirado*, because they made this last period much fun with the *padel* matches and the after-beers.

The students I have supervised, *Fernando Martin*, *Daniel Osanz*, *Isabel Gaspar*, and *Manuel Lagunas*. Thank you for trusting me.

My mentors at Adobe Systems, *Sunil Hadap*, *Aaron Hertzmann*, and *Aseem Agarwala*, for accepting me in their teams, and for teaching me so many things. Particularly, Aseem, because I really learned from his pragmatism and his involvement.

All my co-authors and collaborators. Thank you for your patience and your efforts to make the projects succeed, for sharing you knowledge and your time.

I would like to thank my mother and father for always encouraging me to do what I like. For their unconditional love, patient, and support in the hardest moments. I thank my sister as well, because she always believed in me, even more than myself, no matter what.

My friends, for understanding my long silent periods because of deadlines, and always be there for a beer.

and *Carlos*. Because he has suffered the most, both the ups and downs. I thank him for always being there for me, and putting me ahead. He has helped me in all the steps of the process, emotionally and technically, and has made a better person of me.

This thesis has been funded by the Gobierno de Aragón, Adobe Systems, and the Ministerio de Economia y Competitividad (projects LIGHTSLICE and BLINK).

Contents

List of Figures

List of Tables

Part I

INTRODUCTION & OVERVIEW

1 INTRODUCTION

Capturing and representing the world has been subject of attention throughout the human history: from cave paintings to modern photography, we have found many ways to communicate our perception of the world under different degrees of realism. For example (see Figure 1.1), a stylized drawing is as an abstracted representation of the reality, where painter skills and emotions are projected onto the canvas at the time of the creation. On the contrary, a photograph usually captures a more objective reality, result of the complex interactions between light, matter and geometry that are integrated into the camera sensor.

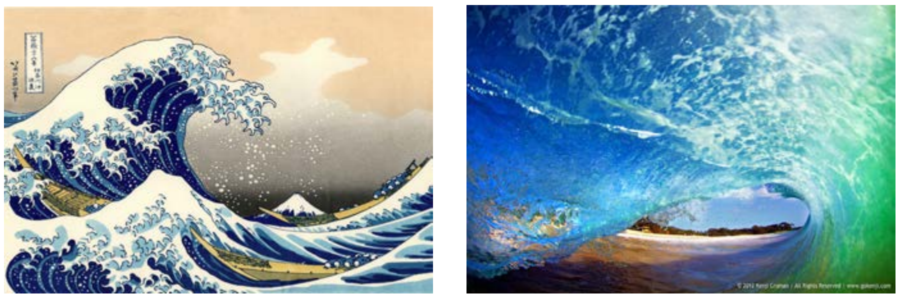


Figure 1.1: *Visual representations of a scene. Left, illustration - The Great Wave Off Kanagawa by Katsushika Hokusai. Right, photograph of sea wave by Kenji Croman©.*

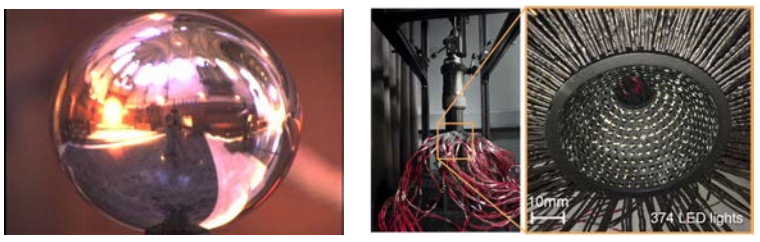
The need to process and edit these visual data after their production is constantly growing, much like the amount of available data. Thus, computer graphics and computer vision fields have been investing a lot of time and resources to develop new tools to manage any kind of visual content with a variety of purposes. For instance, image and video segmentation [45, 64], material editing [89, 158], 3D reconstruction [139], stylization [58], semantic labeling [26], or image retrieval [5] to name a few. The solutions to these problems mainly depend on the amount of information we have as input, as once the content is captured or digitized and processed on a computer, most of the real world information that was available at the instant of the capture is posteriorly missing, making the problem highly under-constrained if the input is just a single image. For example, relighting a scene [117, 116] requires knowing the materials of the objects, the position of the light sources, and the geometry. This problem commonly known as *inverse rendering* [145] can be solved, for example, recovering geometry by taking multiple images of the scene [52, 166] and capturing the illumination with a light probe [15]. However, if our input is just a single image, the task becomes extremely difficult requiring a lot of manual work [139]. Another example of a common problem that is becoming more difficult with the increasing amount of available visual data is the need to manage and process huge datasets, which are usually not labeled or classified. Thus, it would be beneficial to find automatic ways to label [95, 71] and explore this data efficiently.

In this thesis, we focus on creating alternative representations of the visual content which will facilitate posterior editing and exploration tasks. In particular, for real scenes, represented by an image, a video, or a light field, we represent their *appearance* as a combination of two components: reflectance and shading. For artistic scenes, such as illustrations, we provide a feature-based approach to *style*, which will allow us to perform style-based exploration and image retrieval. In the remainder of this chapter, we define these two aspects separately, overview the types of visual content we deal with within the context of these two aspects, and discuss the main challenges. By the end of the chapter we summarize the general contributions of this thesis.

1.1 APPEARANCE IN REAL SCENES

The appearance of an object in an image encodes fundamental information about that object and its environment. It helps to recognize real-world objects and convey information about them. For example, the illumination can tell us whether the scene is indoors or outdoors; or the reflectance properties can reveal information about the material that make up the object.

Illumination in a scene can be captured in several ways, being the most common one the use of a light probe which captures the environment illumination with a 360º HDR panoramic image [34, 35, 148] (Figure 1.2a). For instance, this set up has been used to estimate directions of multiple light sources in a scene [195, 75]. The mathematical models for representing *material* appearance can be of diverse nature- ranging from analytical functions to tables- or capture different types of materials. For example, BRDF representations are suitable for homogenous materials [136]; while BTFs or spatially-varying BRDFs are suitable for heterogeneuous appearances [32, 82]. There is not a unified representation of appearance and the success of these models is usually determined by: 1) the nature of the modeled material or illumination, and 2) the complexity of the capture system, since we usually need complex set ups with expensive equipment [128, 46, 131] (Figure 1.2b). Thus, our goal is to get rid of the equipment requirements and acquire a suitable approximation of the appearance of an objet in a scene from conventional input, like photographs, video and light fields. In such cases, the input data -plain colored pixels- is not suitable to fit complex representations of appearance so we leverage a simpler model, named *Intrinsic Scene Decomposition* which is described next.



(a) *Light probe* **(b)** *Microscopic acquisition setup [131]*

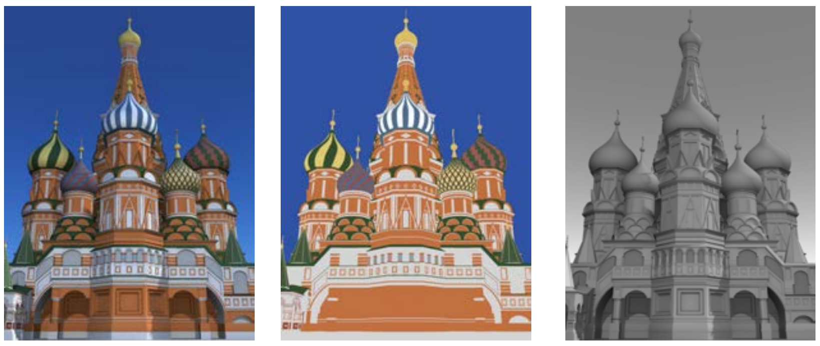
Figure 1.2: *Example of techniques used to capture illumination (a) and reflectance appearance (b).*

INTRINSIC IMAGE DECOMPOSITION Barrow and Tenenbaum [10] proposed a model that describes the appearance of an object in an image as a combination of several layers: reflectance, illumination and orientation. This choice was inspired by our ability to identify these aspects regardless of familiarity with the scene or existing illumination. They called the model *in-*

trinsic scene characteristics. Later, this problem was simplified with the name of **intrinsic image decomposition** to obtaining just the reflectance **R** and shading **S** layers from a single image **I**:

$$\mathbf{I} = \mathbf{R} \times \mathbf{S} \tag{1.1}$$

Figure 1.3 shows an example of decomposition for a synthetic scene where an object is illuminated by a white light source. Each pixel of the input image results from the combination of these two components: the reflectance, which describes the material properties and is invariant to illumination, and the shading, which contains information about the shading and shadows, and depends on the geometry and the position of the light source.



(a) *Input image* **I** **(b)** *Reflectance* **R** **(c)** *Shading* **S**

Figure 1.3: *Synthetic scene which illustrates the intrinsic decomposition. Image (a) is the result of the product of the reflectance layer (b) and the illumination in (c).*

Obtaining this decomposition is a highly ill-posed problem, as for each known value of the input image, we have two unknowns. Therefore, to make the problem tractable it is convenient to impose additional constraints. First, it is assumed that materials are lambertian, that is, materials whose visual appearance is invariant to the position of the viewer, thus ignoring effects like specularities. Second, it is assumed that images are white balanced, and illuminated with a monochromatic light source. Finally, it is not taken into account shading effects which are due to interreflections. Despite these assumptions, we will see later that the shown formulation (Equation1.1) is able to deal with a huge number of scenes present in the natural world. In this thesis we present a solution based on clustering, which relies on color chromaticity values to find regions of the image with constant reflectance.

INTRINSIC VIDEO DECOMPOSITION So far, we have only focused on the problematic of single and static images. Now, we introduce the problem of intrinsic video decomposition. With the same assumptions we presented for single images, we incorporate the temporal dimension, and the restriction of the temporal coherency. In principle, the problem is the same, since a video is a just a sequence of single images -or frames. However, naïvely applying a single image intrinsic algorithm independently to each frame yields to a temporal unstable solution, visible in the form of flickering artifacts. Thus, we seek a solution which maintains smooth transitions between the frames. In this thesis, we propose an approach based on reflectance propagation. Starting from a intrinsic decomposition of the first frame, we propagate reflectance values to subsequent frames following Bayesian probabilistic inference. At every point, the quality of the decomposition and the transitions are guaranteed by confidence intervals, and those unreliable values are completed with local intrinsic decompositions.

INTRINSIC LIGHT FIELD DECOMPOSITION Light field photography has recently become very popular with the emergence of consumer cameras (Lytro^{TM}, Raytrix^{TM}, PCam^{TM}), professionals (Lytro Cinema^{TM}) and the flourishing of virtual reality. Light field imaging outperforms traditional systems in that it captures the same scene from slightly different points of view. These variations, which are done in the angular domain, allow sophisticated post-capture effects [80], such as view-panning [107, 61], refocus [78, 134], super-resolution [118] or 3D reconstruction [90]. The problem of intrinsic light field decomposition inherits the previous problems we have found for single images and video sequences, with the additional difficulty of having to maintain angular coherency. Contrarily to video based solutions, which keep coherency in just one dimension (the temporal domain), 4D light fields need to be consistent in all angular dimensions. This makes the problem highly challenging since a global optimization is unfeasible. Our solution explores the structure and high dimensionality of the light field data to obtain better cues about reflectance and shading variations which are not available in single image or video sequences.

APPLICATIONS As we have previously discussed, the benefits of obtaining this decomposition are multiple and particular examples will be shown in detail in the corresponding chapters. In summary, having the intrinsic components is useful for any task that would require some structural knowledge about the scene. For example, once we have the intrinsic layers, changing the materials of a scene is simplified to changing the reflectance layer; segmentation is improved if the reflectance layer is used instead of the original input image; or stylization and relighting can be done more accurately by optimizing the shading and reflectance components separately.

1.2 STYLE IN ARTISTIC SCENES

The notion of style has multiple definitions depending on the context [19, 184, 39]. In paintings, the term *style* can be applied to describe common features of works which belong to a certain a period or to the particular work of an artist. Also, the concept of style can vary with culture and perception. Contrarily to *appearance* which, as we have seen before, follows physical laws, style is a subjective concept for which there is no general agreement, and thus, each domain should be studied separately. Recent works assume that the style can be identified by combining several elements that share it, and removed to obtain different levels of abstraction, e.g., for faces [14, 193], shape collections [192], buildings [132] or curves [109].

Illustration art or vector art has become quite popular in the digital domain. It is very common to find online libraries containing hundreds of thousands of pieces of vector art designed to be copied into documents or illustrations. The style of these collections ranges from simple sketches to comic-like styles with complex effects. While there is a lot of work on non photorealistic rendering to simulate styles [167, 60, 40], little attention has been paid to understand it. A very interesting categorization was made by Scott McCloud [124] in his seminal books about comics, where he proposed a triangle-based representation where images where classified among three corners: realistic, simplified and stylized. However, he did not provide a concrete definition of style as, as mentioned before, it is fundamentally difficult.

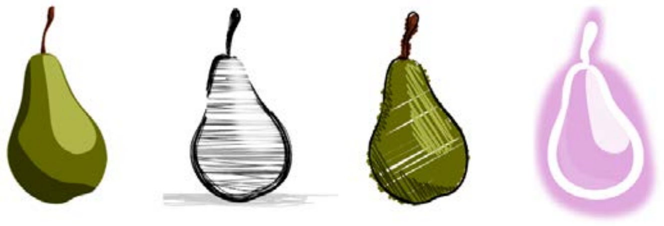


Figure 1.4: *Same object depicted in four different styles [40].*

On the other hand, the amount of data available online has promoted the creation of new tools that could handle this data from a user perspective. For example, the problem of searching particular objects within a dataset of non-labeled images used to be very difficult. However, nowadays, the use of deep neural networks [95, 169] for this task, has been a great success. A related problem is image retrieval from sketches [153], where the user provides a simple sketch of the object and the goal is to retrieve images that contain it without providing a particular label or name. In both cases, we need some sort of measure to compare between concepts and images, and to compare between images directly. Due to the high dimensionality of the domain, solutions to these problems involve machine learning techniques, where supervised approaches prevail. Thus, this kind of techniques rely on training a non-linear model with huge amounts of input-output data, where the more successful models are often uncomprehensible. So far, most of the efforts have focused on semantic labeling and search, and there are just a few works [88] which have explored the concept of *style* in this way.

In this thesis, we focus on the analysis of style in illustration; we provide a metric which measures the similarity in style between two pieces of vector art. The metric is based on pixel-wise features and is learnt from human perception of style since there is no labeled dataset which provides this kind of information. Thanks to the metric we show several applications where this metric is useful such as style-based image retrieval, mash-up generation and style-based exploration.

1.3 GOAL & OVERVIEW

The main goal of this thesis is to develop new algorithms to find suitable representations of *appearance* from real scenes and *style* from illustration. The purpose is to obtain alternative representations of the visual content which will facilitate editing operations for the case of appearance, and search and exploration for the case of style. Both problems share the inherent difficulty of dealing with raw data as is the case of pictures, images, video and light fields, and no additional sources of information were used at the time of the capture or the creation. That makes these problems particularly challenge requiring to impose external assumptions: Appearance can be defined by the physical laws that govern material and illumination interactions; while style is a subjective property for which we need to leverage human perception via crowdsourcing strategies.

Figure 1.5 provides an overview of the main structure of the thesis. This thesis is divided in two main parts, one for each aspect of study. Each type of visual content will be addressed separately per chapter, while applications

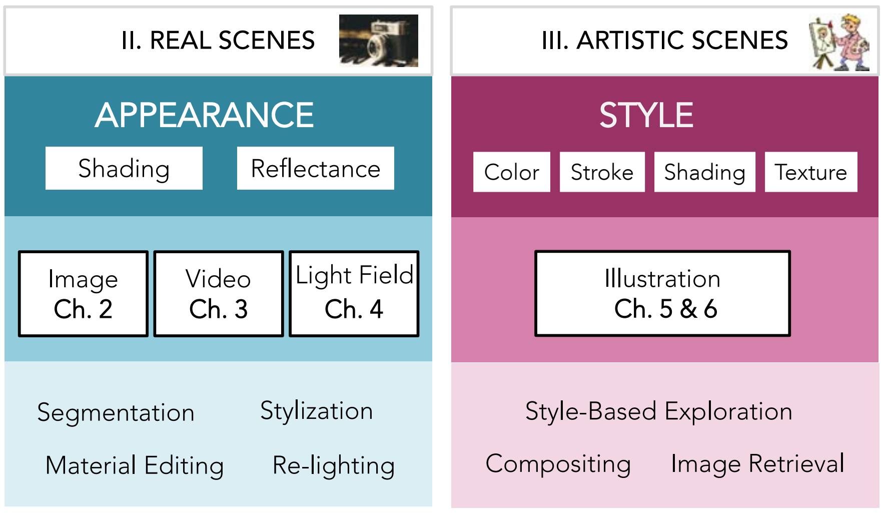


Figure 1.5: *Overview of the structure of the thesis.*

(bottom row of Figure 1.5) will be mentioned crosswise. In summary, the structure is the following:

- Part II deals with the problem of representing appearance in heterogenous data sources. In particular, we define appearance as the product of reflectance and shading, problem commonly known as *intrinsic scene decomposition*. We tackle the same problem for three types of data sources, starting from lower to higher amount of input data: images in Chapter 2, video sequences in Chapter 3 and light fields in Chapter 4. Applications that enable this decomposition like segmentation, re-texturing or relighting will be shown within the context of each domain.
- Part III tackles the problem of style in illustration data. In Chapter 5 we provide a definition of style based on pixel-based features and a similarity metric learned from humans perception. In Chapter 6, we extend the study of style to a labeled data set, and present an interface which allows style-based exploration.

This work has led to a number of publications detailed below. Of course, they have been done in collaboration with other colleagues; while my level of contribution in each can be inferred from my position in the authors list, at the beginning of each chapter I will describe and contextualize my contribution when needed.

1.4 CONTRIBUTIONS AND MEASURABLE RESULTS

1.4.1 *Publications*

Most of the work presented in this thesis has been already published, in particular in four journals indexed in JCR, including two papers in ACM Transactions on Graphics and presented at SIGGRAPH, and one peer-reviewed international conference:

- Intrinsic Images by Clustering (Chapter 2, Part II)

The main work on intrinsic image decomposition was accepted in Eurographics Symposium on Rendering (EGSR) 2012, and published in Computer Graphics Forum [57]. This journal has an impact factor of 1.542, and its position in the JCR index is 17th out of 106 (Q1) in the category Computer Science, Software Engineering (data from 2015).

Partial results were published in Ibero-American Symposium in Computer Graphics (SIACG) 2011 [56].

- Intrinsic Video and Applications (Chapter 3, Part II)

  This work was accepted at SIGGRAPH 2014, and published in ACM Transactions on Graphics [191]. This journal has an impact factor of 4.218, and its position in the JCR index is 1st out of 106 (Q1) in the category Computer Science, Software Engineering (data from 2015).

- Intrinsic Light Fields (Chapter 4, Part II)

  This work has been published as technical report in arXiv [55]

- A Similarity Measure for Illustration Style (Chapter 5, Part III)

  This work was accepted at SIGGRAPH 2014, and published in ACM Transactions on Graphics [53]. This journal has an impact factor of 4.218, and its position in the JCR index is 1st out of 106 (Q1) in the category Computer Science, Software Engineering (data from 2015).

- Style-Based Exploration of Illustration Datasets (Chapter 6, Part III)

  This work was published in Multimedia Tools and Applications 2016 [54]. This journal has an impact factor of 1.331, and its position in the JCR index is 31st out of 106 (Q2) in the category Computer Science, Software Engineering (data from 2015).

In addition to these previous publications, during my PhD I have collaborated in other research projects directly or indirectly related to the topic of this thesis:

- Icon Set Selection via Human Computation.

  In this work, lead by Lasse Laursen, we propose a method based on crowdsourcing which is able to select an optimal subset of icons according to two properties of icon design: comprehensibility and identifiability. It has been published as short paper in Pacific Graphics Conference [44], and invited to Springer Journal Computational Visual Media.

- Convolutional Sparse Coding for capturing High Speed Video Content.

  In this work, lead by Ana Serrano, we use sparse coding strategies to reconstruct a high speed video from a single shot. We guarantee smoothness in the temporal dimension by enforcing continuity in the first-order derivatives of the sparse coefficients. The initial work was published at CEIG 2014 [157]. It is currently under review at Computer Graphics Forum [156].

- Depth from a Single Image through User Interaction.

  In this work, lead by M. Angeles Lopez, we develop and algorithm which computes depth from a single image aided by human interaction. The work was presented in the XXIV Spanish Conference of Computer Graphics [115].

- Multiple Light Source Estimation from a Single Image.

  In this work, lead by Jorge Lopez-Moreno, we designed an algorithm to estimate the position of lights in a single image. It was published in Computer Graphics Forum [116], and presented in EGSR 2014.

1.4.2 *Awards*

We include here a list of awards and fellowships received throughout this thesis, that have allowed the realization of the work here presented:

- FPI Grant from the Regional Government, Diputacion General de Aragon (4-year PhD grant).
- Adobe Systems funding to extend the collaborative work after each of the research internships

Additionally, some projects described in this thesis have received different awards or recognitions:

- Our work A Similarity Measure for Illustration Style was invited to the XXIV Spanish Conference in Computer Graphics (CEIG 2014).

1.4.3 *Research Stays and Visits*

Two research stays, totaling 6 months, were carried out during this PhD in two different locations:

- June 2011 – August 2011 (three months): Research Intern at the Advanced Technology Labs at Adobe Systems (San Jose, California, USA). Supervisor: Dr. Sunil Hadap. Worked on monocular depth estimation from a single image.
- June 2013 – August 2013 (three months): Research Intern at the Creative Technology Labs at Adobe Systems (Seattle, Washingtong, USA). Supervisor: Dr. Aseem Agarwala and Aaron Hertzmann. Publication [53] result of this collaboration.

1.4.4 *Supervised Students*

During the development of this thesis I have supervised the Graduate Thesis of four students:

- Ongoing: Isabel Gaspar. Computational Icon Design. Expected graduation date: September 2016.
- Ongoing: Manuel Lagunas. Deep Learning for Art and Illustration: September 2016. Expected graduation date: September 2016.
- Daniel Osanz (Industrial Design, 2013). Design of an application for style-based image retrieval.
- Fernando Martin (Computer Engineering, 2012). Low Cost Decomposition of Direct and Global Illumination in Real Scenes.

1.4.5 *Research Projects and Industry Collaborations*

During my PhD studies I have participated in the following research projects and collaborated with the industry:

- SkinAnalytics: Since June 2016, I am external consultant of the startup SkinAnalytics, which is based in London. The aim of the company is to early predict melanoma cancer from mole pictures and historical data of the patient.
- VERVE: Vanquishing fear and apathy through e-inclusion: personalised and populated realistic virtual environments for clinical, home and mobile platforms. European Commission (FP7-ICT-2011-7). Grant no.: 288914. PI (in Spain): Diego Gutierrez
- LIGHTSLICE: Capture, analysis and applications of the multidimensional light transport (application to medical imaging). Ministerio Español de Economía y Competitividad. PI: Diego Gutierrez
- MIMESIS: Low cost techniques for the acquisition of material appearance models. Ministerio Español de Ciencia y Educación. (TIN2010-21543). PI: Diego Gutierrez

Part II

APPEARANCE IN REAL SCENES

In this part we tackle the problem of appearance capture formulated as the decomposition of a scene into its intrinsic components. We start presenting the problem for single images, and describing our solution based on color clustering. Then, we extend the problem to video sequences, including the temporal dimension in the formulation and present our solution based on reflectance propagation. Finally, we further extend the problem to light field volumes, where the total number of dimension is four, due to the inclusion of the angular dimension.

2 INTRINSIC IMAGE DECOMPOSITION

In this chapter we present the problem of intrinsic image decomposition, which is defined as the separation of an image into its intrinsic shading and reflectance components. We present a novel algorithm that requires no user strokes and works on a single image. Based on simple assumptions about its reflectance and luminance, we first find clusters of similar reflectance in the image, and build a linear system describing the connections and relations between them. Our assumptions are less restrictive than widely-adopted Retinex-based approaches, and can be further relaxed in conflicting situations. The resulting system is robust even in the presence of areas where our assumptions do not hold. We show a wide variety of results, including natural images, objects from the MIT dataset and texture images, along with several applications, proving the versatility of our method.

This work is published in *Computer Graphics Forum* and presented at *Eurographics Symposium on Rendering (EGSR) 2012*. Preliminary results about reflectance clustering were presented at the Ibero-American Symposium in Computer Graphics (SIACG) 2011.

E. Garces, D. Gutierrez & J. Lopez-Moreno
Graph-Based Reflectance Segmentation
SIACG 2011

E. Garces, A. Munoz, J. Lopez-Moreno & D. Gutierrez
Intrinsic Images by Clustering
Computer Graphics Forum, Vol.31 (4), EGSR 2012

2.1 INTRODUCTION

The problem of separating an input image into its intrinsic shading and reflectance components [10] is extremely ill-posed. However, many applications would benefit from the disambiguation of a pixel value into illumination and albedo, such as image relighting or material editing. This problem is usually formulated as the input image I being a per-pixel product of its unknown intrinsic shading S and reflectance R, so the space of mathematically valid solutions is in fact infinite. Existing methods therefore need to rely on additional sources of information, such as making reasonable assumptions about the characteristics of the intrinsic components, having multiple images under different illuminations or asking the user to add image-specific input.

In this chapter, we describe a new algorithm that works on a single off-the-shelf image and requires no user strokes. We do make some reasonable assumptions, in the form of flexible constraints. We formulate the decomposition of an input image into its shading and reflectance components as a linear system that exploits relations between clusters of similar reflectance. Classic Retinex approaches assume that i) reflectance is piecewise constant, and ii) shading is spatially smooth (C^0 and C^1 continuity) [102, 74]. Based on this, a number of authors have proposed different approaches [51, 91, 163, 59]. In this work we first find clusters of similar reflectance in the image following the observation that *changes* in chromaticity usually correspond to changes in reflectance.

We then relax the second Retinex assumption that shading is spatially smooth in two ways: we assume only C^0 continuity on the shading, and only *at the boundaries* between clusters (as opposed to the whole image), and describe this as a set of linear equations. Our linear system is completed by additionally preserving reflectance between clusters even if they are not contiguous, and adding a regularization term to make it more stable.

Our main contribution is a novel algorithm for intrinsic images decomposition which deals with a wider range of scenarios than traditional Retinex-based algorithms, yields better decompositions than existing automatic methods from single images, and offers an attractive trade-off between quality and ease of use, compared with techniques requiring either significant user input or multiple input images. We present an exhaustive comparison against most existing techniques, which we are public along to our source code. Last, we show compelling example applications of retexturing, relighting and material editing based on our results.

Like all existing methods that deal with this ill-posed problem, our work is not free of limitations: Our C^0 assumption is a simplification that breaks for some occlusion boundaries and sharp edges, which translate into inaccurate equations in the system. However, given our robust formulation which usually translates into a few thousand equations, these inaccurate equations represent a very small percentage, and our method generally handles these situations well.

2.2 RELATED WORK

AUTOMATIC Some automatic methods rely on reasonable assumptions about the nature of these two terms, or the correlation between different characteristics of the image. Horn [74] presents a method to obtain lightness from black and white images, using pixel intensity information and assuming that lightness corresponds to reflectance. He further assumes that the reflectance remains locally constant while illumination varies smoothly (as described by the Retinex theory [102]). Funt et al. [51] extend this approach to color images, and propose the analysis of chromaticity variations in order to identify the boundaries of different reflectance areas. They enforce integrability of the shading at these boundaries and propagate their values to their neighboring pixels by diffusion, solving the subsequent Poisson equation with a Fourier transformation. This was later extended by Shen et al. [163] with global texture constraints, forcing distant pixels with the same texture to have the same reflectance. This constraint greatly improves the performance of the standard Retinex method, although it relies on objects with repeated texture patterns and may yield posterization artifacts due to the wrong clustering of distant pixels. The related method by Finlayson and colleagues [62] is mainly oriented to remove shadows by minimizing entropy, but does not to recover intrinsic images. The work by Jiang et al. [83] assumes that correlated changes in mean luminance and luminance amplitude indicate illumination changes. By introducing a novel feature, *local luminance amplitude*, the authors obtain good results, although limited to images of relatively flat surfaces and objects from the recently published MIT dataset for intrinsic image decomposition [63]. This actually simplifies the problem since such objects are treated in isolation, avoiding the problem of occlusion boundaries at the outlines. Recently, Gehler and colleagues[59] proposed a probabilistic model, based on a gradient consistency term and a reflectance prior, which assumes that reflectance values belong to a sparse set of basis colors. The problem is formulated as an optimization of the proposed energy function. The method yields good results, although again limited to isolated objects from the MIT dataset. Our linear system formulation allows for much faster computational times (up to a thousand times faster), and generalizes well over a wider range of images (including both natural and texture images).

USER INTERVENTION Another set of techniques rely on assumptions *and* user intervention. Bousseau and colleagues [22] simplify the problem by assuming that local reflectance variations lie in a 2D plane in RGB space not containing black, which may not be compatible with certain texture or grayscale images. This assumption is also used in the work by Shen and Yeo [164], who further consider that neighboring pixels in a local window with similar intensity have also similar reflectance. In addition, Bousseau's method requires that the user define constraints over the image by means of three different types of strokes: constant-reflectance, constant-illumination and fixed-illumination. Their method produces very compelling results, although creating the appropriate strokes for each particular image (from 15 to 81 for the figures in the paper) may be far from intuitive for unskilled users. The same set of user tools is employed in the recent work by Shen and colleagues [162], who use an optimization approach that further assumes that neighboring pixels with similar intensity have similar reflectance values. In the context of material editing, Dong et al. [37] assume input images of globally flat surfaces with small normal perturbations and lit with a directional light, and require user strokes for optimal decompositions. In contrast, our method is almost fully automatic (usually a single parameter is needed) and requires no user strokes.

MULTIPLE IMAGES Last, another strategy consists of incorporating additional information, either from other sources or from multiple images. Tappen et al. [173] classify the derivatives of the image as produced either by illumination or albedo. Ten classifiers are obtained from training Adaboost [47] with a set of computer generated images containing only reflectance or illumination components. They further refine their approach by introducing a new training set of real-world images and including a method to weigh the response to these classifiers [174]. Despite these advanced techniques, several configurations of illumination and reflectance remain very difficult to decompose and additional techniques like Markov Random Fields (MRF) and Belief Propagation (BP) are necessary in order to yield good solutions. Weiss [182] uses a large sequence of images of the same scene (up to 64 images, and no less than 35, taken in controlled settings), where the reflectance remains constant and illumination varies in time. Also using multiple images, Laffont and colleagues [99] leverage multi view stereo techniques to approximately reconstruct a point cloud representation of the scene. After some user intervention, illumination information computed on that point cloud is propagated in the image. Their method decomposes the illumination layer into three components: sun, sky and indirect light. Last, the concept of *intrinsic colorization* is introduced by Liu et al. [113]; to colorize a grayscale image, their method recovers the needed reflectance component from multiple images obtained from the web, in order to transfer color from there. All these techniques require multiple images

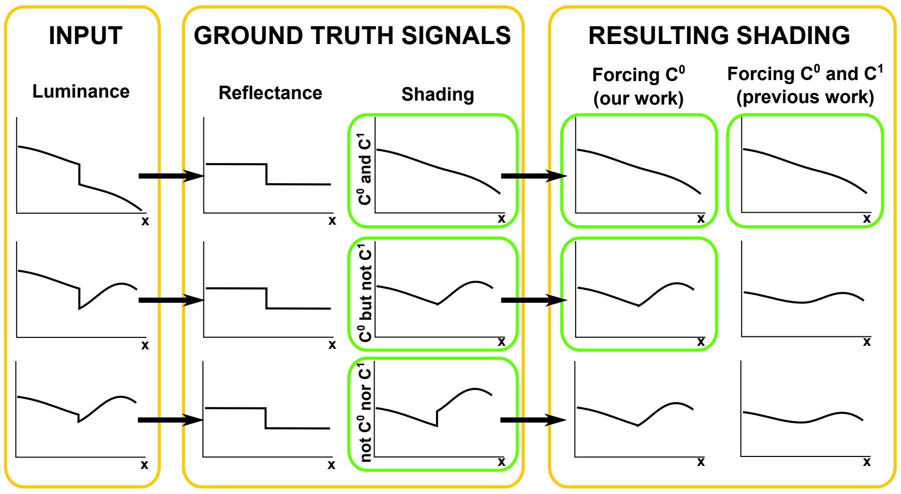


Figure 2.1: *Intrinsic shading estimation when both the shading and the reflectance present a discontinuity at the same point (as in some occlusion boundaries). Left column: Three different input luminance signals. Middle columns: Ground truth intrinsic signals. All three input signals are the result of multiplying the same reflectance with three different shading signals, presenting different continuity characteristics. Right columns: Results assuming both C^0 and C^1 continuity on the shading, compared to C^0 only (our method). Notice how our algorithm leads to an accurate result in two of the three cases, while yielding less error in the most unfavorable case.*

as input, sometimes captured under controlled settings, while our approach simply takes an off-the-shelf single image.

2.3 ALGORITHM

The desired decomposition consists of separating an image into two components (images): one representing reflectance information, and another containing the illumination or shading. We use RAW or linearized RGB values as input. For a Lambertian scene, the problem can be simply formulated as:

$$I(x, y) = S(x, y) * R(x, y) \tag{2.1}$$

where $I(x, y)$ is the input image, $S(x, y)$ is the shading image, $R(x, y)$ represents reflectance information and $*$ is a per-channel Hadamard product. Our goal is to obtain $S(x, y)$ and $R(x, y)$, given $I(x, y)$. We make the problem tractable with a few assumptions well-grounded on existing vision and image processing techniques. While of course our assumptions may not always be accurate throughout the whole image, they allow us to devise a method that works very well on a large range of images while keeping our algorithm simpler than other approaches.

ASSUMPTIONS Horn made the key observation that, for grayscale images, sharp reflectance changes cause intensity discontinuities in the luminance of an image [74]. Our first assumption relies on the later generalization to color images by Funt et al. [51], who associate changes in reflectance with changes in chromaticity. We first leverage this correlation between reflectance and chromaticity values by detecting regions of similar chromaticity in the input image, which are assumed to approximate regions of similar reflectance. We

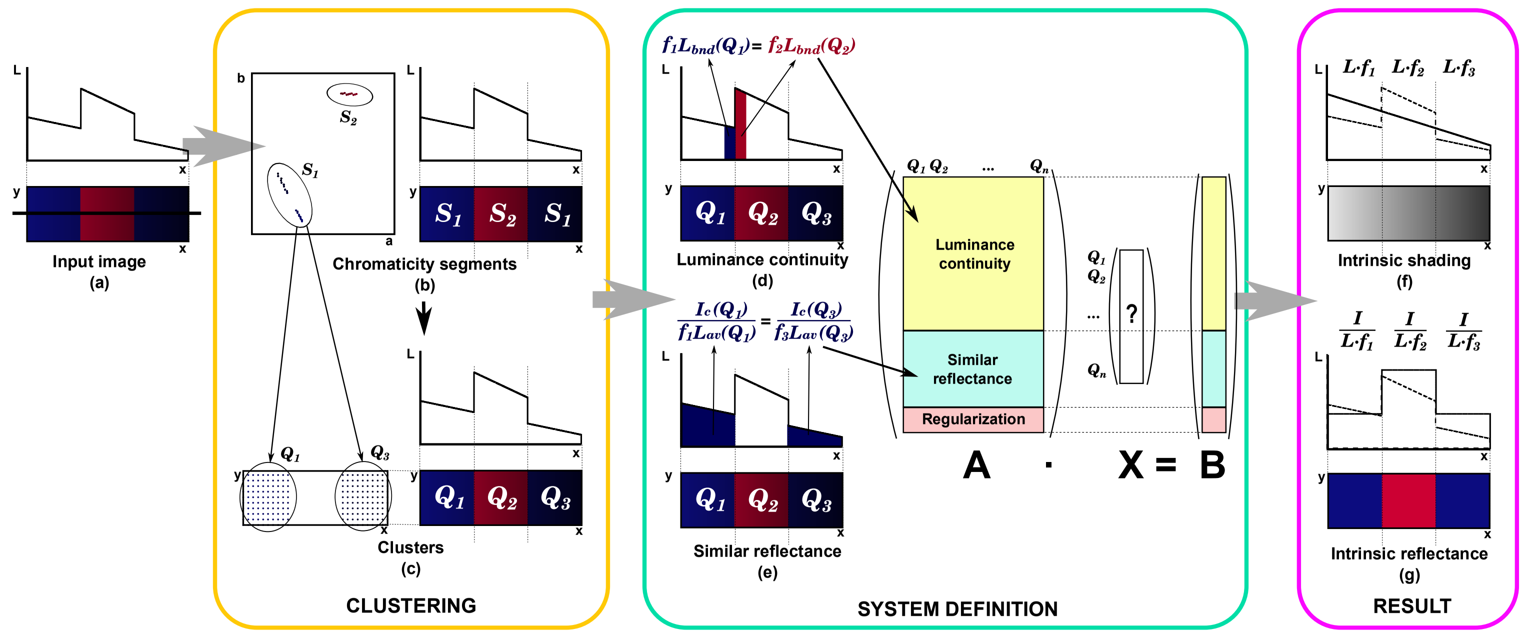


Figure 2.2: *Overview of the algorithm for the simple case of three colored patches and a continuous shading gradient.* ***(a)*** *Input image, with a plot of the pixels luminance along a scan line.* ***(b)*** *Initial k-means segmentation. Left: A scatter plot of the (a, b) coordinates (Lab color space) shows two segments of different chrominance (S_1 and S_2). Right: These segments belong to different parts of the image, with S_1 split in two image areas (labeled accordingly in the figure).* ***(c)*** *Subsequent clustering. Left: Segments are further divided into clusters of contiguous pixels. The example shows S_1 being clustered into Q_1 and Q_3 in image space. Right: clusters labeled in the image.* ***(d)*** *Enforcing luminance continuity on the boundaries between two clusters yields a large number of equations for the linear system.* ***(e)*** *Clusters originally belonging to the same segment maintain similar reflectance properties (the example shows Q_1 and Q_3, both belonging initially to S_1). This yields another set of equations. The final system is completed with a regularization term.* ***(f)*** *Result: Intrinsic shading. It is a continuous signal, as described by the equations in (d).* ***(g)*** *Result: Intrinsic reflectance. Q_1 and Q_3 share the same reflectance, as described by the equations in (e). Please refer to the text for further details on the equations and their notation.*

implement this as a soft constraint, though, which we relax in specific cases (see Sections 2.3.1 and 2.3.2).

Furthermore, existing Retinex-based techniques (see for instance [51, 91, 163]) assume that shading is a smooth function, therefore being both C^0 and C^1 continuous. However, there are a number of particular cases (such as some occlusion boundaries) in which this assumption does not hold. Our second assumption relaxes this restriction by imposing only C^0 continuity *at boundaries* between the regions previously detected. This allows us to handle a wider variety of cases correctly; in cases where the smooth shading assumption does hold, our method naturally maintains C^1 continuity as well (see Figure 2.1). In cases where this assumption breaks, our method still provides a more accurate reconstruction of the intrinsic signals. Last, as previous works, we assume a white light source and a correctly white balanced input image.

OVERVIEW Figure 2.2 shows an overview of our algorithm, applied to patches of different colors with a continuous shading gradient. It works in two steps, which we term *clustering* and *system definition*. First, we segment the input image according to chromaticity. We then subdivide the resulting segments, and obtain a set of clusters of connected pixels with similar chromaticity. This clustering is then refined to better approximate reflectance (as opposed to chromaticity) discontinuities, according to our first assumption.

Based on this clustering, we then build a linear system of equations defining the connections and relations between the different clusters, as well as the different constraints. One set of equations describes the C^0 continuity in the shading at cluster boundaries (our second assumption). We then make the observation that all clusters originally coming from the same segment should in principle maintain similar reflectance, even if they are not contiguous. This is similar to the observation made by Shen et al. [163]; however, we improve this in two important ways: first, we do not need to rely on texture information; second, we work at cluster level, as opposed to pixel level, which translates into a more stable solution. This yields our second set of equations. The system is completed with an additional regularization term. By solving the resulting linear system we obtain the intrinsic shading image; reflectance is obtained by means of a simple per-pixel RGB division (Equation 4.1), as previous works [22]. The next sections describe these steps in detail.

2.3.1 *Clustering*

We aim to divide the image into clusters of similar chrominance properties. Given our assumptions, the boundaries between those clusters will indicate reflectance discontinuities. This is a reasonable task, given the reduced set of reflectance values in natural images [141]. This reduced set was also leveraged in recent work by Bousseau et al. [22], who further assumed that reflectance colors lie in a 2D plane not intersecting the origin. Several existing segmentation techniques, such as Mean Shift, the graph-based method by Felzenszwalb and Huttenlocher [45] or its subsequent modification [56] have been thoroughly tested, but unfortunately none would yield satisfying results for our purposes. We thus have designed a novel two-step clustering strategy, specially tailored for the problem of intrinsic images decomposition. For the sake of clarity, we refer to the first step as *segmentation*, and to the second as *clustering*.

SEGMENTATION We first segment the image according to chromaticity values, regardless of the spatial location of the pixels. We define our segmentation feature space as $\mathcal{F} = \{\beta, a, b\}$ where (a, b) are the chromatic coordinates of the input image in *CIELab* space, and β is a feature defined to handle strong blacks or whites (these are defined as pixels with very low chromaticity values and very low or high luminance). These values would be difficult to segment properly in a chromaticity-based algorithm, and usually describe important reflectance features. For each pixel in the image, we define β as:

$$\beta = \begin{cases} -\mu & \text{if } (|a| < \lambda)\ \&\ (|b| < \lambda)\ \&\ (L < L_{min}) \\ +\mu & \text{if } (|a| < \lambda)\ \&\ (|b| < \lambda)\ \&\ (L > L_{max}) \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where $\mu = 10^5$, $\lambda = 0.20 \max(|a|, |b|)$, $L_{min} = 0.15 \max(L)$ and $L_{max} = 0.95 \max(L)$. For this initial segmentation, we use the k-means implementation from Kanungo et al. [87]. Gehler and colleagues [59] also used k-means for their global sparse reflectance prior, which along with their shading prior and their gradient consistency term, fit into their global optimization system. In contrast, we use this segmentation to drive a simple and efficient system of linear equations. Note that the high μ value in the definition of β in equation 2.2 effectively forces the algorithm to create different segments with only strong black (or white) pixels. Except otherwise noted, we set $k = 10$ as the number of segments, but in our implementation it is left as a user parameter. The result of this step is a set of segments $\mathcal{S} = \{S_i\}$ (see Figures 2.3.a and 2.3.b). These will guide the clustering step of the process, and help define global reflectance constraints between disconnected areas of the image during the *system definition* stage of the algorithm (Section 2.3.2).

CLUSTERING The previous segmentation defines global relations between (possibly disconnected) regions of the image. We now take into account local constraints by considering spatial contiguity. We first subdivide each segment $S_i \in \mathcal{S}$ into a set of *clusters* of contiguous pixels (8-neighborhood in image space), obtaining $\mathcal{Q}^o = \{Q_i^o\}$ (Figure 2.3.c). This set $\mathcal{Q}^o$ may contain very small clusters (due to quantization, aliasing or smooth varying textures), which could potentially later make our system less stable, or pairs of connected clusters where changes in chromaticity do not correspond to changes in reflectance (maybe due to shadows [62]).

Merging small clusters: Given a cluster Q_r^o containing less than p pixels, we locate its neighbor cluster Q_s^o with the closest average chrominance and merge them together: $Q_s^{o*} = Q_r^o \cup Q_s^o$. For the results in this work we use $p = 10$. This process is iterated until no more small clusters remain (Figure 2.3.d).

Merging smooth boundaries: Since chrominance and reflectance are not always exactly related, the k-means algorithm might yield over-segmented results. Given two adjacent clusters Q_r^{o*} and Q_s^{o*}, we average RGB pixel differences across the common border, obtaining a scalar d. The clusters are merged into Q_{rs} if $d < D$. The threshold D is set to 0.01 times the maximum pixel value in the image.

The result after these operations is our final cluster set $\mathcal{Q} = \{Q_i\}$ (see Figure 2.3.e).

2.3.2 *System definition*

The previous step has yielded a set of clusters separated by reflectance discontinuities. We now describe how to estimate the intrinsic shading from this initial clustering. We define a per-cluster factor f_i that, multiplying the luminance of the pixels of the cluster, will result into the intrinsic shading:

$$S(x,y) = f_i L(x,y) \tag{2.3}$$

where $(x,y) \in Q_i$. Instead of using expensive optimization techniques, we build a linear system of equations where f_i are the unknowns of our system. This system is built from three sets of equations as described below.

LUMINANCE CONTINUITY We first enforce C^0 luminance continuity at the boundaries between clusters, in effect assigning abrupt changes at such boundaries to reflectance variations. Given the boundary between two clusters Q_r and Q_s:

$$f_r\, L_{bnd}(Q_r) - f_s\, L_{bnd}(Q_s) = 0 \tag{2.4}$$

where $L_{bnd}(Q_r)$ represents the luminance of the pixels in cluster Q_r at the boundary with cluster Q_s (and vice versa for $L_{bnd}(Q_s)$, see Figure 2.2.d). Last, f_r and f_s are the unknowns that force luminance continuity. In practice, we make Equation 2.4 more robust and obtain $L_{bnd}(\cdot)$ by averaging the luminance values of several pixels in a small window to each side of the boundary. We set the width of this window to three pixels for all the images in the chapter.

However, applying exactly Equation 2.4 leads to an unstable behavior of the linear system; instead, we rewrite it in log-space:

$$\ln(f_r) - \ln(f_s) = \ln\left(\frac{L_{bnd}(Q_s)}{L_{bnd}(Q_r)}\right) \tag{2.5}$$

which leads to a more stable system and avoids both the trivial solution $f_i = 0$ and solutions with any $f_i < 0$. We apply Equation 2.5 to each pair of contiguous clusters.

CLUSTERS OF SIMILAR REFLECTANCE All clusters in $\mathcal{Q}$ coming from the same segment $S_i \in \mathcal{S}$ should in principle maintain similar reflectance. For each pair of clusters $\{Q_r, Q_s\} \in S_i$ we then have one equation per-channel with $c = \{R, G, B\}$:

$$\frac{I_c(Q_r)}{f_r\, L_{av}(Q_r)} = \frac{I_c(Q_s)}{f_s\, L_{av}(Q_s)} \tag{2.6}$$

where $I_c(r)$ is pixel average of the input image for all the pixels of the cluster Q_r and $L_{av}(r)$ is the average luminance of cluster Q_r (with an analogous definition for Q_s). We again reformulate this in log-space:

$$\ln(f_s) - \ln(f_r) = \ln\left(\frac{I_c(Q_s)L_{av}(Q_r)}{I_c(Q_r)L_{av}(Q_s)}\right) \tag{2.7}$$

However, clusters of the same chromaticity may actually have different reflectance, in which case the corresponding equations should not be included in the system. We adopt a conservative approach, and turn to the L coordinate to distinguish between different reflectances (e.g. light red and dark red). We define a threshold T_L of 5% of the maximum luminance of the image, and apply Equation 2.7 across clusters only if $|L_{av}(Q_r) - L_{av}(Q_s)| < T_L$.

LUMINANCE REGULARIZATION Last, we add a regularization equation to the system as follows:

$$\sum_i \ln f_i = 0 \tag{2.8}$$

With these three steps, we have posed our problem of obtaining intrinsic images as a system $\mathbf{AX} = \mathbf{B}$, where the number of equations equals the number of cluster boundaries plus the reflectance similarity equations plus the regularization equation. The elements of the unknown vector $\mathbf{X}$ are $x_i = \ln(f_i)$. In order to ensure the numerical stability of the solution, we solve the equivalent system $\mathbf{A^T A X} = \mathbf{A^T B}$ by means of a Quasi-Minimal Residual method (QMR) [130]. We found that using a standard Jacobi pre-conditioner yields good results in our context. From the solution vector $\mathbf{X}$ we can trivially obtain the final $f_i = \exp^{x_i}$ for each cluster Q_i. Last, we account for the fact that our algorithm assigns each pixel to a given cluster (thus creating hard borders between clusters), while in contrast all the input images present some blur at the edges due to the imaging process; we therefore apply a 5×5 median filter at the cluster boundaries to obtain the final shading image.

2.4 RESULTS AND DISCUSSION

Most of the results shown in this work have been generated from 16-bit RAW images, although in general our algorithm works well on linearized, 8-bit images. Larger versions of the results, along with detailed clustering and scatter plot data, are included in the Appendix 2.A. Using our unoptimized code, our algorithm works at interactive rates e.g. an average image such as *clown* (see Figure 2.11), which results in 834 clusters, takes around 5 seconds on an Intel Core i5-2500 CPU at 3.30 GHz. In our tests, the most complex images may have up to 3000 clusters, resulting in about 15 seconds of processing time. Figure 2.4 shows how our algorithm successfully deals with the varied geometrical and texture complexities of several challenging web images, producing satisfying results (more examples can be found in the Appendix 2.A). *Dragon* in the last row illustrates how 8-bit images may present pixel values close to zero, which cause numerical instabilities and are hard to disambiguate for any intrinsic images decomposition algorithm. This problem is greatly ameliorated with 16-bit images.

In Figures 2.9, 2.10, 2.11 and 2.12 we include an exhaustive comparison against state-of-the-art methods. We have tried to gather together all the published results common to most methods, but not all methods report results on all images. Compared to other automatic, single-image approaches, Tappen et al.'s technique [173] shows perceivable reflectance artifacts in the shading image. Shen et al.'s method [163] suffers from posterization artifacts in the recovered reflectance (see for instance the *doll* image or the sky in *St. Basil*); additionally, almost all the shading images show severe continuity artifacts.

Shen and Yeo [164] provide a somewhat limited selection of images in their paper, both of which appear in our figure. It can be seen how in *baby*, the shading shows inconsistencies such as the exaggerated contrast between the legs and floor, while the eyes have been wrongly assigned to the shading layer.

The method of Gehler et al. [59], although producing reasonable results for the MIT dataset, tends to retain reflectance in the shading layer for natural images (see for instance *baby* image in Figure 2.10 or *synthetic doll* in

Figure 2.12). Moreover, their optimization function is computationally expensive, taking several hours, while our technique takes seconds to finish.

Last, the *doll* result using Weiss's approach [182], despite using 40 images taken under carefully controlled settings, shows clear shading residuals in the reflectance layer, and wrong gradients in the shading image.

Our work yields results on-par with the user-assisted techniques by Bousseau et al. [22] and Shen et al. [162], without requiring any user strokes. In *St. Basil*, our solution shows some artifacts especially visible in the black areas assigned to reflectance. Bousseau's result is free from such artifacts, although at the expense of requiring 81 user strokes, divided in three different kinds. Although the authors show that their method is robust to small perturbations applied to such strokes, placing them from scratch is probably challenging for unskilled users. In contrast, our automatic method does a better job at assigning the doll's eyelashes to the reflectance layer Furthermore, Bousseau's method is not well fitted for texture images with rich reflectance variations, as recently demonstrated by Dong and colleagues [37]. Our approach is free from such restriction, and it handles those cases well (see Figures 2.5 and 2.6).

APPLICATIONS Accurate decomposition into intrinsic images can play an important role in a broad range of applications such as material editing or image relighting. Figure 3.13 shows some applications using our intrinsic decomposition. For re-texturing we modify the reflectance layer and use the same normal-from-shading approach to deform the new textures. The relighting example, in this case, uses an image-based relighting algorithm [117] that modifies the shading image before multiplying it by a sepia-shifted reflectance. Last, the wheel on the right shows another example of global reflectance manipulation. Furthermore, our technique can also be used in conjunction with others, such as the image-based material modeling technique recently presented by Dong et al. [37].

MIT DATASET We have also tested our method using images from the MIT image dataset provided by Grosse et al. [63]. This dataset is designed to test intrinsic image decomposition methods, providing ground truth images for a variety of real-world objects. Figure 2.5 shows some examples of our results, compared against Color-Retinex, the combination of Weiss [182] and Retinex and the recent approach by Dong et al. [37]. Note that Weiss's technique requires more than 30 input images, while the algorithm by Dong et al. requires user strokes. As we can observe, our algorithm obtains near optimal results in these cases. The full set of results can be seen in Figures 2.5 and 2.6.

Other MIT images are much more ill-posed for any intrinsic images algorithm, presenting extremely complex combinations of shading, smooth varying textures and poor chromaticity. Figure 2.7 shows our results for the cases of *squirrel* and *deer*. On the top rows, we show the original image along with our recovered shading and reflectance. Although our method yields reasonable results, some artifacts are clearly visible. This is mainly due to the extremely poor chromaticity variation of the original images, which hampers our segmentation step. The bottom row shows scatter plots of *squirrel* and *deer*, along with *St. Basil* and *raccoon* for comparison purposes. Note how, despite its overall lack of chromaticity, *raccoon* presents clear distinct areas in the (a,b) plane, due to the scribbles painted on its surface. To the best of our knowledge, there is no published method that can successfully deal with images such as *squirrel* or *deer*.

OCCLUSION BOUNDARIES Our method lets us handle a wide variety of images even in the presence of occlusion boundaries or sharp edges, where our assumptions do not hold. This is due to our robust linear system formulation: The number of equations describing C^0 continuity at an offending occlusion boundary is usually very small compared with the total number of equations in the system, which diminishes their influence in the result. For instance, the system solved for *St. Basil* is made up of 3557 equations, from which only 267 belong to occlusion boundaries (a mere 7%). In all the images shown in this work, we only found one case (*doll*) where the percentage of offending equations was higher than 10% (48 over 282 equations, 17%). This image presents a unique combination of reflectance distributions that makes it especially challenging for our algorithm. It shows a very uniform background (which translates into very few clusters adding equations to the system), whereas the doll itself has lots of patches of different reflectance in contact with such background, due to the striped pattern of its clothes (adding a relatively high number of occlusion boundary equations). Note how, in contrast, *St. Basil* presents most of the colorful patches inside the building itself (few equations describing occlusion boundaries with the sky). Even though the automatic results provided by our system may be considered satisfactory, we can improve them by simply creating a matte of the doll, which can be easily done with existing image editing tools, and then solve the two resulting images (doll and background) as separate problems. Figure 2.12, bottom row, shows the result of directly applying our method, and the improved results with our quick matting profile created in *Photoshop*©. Artifacts due to the inherent difficulty of this image can be seen across all methods.

2.5 CONCLUSIONS AND FUTURE WORK

Decomposing an image into its intrinsic components is still an open problem with multiple potential applications. Automatic methods such as ours need to rely on reasonable assumptions or additional sources of information. On the other hand, existing user-assisted methods remain challenging for the average unskilled user, given the difficulty in telling apart the confounding factors of reflectance and shading in some situations. Our problem formulation is less restrictive than traditional Retinex-based methods, and allows us to relax our initial assumptions in certain cases. We have shown a wide variety of results, not only on natural scenes, but on the MIT dataset and even texture images as well. Additionally, we have also provided a thorough comparison with previous approaches. Our algorithm produces better results than other automatic techniques on a broad range of input images, and on-par compared to user-assisted methods, but without the challenging task of providing the right strokes.

A potential line of future work would be to use our results as input to a simplified user interface, where it would be simpler to fix remaining artifacts. Also, our work could inspire and benefit from further research on segmentation methods. In conclusion, we believe that our approach offers an attractive trade-off between accuracy of the results, ease of use and efficiency.

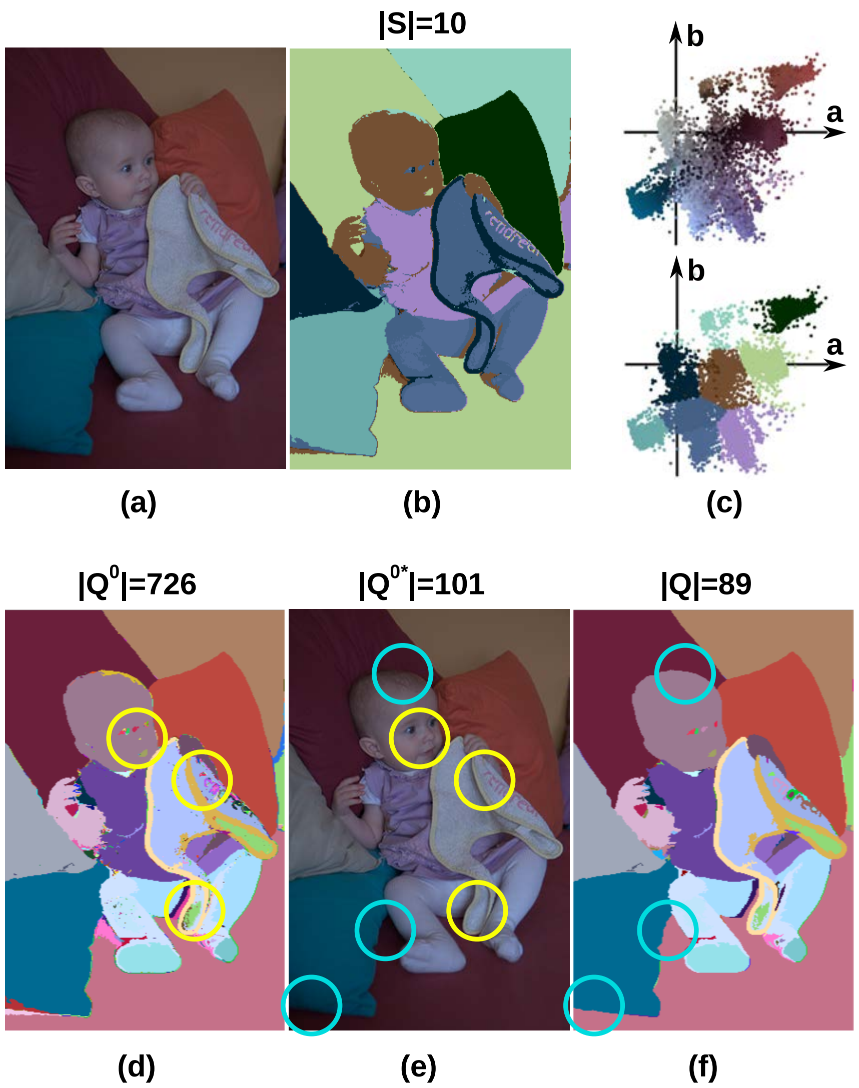


Figure 2.3: *Our segmentation-clustering process. (a) Input image (b) Result of the first segmentation step, yielding 10 distinct segments in $\mathcal{S}$. (c) Top, scatter plot of the input image in the (a,b) plane. Bottom, scatter plot of the first segmentation step. (d) Initial clustering (726 clusters in $\mathcal{Q}^0$). (e) Merging small clusters. (f) Final cluster set $\mathcal{Q}$ after merging smooth boundaries (89 clusters). The yellow and blue circles highlight areas were the effects of these last two steps are clearly visible. Notice how noise is eliminated (yellow circles), as well as smooth gradients due to shadows (blue circles).*

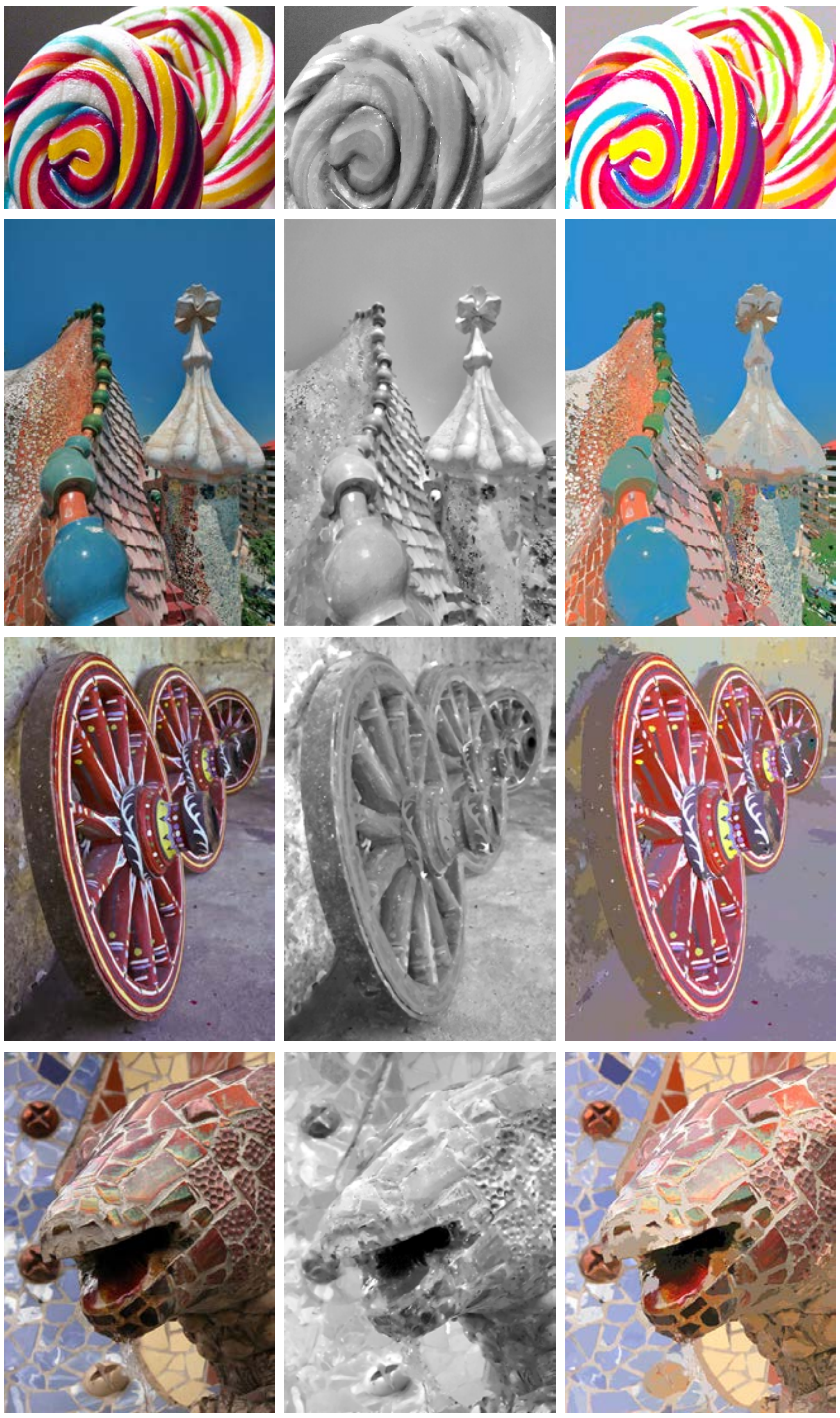


Figure 2.4: *Intrinsic images obtained with our method (using 8-bit input images). Left column: Input image. Middle column: Intrinsic shading. Right column: Intrinsic reflectance. First row: lollipop,* k = 10 *(original image by Thalita Carvalho, flickr.com). Second row: Batlló house,* k = 12 *(original image by lukasz dzierzanowski, flickr.com). Third row: wheels,* k = 16 *(original image by Angela Smith Kirkman). Last row: dragon,* k = 22 *(original image by Jordanhill School D&T Dept, flickr.com)*

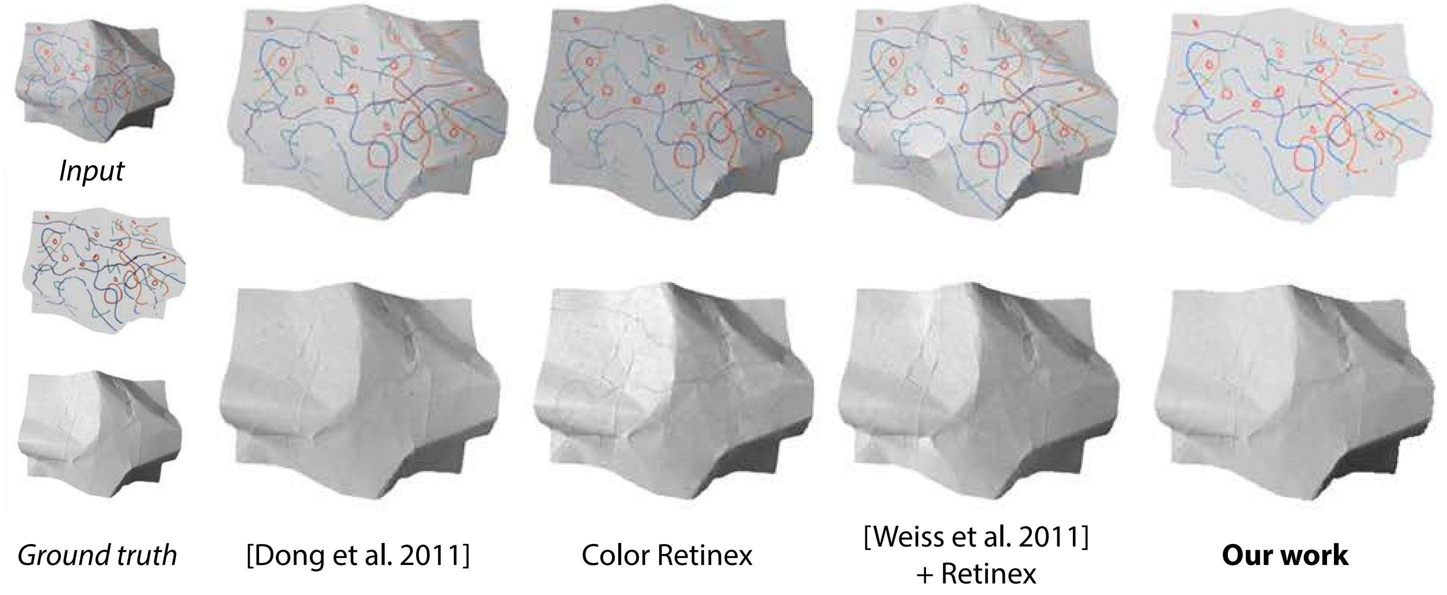


Figure 2.5: *Results using the MIT image dataset provided by Grosse et al. [63] and comparisons with previous works.*

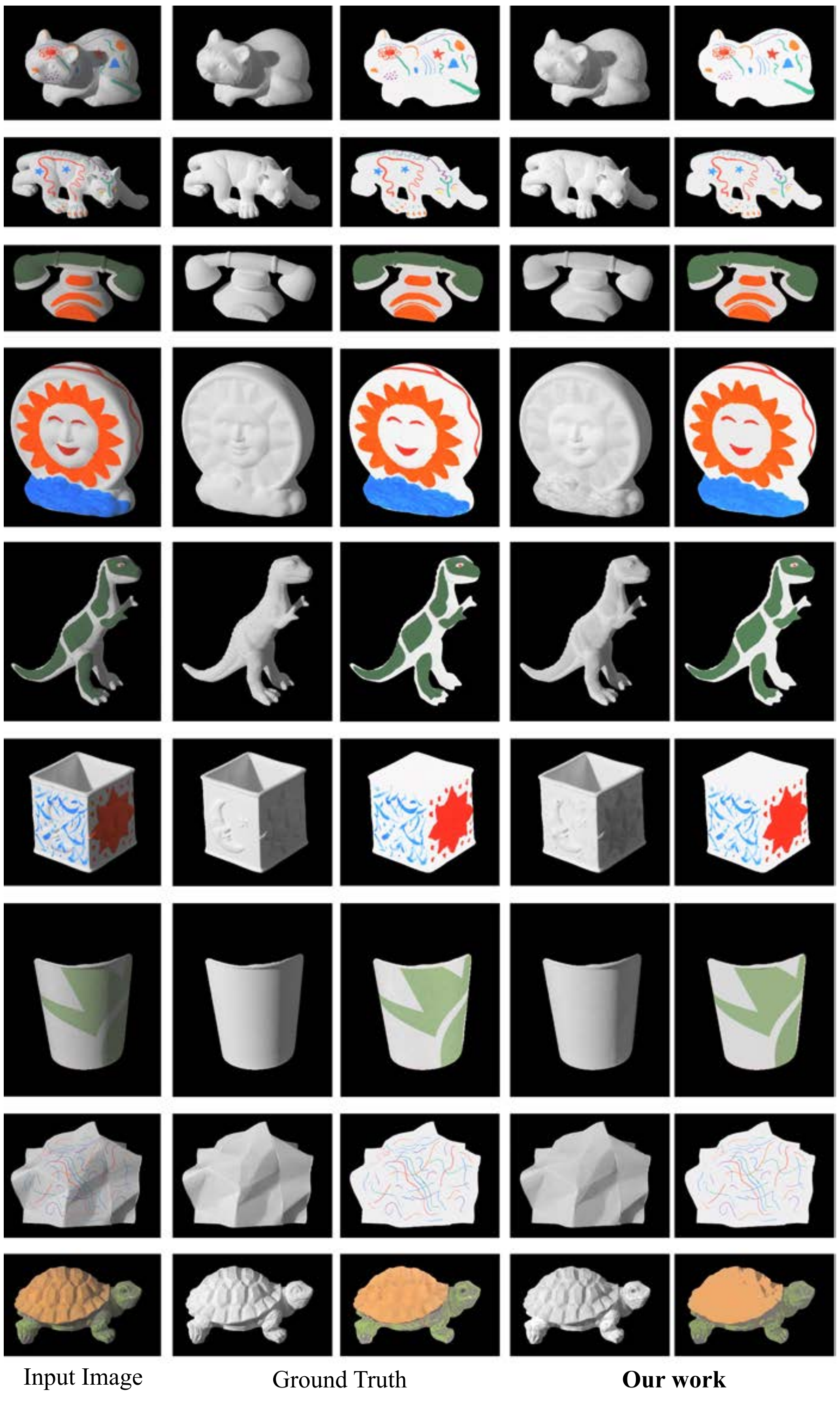


Figure 2.6: *Intrinsic images for the MIT dataset. First column, input image. Second and third columns, ground truth shading and reflectance. Last columns, our resulting shading and reflectance. The gamma of the images has been corrected for visualization purposes.*

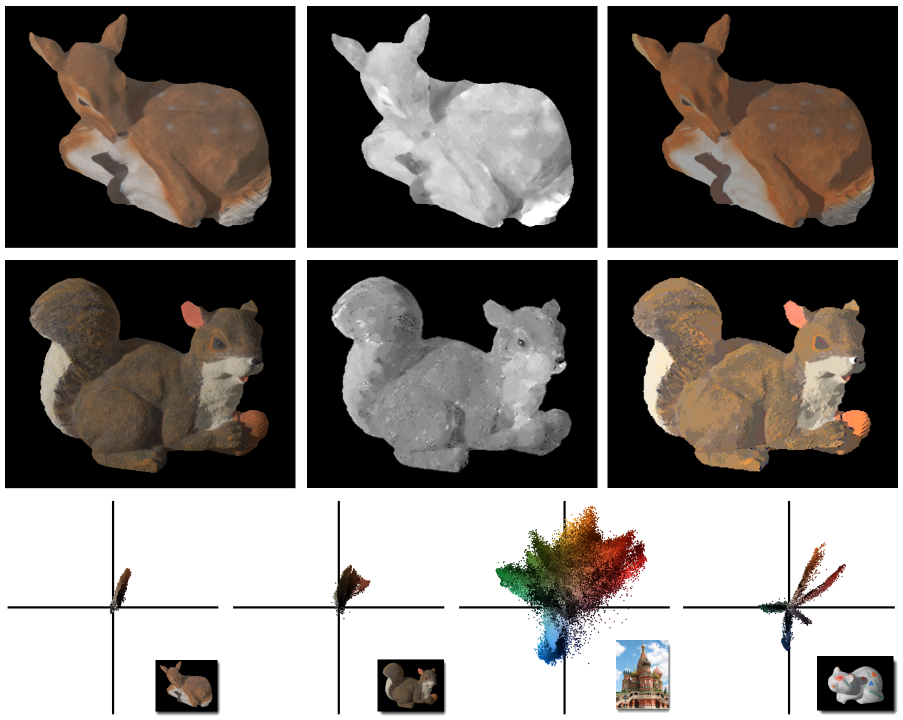


Figure 2.7: *A challenging case for our algorithm. Top rows, from left to right: input image, and our intrinsic shading and reflectance. Notice the shadow close to the tail in the reflectance image. Bottom row: Comparison of scatter plots. From left to right: deer, squirrel, St. Basil and raccoon. Notice the lack of chrominance variation in squirrel and deer, compared to the other two plots.*

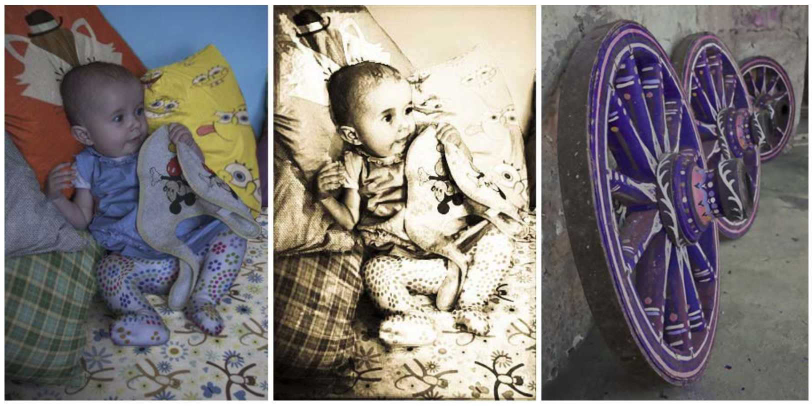


Figure 2.8: *Image edits accomplished using our intrinsic decompositions. Left: Re-texturing by editing the intrinsic reflectance. Center: Relighting of the previous result editing the intrinsic shading (sepia effect by editing intrinsic reflectance). Right: Global color edits on the intrinsic reflectance.*

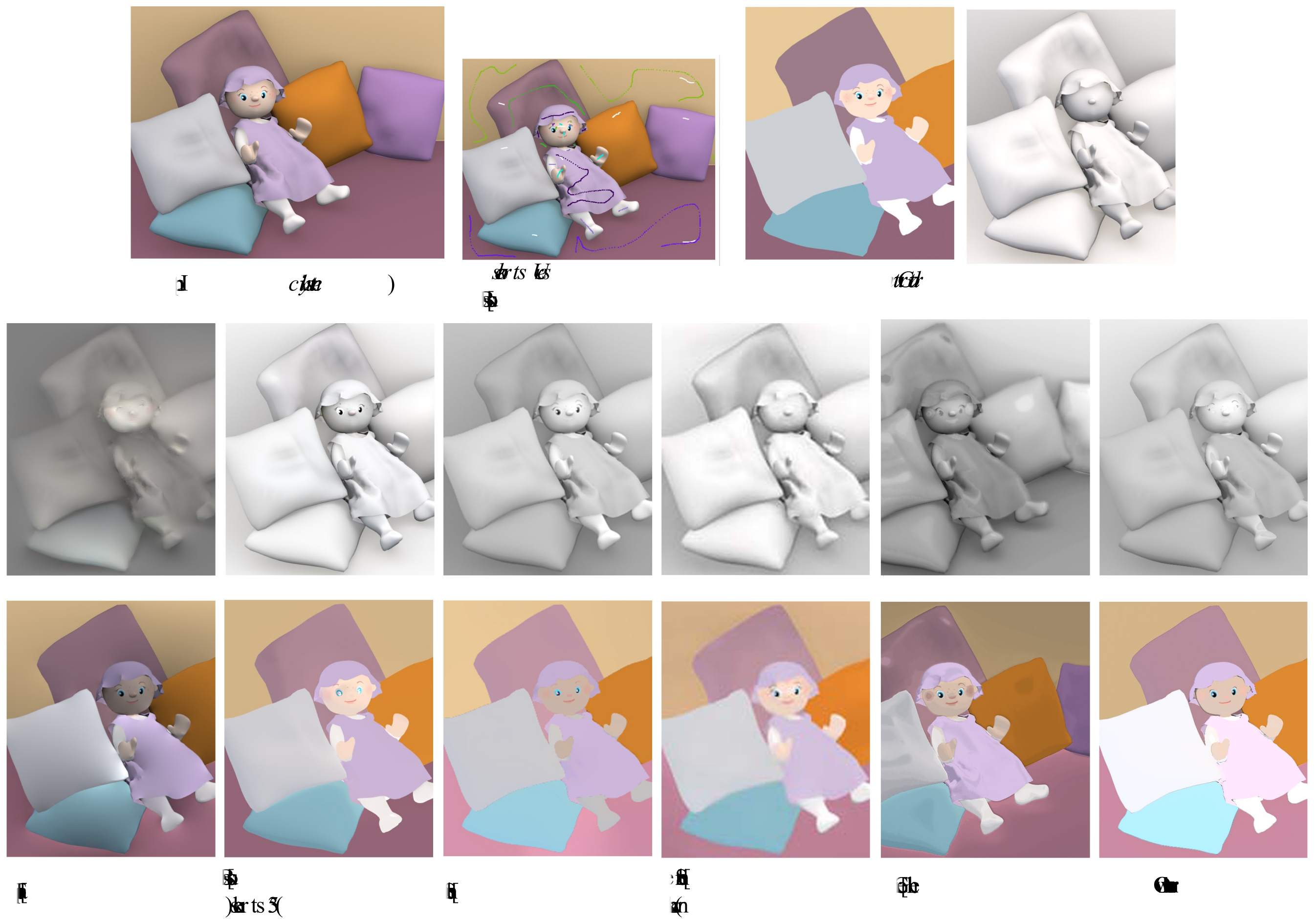


Figure 2.9: *Synthetic Scene and comparison with previous work.*

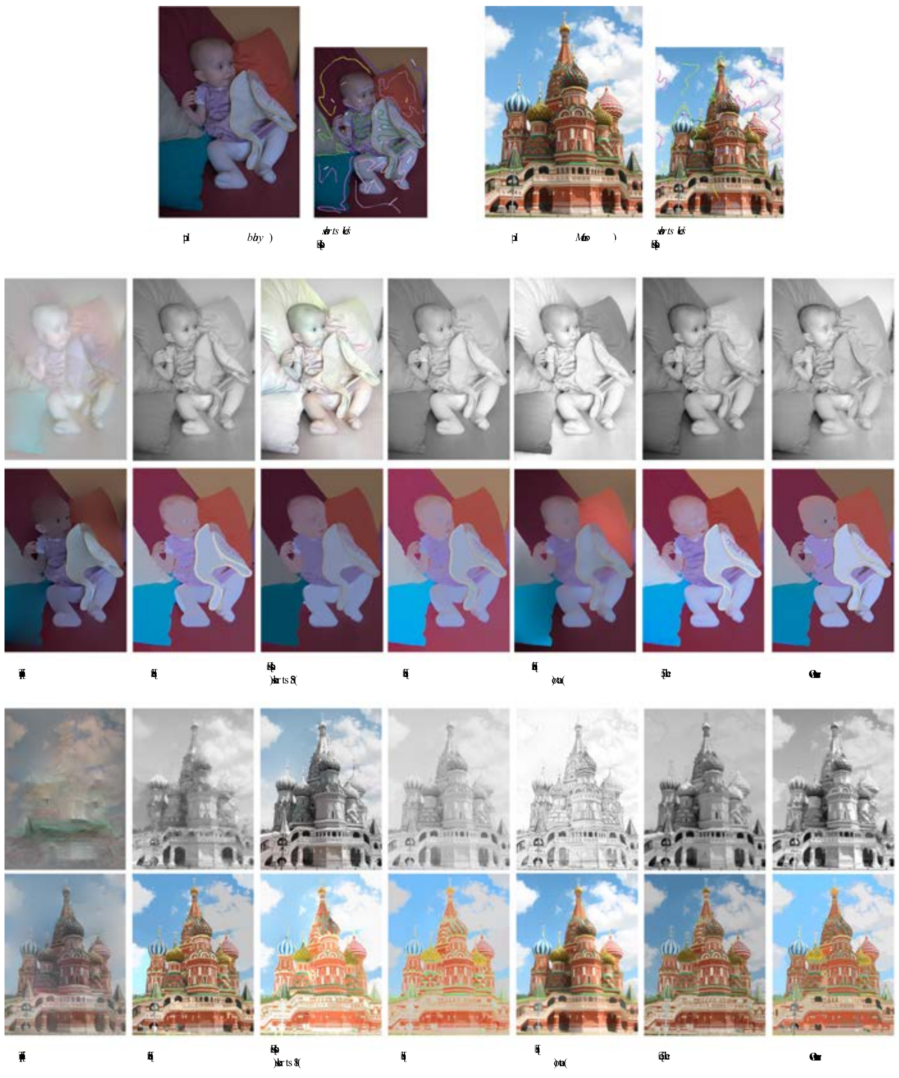


Figure 2.10: *Baby and St. Basil. baby: Our shading image is comparable with the one obtained by Bousseau et al. [22] which needs 58 user strokes. Note that the result by Shen et al. [163] has quantized the leg of the baby while our result keeps the continuity in the shading. The shading by Shen and Yeo [164] is not totally homogeneous and contains reflectance information. Moreover, our reflectance image successfully captures the facial features, outperforming the other methods. St. Basil: The methods by Shen et al. [163] and Tappen et al. [173] introduce obvious artifacts in the shading. Bousseau et al.'s method produces great results, although it requires 81 user strokes (original image by Captain Chaos, flickr.com).*

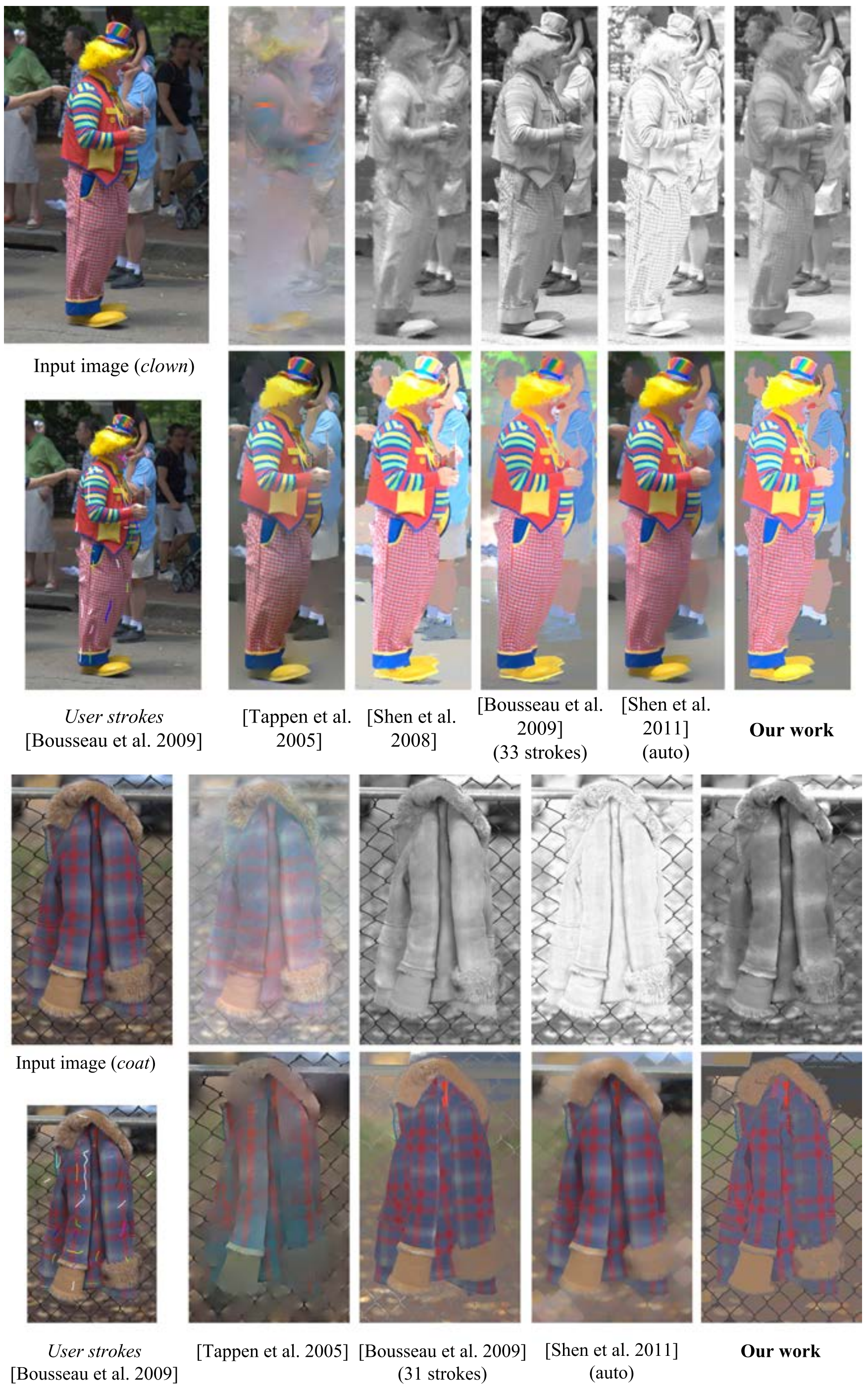

Input image (*clown*)
User strokes [Bousseau et al. 2009]
[Tappen et al. 2005]
[Shen et al. 2008]
[Bousseau et al. 2009] (33 strokes)
[Shen et al. 2011] (auto)
Our work
Input image (*coat*)
User strokes [Bousseau et al. 2009]
[Tappen et al. 2005]
[Bousseau et al. 2009] (31 strokes)
[Shen et al. 2011] (auto)
Our work

Figure 2.11: *Top: clown. The methods by Shen et al. [163] and Tappen et al. [173] introduce obvious artifacts in the shading. Our result is comparable with Bousseau et al. [22] without requiring user strokes. Bottom: coat. Our result without any user strokes is not far from the result obtained by Bousseau et al. [22] using 28 strokes.*

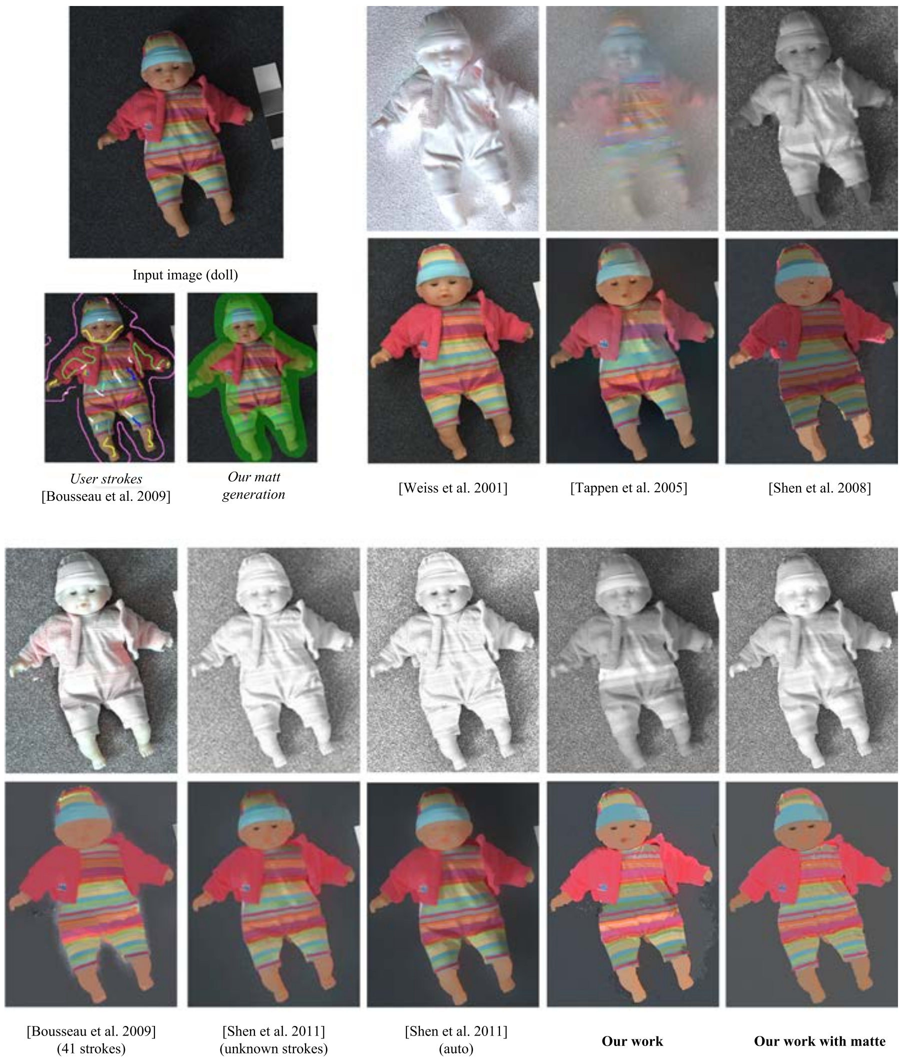


Figure 2.12: *doll. The automatic methods by Shen et al. [163] and Tappen et al. [173] fail to obtain an homogeneous shading on the legs.*

APPENDICES

2.A ADDITIONAL RESULTS: CLUSTERING AND DECOMPOSITION PER SCENE

Figures 2.13– 2.21. (From left to right) First column, input image and scatter plot of pixel data in the (a,b) plane (Lab color space). Second column, k-means segmentation according to (a,b) pixel coordinates; third column, final clustering yielded by our method taking into account spatial information (both, second and third rows, are depicted in false color). Last columns, the resulting shading and reflectance intrinsic images.

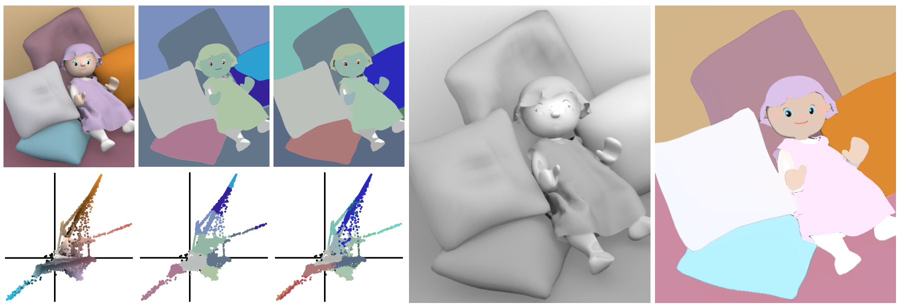


Figure 2.13: *Synthetic*

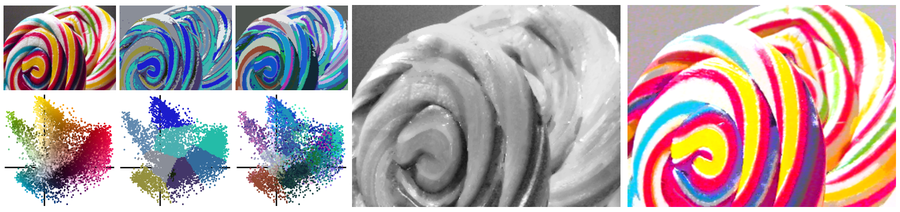


Figure 2.14: *Lollipop (original image by Thalita Carvalho, flickr.com)*

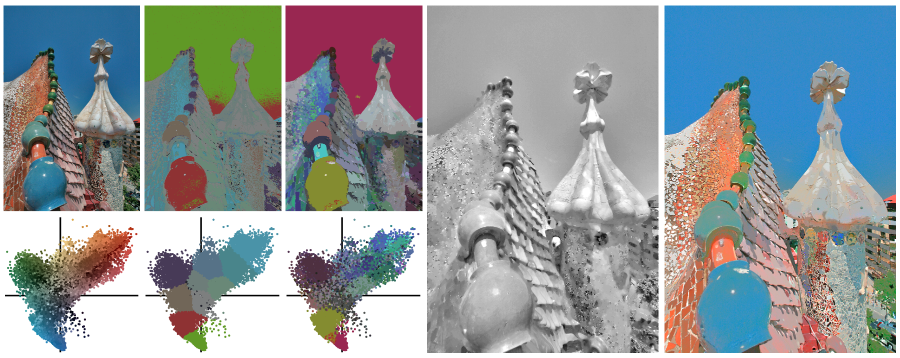


Figure 2.15: *Batlló house (original image by lukasz dzierzanowski, flickr.com)*

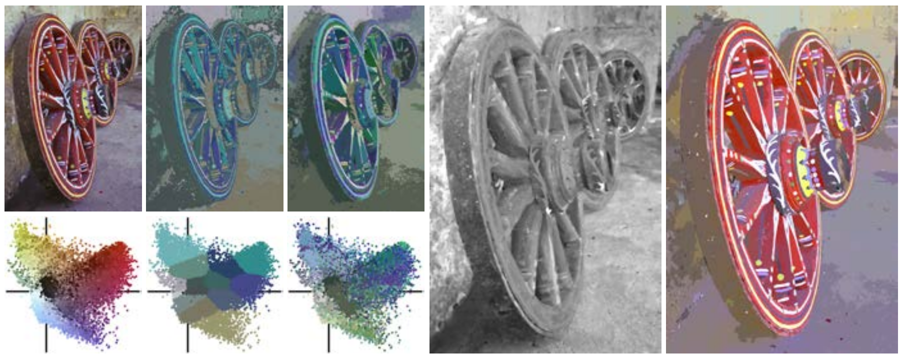


Figure 2.16: *Wheels (original image by Angela Smith Kirkman)*

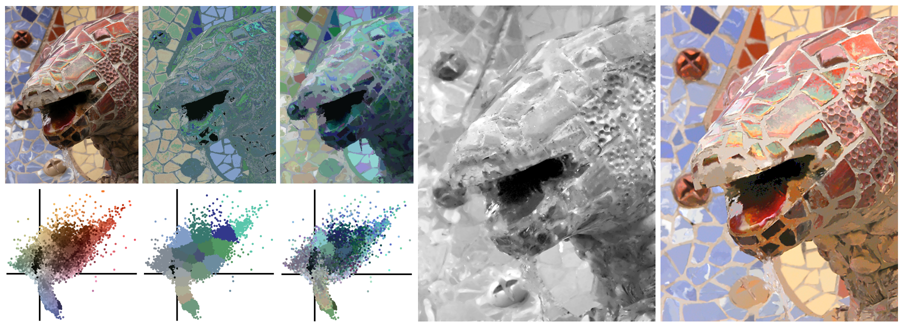


Figure 2.17: *Dragon (original image by Jordanhill School D&T Dept, flickr.com)*

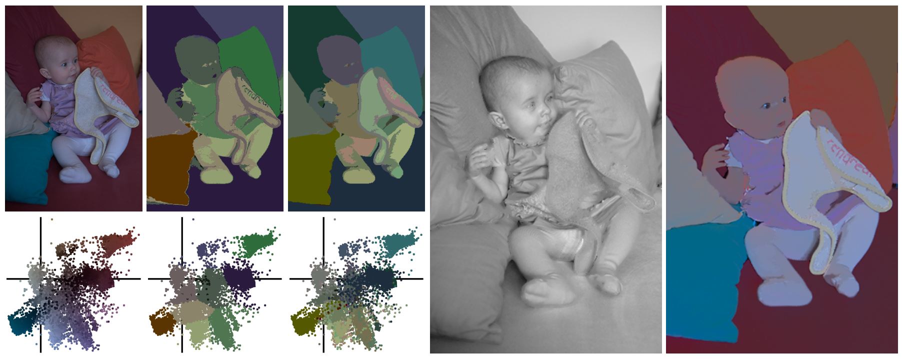


Figure 2.18: *Baby*

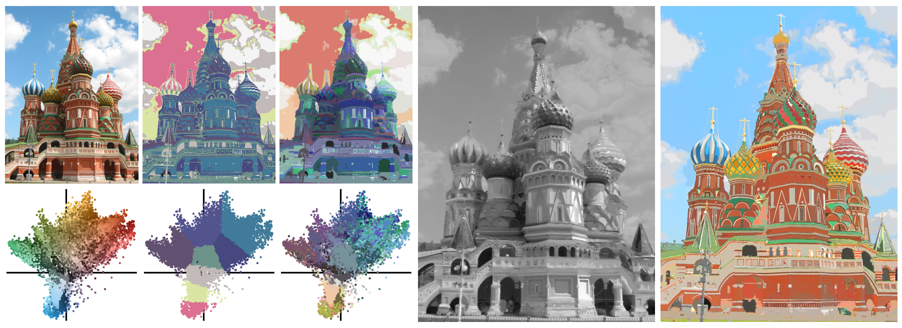


Figure 2.19: *St. Basil (original image by Captain Chaos, flickr.com)*

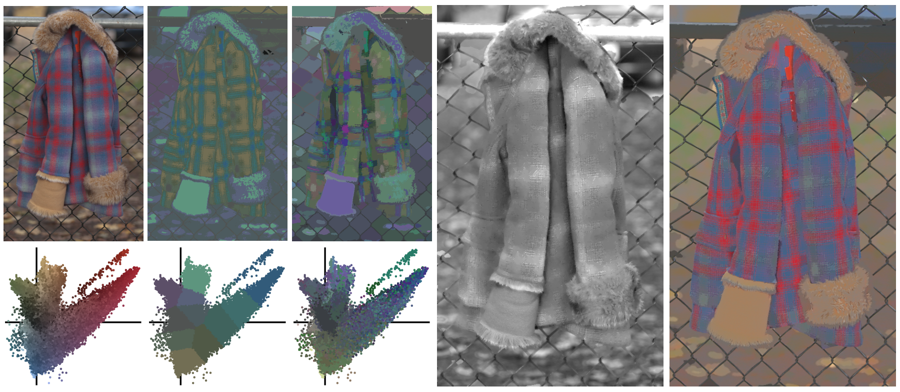


Figure 2.20: *Coat*

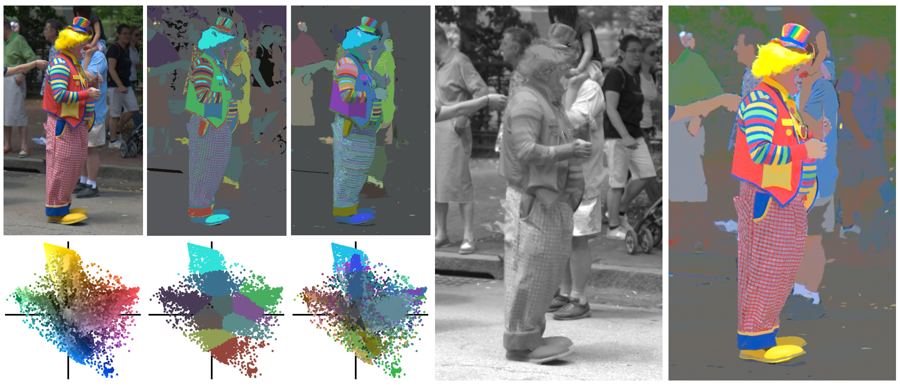


Figure 2.21: *Clown*

3 INTRINSIC VIDEO DECOMPOSITION

In this chapter we extend the problem of intrinsic image decomposition to the *temporal dimension* and present a method to decompose a *video* into its intrinsic components of reflectance and shading, plus a number of related example applications in video editing such as segmentation, stylization, material editing, recolorization and color transfer. Intrinsic decomposition is an ill-posed problem, which becomes even more challenging in the case of video due to the need for temporal coherence and the potentially large memory requirements of a global approach. Additionally, user interaction should be kept to a minimum in order to ensure efficiency. We propose a probabilistic approach, formulating a Bayesian Maximum a Posteriori problem to drive the propagation of clustered reflectance values from the first frame, and defining additional constraints as priors on the reflectance and shading. We explicitly leverage temporal information in the video by building a causal-anticausal, coarse-to-fine iterative scheme, and by relying on optical flow information. We impose no restrictions on the input video, and show examples representing a varied range of difficult cases. Our method was the first one designed explicitly for video; moreover, it naturally ensures temporal consistency, and compares favorably against the state of the art in this regard. This work has been published in *ACM Transactions on Graphics* and presented at *SIGGRAPH 2014*.

Y. Genzhi, E. Garces, Y. Liu, Q. Dai & D. Gutierrez
Intrinsic Video and Applications
ACM Transactions on Graphics (SIGGRAPH), Vol.33 (4), 2014

3.1 INTRODUCTION

Decomposing an image into its intrinsic shading and reflectance layers is an ill-conditioned problem with direct applications in computer graphics and image processing, such as retexturing and relighting. In the past few years there have been numerous works tackling this problem from different angles. Some propose fully automatic methods from single images [173, 57], or rely on user annotations [22, 162]. Others leverage information from multiple images [100] or time-lapse sequences [182, 168]. Nevertheless, the problem of decomposing a *video* shot into its intrinsic components remains unexplored.

Intrinsic video decomposition is particularly challenging, since temporal coherence must be preserved, even in the presence of dynamic lighting or occluded surfaces coming into view due to object or camera motion. Naively applying any existing intrinsic images algorithm to every individual frame yields poor results, due to the extremely ill-posed nature of the problem and the lack of built-in temporal coherence. Solving the problem on a few keyframes and then interpolating also leads to bad results, since there is no guarantee that resulting reflectance values will be coherent across keyframes. Last, trying to solve the problem globally for the whole sequence would be impractical due to large memory requirements. Another possible approach would be to rely on video segmentation: however, given the rich complexity of video shots, no existing algorithm can guarantee a reliable temporal coherent segmentation. Instead, we focus on an accurate and efficient *propagation* of the reflectance from an initial intrinsic decomposition on the first frame.

Propagating reflectance is different from other propagation approaches (such as video colorization) given the impossibility of building a reliable feature space: The information we wish to propagate (reflectance) is not explicitly coded in the RGB values of the frames and, as we argued, obtaining it on a per-frame basis leads to disturbing flickering artifacts. Instead, we propose a relaxed propagation approach based on a Bayesian framework and solve a Maximum A Posteriori (MAP) problem. To avoid accumulation errors, we define a local confidence threshold and stop the propagation when the number of unreliable pixels surpasses it. We then leverage *shading* information to complete the reflectance layer at the stopping frame, and propagate backwards. We iterate this process using a coarse-to-fine approach.

Our approach has the following desirable characteristics: 1) it is efficient, since at each step it only uses information from the current and previous frame; 2) it takes advantage of information in the temporal dimension given its causal-anticausal scheme; 3) it is temporally stable; 4) it leverages the characteristics of intrinsic images, both from reflectance and shading; and 5) it keeps cumbersome user interaction to a minimum (most of the results shown in this chapter are fully automatic, while a few others required a few scribbles on the first frame only). Additionally, we show some video editing applications of our work, including segmentation, recolorization, color transfer, stylization and material editing.

3.2 RELATED WORK

INTRINSIC IMAGE DECOMPOSITION FROM A SINGLE IMAGE Automatic decomposition of a single image into its intrinsic components usually relies on Retinex theory [102], by analyzing local pixel derivatives, to distinguish a change in reflectance from a change of shading [173]. Some methods add priors to a Retinex-based framework, either on the illumination, the reflectance, or both [83, 59, 164]. Garces et al. [57] build clusters of similar chromaticity, from which connections and constraints are defined. Inspired by this, we work on a simplified cluster space of similar *reflectance*, obtained from an initial single-frame decomposition. Lombardi and Nishino [114] introduce a probabilistic formulation with data-driven and entropy constraints, but the method is limited to obtaining the reflectance of single objects with homogeneous materials. A recent work by Zhao et al. [197], building on the work by Shen et al. [163], formulates Retinex as a linear optimization, forcing distant pixels with the same texture to have the same reflectance. A different set of techniques rely on user intervention to constrain the problem by specifying sparse sets of pixels with similar properties [22, 162]. Applying any of these techniques on each frame of a video causes disturbing flickering artifacts, due to the lack of built-in temporal coherence mechanisms.

INTRINSIC IMAGE DECOMPOSITION FROM MULTIPLE IMAGES A few methods try to leverage information from multiple images of the same static scene, and analyze pixel variations under varying illumination [182, 69]. A more flexible solution is given by Laffont et al. [100], which reconstructs a point-based 3D representation of the scene. Similarly, Liu et al. [113] use several images of the same scene with the purpose of image colorization.

Different from these techniques, our method is designed to work on video sequences, where the key simplifying assumption of a static scene no longer holds.

INTRINSIC VIDEO DECOMPOSITION The most similar approach to ours is the work of Yan et al. [187]. The authors rely on per frame intrinsic decomposition in *local* regions, in order to re-texture specific objects in a video. However, editing a complete video this way would again produce temporal inconsistencies. Specific intrinsic video has only recently been done by Lee et al. [104], but it requires additional depth information from Kinect. Matsushita et al. [123] use a simpler approach where the goal is only to eliminate shadows in open scenes, as a pre-processing step for video surveillance, and not to produce a complete intrinsic decomposition. Finally, the heuristic approach of Sunkavalli et al. [168] decompose a time-lapse sequence into its intrinsic components assuming certain properties of the sky light of the scene. The method works well on static images, but is not able to cope with camera movements.

VIDEO COLORIZATION Colorization algorithms are also somewhat related to our problem [106, 190, 16, 142]. However, better results can be obtained for such a task if the intrinsic components are available, where shading information does not interfere in the process. Moreover, these methods usually require user input every few frames; in contrast, our method is fully automatic in most sequences, requiring at most minimal user input only on the first frame.

TEMPORAL CONSISTENCY Guaranteeing temporal consistency across frames is a challenge for video editing algorithms. Paris [144] combines isotropic diffusion and Gaussian convolution to adapt classical algorithms like mean shift segmentation or bilateral filtering to video streams. Farbman and Lischinski [43] propagate values using a combined technique of optical flow and interpolation, for the purpose of tonal stabilization. Recently, Bonneel et al. [20] applied curvature-flow smoothing in the space of color transformations to transfer color palettes between videos. These techniques are adapted to the particular problems they address, and cannot be easily modified for our purposes. Last, Lang et al. [103] propose an efficient framework to enforce temporal smoothness across frames. They approximate a global optimization, and show very good results for applications such as disparity estimation, depth upsampling or colorization. However, even after a decent amount of scribbles and parameter tuning, there is an inherent trade-off between the spatial filtering necessary to perform temporal filtering. For the case of intrinsic video, given the large variability across frames of the individual intrinsic decompositions, this causes clear ghosting artifacts, as Figure 3.10 shows.

3.3 OVERVIEW

Let $f_t(t = 0, 1...)$ be the frames of an input video sequence, with color values $I_{p,t}$ for each pixel p in each time step t. At each frame, the intrinsic decomposition problem can be formulated as finding the reflectance $R_{p,t}$ and shading $S_{p,t}$ layers that satisfy:

$$I_{p,t} = R_{p,t} \times S_{p,t} \tag{3.1}$$

where $\times$ denotes per-channel multiplication. The great challenge lies in ensuring temporal coherence of the results, while avoiding the huge memory requirements that analyzing the video as a whole would impose.

We first obtain the reflectance-shading decomposition of the first frame, and obtain clusters of similar reflectance. We then formulate a probabilistic framework that allows us to propagate reflectance values along the frames of the video. In particular, we solve a Maximum A Posteriori (MAP) problem, using a smoothness prior and imposing additional constraints to ensure robustness in the results. At each step, we only propagate values above a certain confidence threshold, and stop the propagation when the number of unknown pixels becomes significant. We follow a coarse-to-fine approach, and perform a *local* decomposition at the stopping frame, using known reflectance values as constraints, and proceed to propagate the new values backward in time. This combined propagation-completion, forward-backward approach allows us to deal with challenging cases like occluded objects.

This process is iterated three times. Last, we apply a smoothness constraint on the shading on the few remaining unassigned pixels, from which we derive the missing reflectance values by simple division. The algorithm then begins again the propagation in a similar manner from the last processed frame, until a new stopping frame is found or the whole sequence is processed. Figure 4.5 and Algorithm 1 show an overview.

Algorithm 1 Intrinsic Video decomposition

```
1:  $[R_0, S_0]$ = IntrinsicImageDecomposition($I_0$)
2:  $[\mathcal{C}, \mathcal{I}, \mathcal{R}]$ = ReflectanceClustering($I_0, R_0$)
3:  for each iteration do
4:      $f_t \leftarrow 1$
5:      repeat
6:          $f_s \leftarrow lastFrame$
7:          /* Forward propagation */
8:          for $f := f_t : lastFrame$ do
9:              $R_f \leftarrow$ ReflectancePropagation($R_{f-1}, \mathcal{C}, \mathcal{I}$)
10:             if StoppingCondition($R_f$) then
11:                 $f_s = f$
12:                 break
13:             end if
14:         end for
15:         $[R_{f_s}, S_{f_s}] \leftarrow$ ReflectanceCompletion($R_{f_s}, S_{f_s}$)
16:         $[\mathcal{C}, \mathcal{I}, \mathcal{R}] \leftarrow$ ClusteringUpdate($R_{f_s}, \mathcal{C}, \mathcal{R}$)
17:         /* Backward propagation */
18:         for $f := f_s : f_t$ do
19:             $R_f \leftarrow$ ReflectancePropagation($R_{f+1}, \mathcal{C}, \mathcal{I}$)
20:         end for
21:         $f_t \leftarrow f_s + 1$
22:     until $f_s = lastFrame$
23: end for
24: $[R, S] \leftarrow$ ResidualCompletion($R, S$)
25: return $R, S$
```

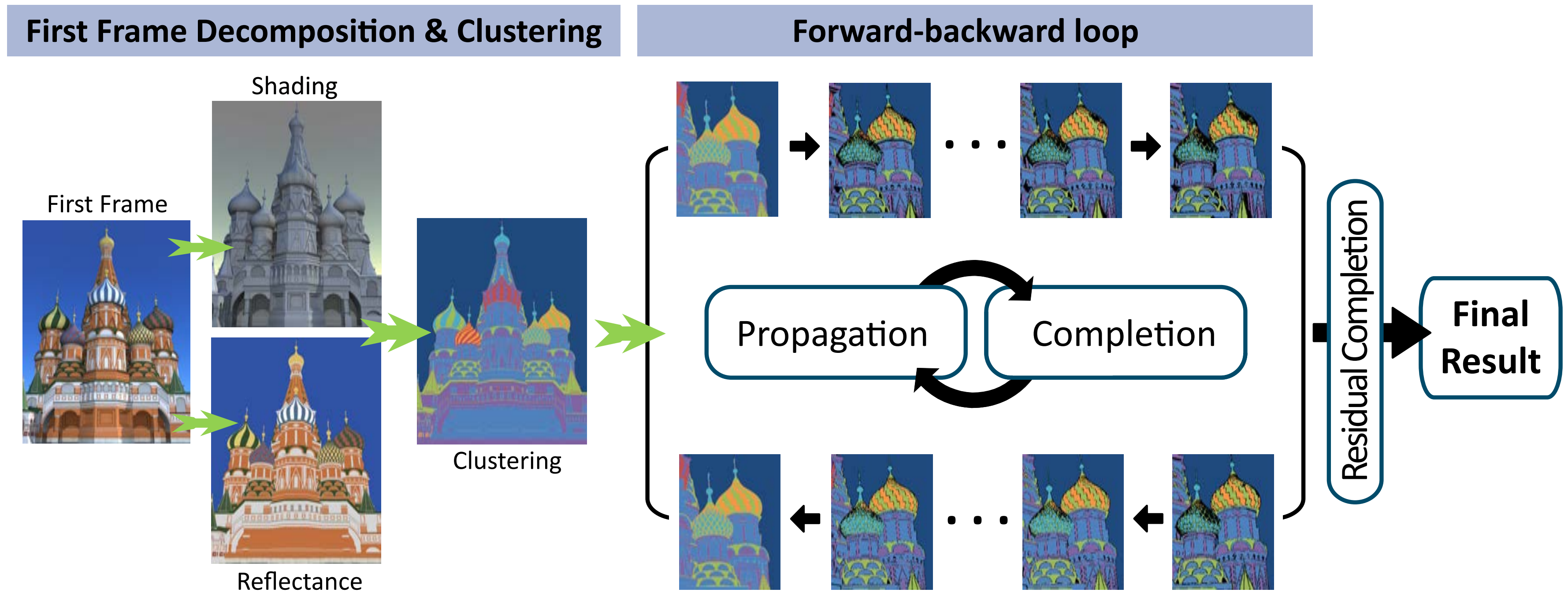


Figure 3.1: *Overview of our algorithm. We first decompose the initial frame into its intrinsic components. To reduce inconsistencies, we first cluster the initial reflectance values (Section 6.5). A forward-backward loop propagates this clustered reflectance in the temporal domain, leaving out unreliable pixels (Section 3.5). After this propagation step, some pixels may still remain unassigned; we rely on Retinex theory to apply shading constraints, thus obtaining the final result (Section 3.6).*

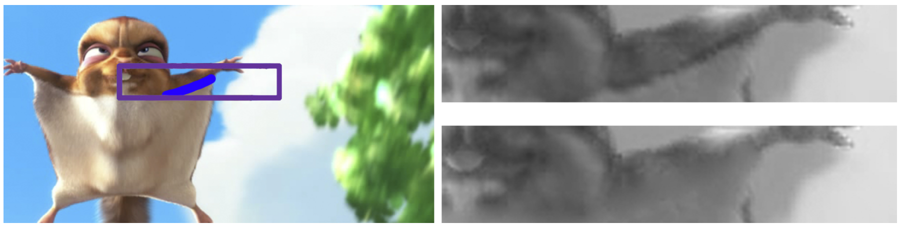


Figure 3.2: *Local optimization. Using a global threshold T, reflectance variations with low chromaticity change may be mis-classified as shading (top-right). By allowing T to vary locally in a user-defined region, this is corrected (bottom-right). The user-defined scribble appears in blue.*

3.4 INITIAL DECOMPOSITION AND CLUSTERING

We first aim to find a clustered reflectance decomposition on the first frame, which will be the input to our probabilistic propagation framework. Given the ill-posed nature of the problem, we need a flexible approach that produces good results automatically, while allowing a certain amount of user interaction to improve the results if needed. We base our approach on a common Retinex formulation [102]; in particular we extend the single-image formulation by Zhao et al. [197], who define a global optimization framework and allow user interaction with constant-albedo and constant-shading strokes [22] (refer to Appendix 3.A for a quick overview). Naively running the algorithm on every frame yields obvious flickering artifacts (see Section 3.8 and the accompanying video), while extending the optimization to the whole video is prohibitively expensive and may lead to poor results.

We thus extend the original formulation by Zhao et al. in three ways: First, by allowing the user to define *local* optimizations in selected areas of the image. Second, by moving from a pixel-based representation to a more consistent (and efficient) cluster-based approach. And third, by imposing additional constraints during propagation, derived from temporal information in the video. We present details of the first two extensions in the following paragraphs, while the third is explained in Section 3.5.1.

LOCAL OPTIMIZATIONS The global optimization by Zhao and colleagues is based on a global threshold T defined over the whole image, which determines how sensitive the decomposition is to changes in chromaticity. In particular, the authors define a balance factor $\omega_{p,q}$ (where p, q denotes all neighboring pairs of pixels) as follows:

$$\omega_{p,q} = \begin{cases} 0, & \text{if } \| J_p - J_q \| > T \\ 100, & \text{otherwise} \end{cases} \tag{3.2}$$

where J denotes chromaticity. We set $T = 10^{-3}$ for all the results shown in the chapter. A lower value of T tends to produce an over smooth shading, while higher values assign most of the intensity variations to the shading layer, approximating the results to a chrominance-luminance decomposition. Since in Zhao's formulation the threshold T is defined globally, this may lead to unequal results in different parts of the image.

We thus allow for local variations of T within different regions of the image, which the user identifies by simply drawing approximate masks (using for instance lazy snapping [111]). Following Retinex theory, we assume that within a mask shading varies smoothly, which is therefore captured by the lower intensity gradients. We then redefine T locally as $T' = 0.05 \times \mu(\nabla J)$, where μ is the mean of the chromaticity gradient of the pixels inside the mask. Figure 3.2 shows an example.

REFLECTANCE CLUSTERING Given the large volume of data contained in a video, a pure pixel-based approach is inefficient, while being more error-prone and causing temporal instabilities in the form of jittering artifacts. Hence, we *cluster* the initial reflectance decomposition by grouping together sets of pixels sharing similar reflectance values. To reduce memory requirements, we first obtain an over-segmented image, where all pixels in a segment are spatially connected. We use a graph-based segmentation approach [45]. Although this method is very sensitive to changes in parameters, we take advantage of the fact that we only need a rough initial segmentation, and use the same fixed parameters for all our results: size of the Gaussian blur $\sigma = 0.8$, segmentation cut threshold $\mathcal{K} = 50$, and minimum size of the segment $min = 256$. We then group these segments into larger clusters where pixels no longer need to be connected in image space. We iteratively merge two segments if the difference between their average RGB values is smaller than the specified threshold $\mathcal{K}$. This process yields our final reflectance clustering $\mathcal{C}$.

Additionally, we create a suitable data structure for our subsequent reflectance propagation (Section 3.5). This structure maps image values with clustered reflectance values. For each cluster $C_k \in \mathcal{C}$, we compute two different tables indexed by 3D (RGB) vectors. The first table $\mathcal{R}_k$ is simply a histogram of reflectance values; the function $\rho(k, \mathbf{r})$ returns the number of pixels with reflectance $\mathbf{r}$ in cluster C_k. The second table $\mathcal{I}_k$ stores, on the one hand, a histogram of color values, where the function $\gamma(k, \mathbf{l})$ returns the number of pixels with color $\mathbf{l}$ in cluster C_k; additionally, it stores the associated reflectance $\mathbf{r}_{k,l}$ for each color $\mathbf{l}$. If more than one pixel have the same color but different reflectances in the same cluster, the average reflectance is stored. This helps ameliorate possible inconsistencies in the initial decomposition, yielding a more coherent reflectance. Tables $\mathcal{R}_k$ and $\mathcal{I}_k$ will be later used and updated during our probabilistic reflectance propagation and completion steps (Sections 3.5 and 3.6). Figure 3.3 shows a comparison of our result with the initial decomposition of Zhao et al. [197].

3.5 REFLECTANCE PROPAGATION

We now proceed to propagate reflectance values along the temporal dimension, following a coarse-to-fine iterative approach. Reflectance propagation consists of three steps: forward propagation, reflectance completion, and backward propagation. Last, a final residual reflectance completion takes place. Figure 3.4 illustrates this process.

3.5.1 *Probabilistic framework*

We introduce a Bayesian formulation for reflectance propagation, which enables the integration of reflectance-color statistical inference from tables $\mathcal{R}_k$ and $\mathcal{I}_k$, which are updated at each step, plus a location-reflectance prior from the former frame. Moreover, we jointly optimize pixel clustering with reflectance propagation. With this combined analysis on color, intrinsic reflectance and clustering, we obtain temporally consistent results.

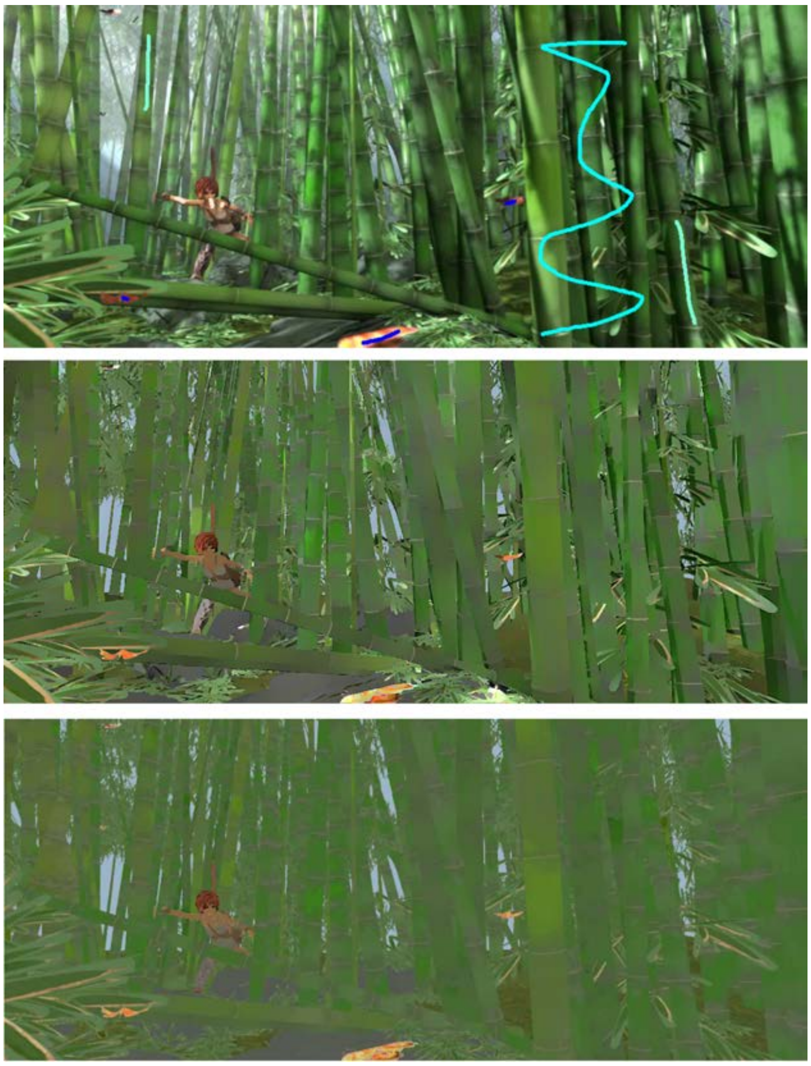


Figure 3.3: *Reflectance clustering. Top: Input frame with user scribbles for constant-albedo regions. Middle: Reflectance from Zhao et al. [197]. Bottom: Our reflectance after the clustering step. Note that our reflectance is more consistent and lends itself better to temporal propagation.*

Given a pixel p in frame f_t, with an input RGB color vector $\mathbf{I}_{p,t}$, we aim to find the matching cluster in the previous frame f_{t-1} with the most similar reflectance value (the reflectance of pixel p is unknown at this point), while avoiding over-fitting. We introduce a probabilistic formulation to find the cluster index $\bar{k}$ which maximizes the posterior probability $\mathcal{P}(C_k|p)$. We can define a Maximum A Posteriori (MAP) problem as:

$$\bar{k}(p) = \arg\max_k \mathcal{P}(C_k|p) \propto \mathcal{P}(p|C_k) \cdot \mathcal{P}(C_k) \tag{3.3}$$

where $\mathcal{P}(p|C_k)$ is the likelihood that pixel p belongs to cluster C_k, and $\mathcal{P}(C_k)$ acts as a prior. We discuss these two terms in the following paragraphs.

To limit our search to the most probable candidate regions (thus improving accuracy and efficiency), we leverage the information contained in the video and compute optical flow [23] between f_t and f_{t-1}, to obtain the cross-frame motion vector $\boldsymbol{u}_p$. This vector defines the corresponding pixel p' in frame f_{t-1} as $p' = p + \boldsymbol{u}_p$. We then only query the pixels inside an $N \times N$

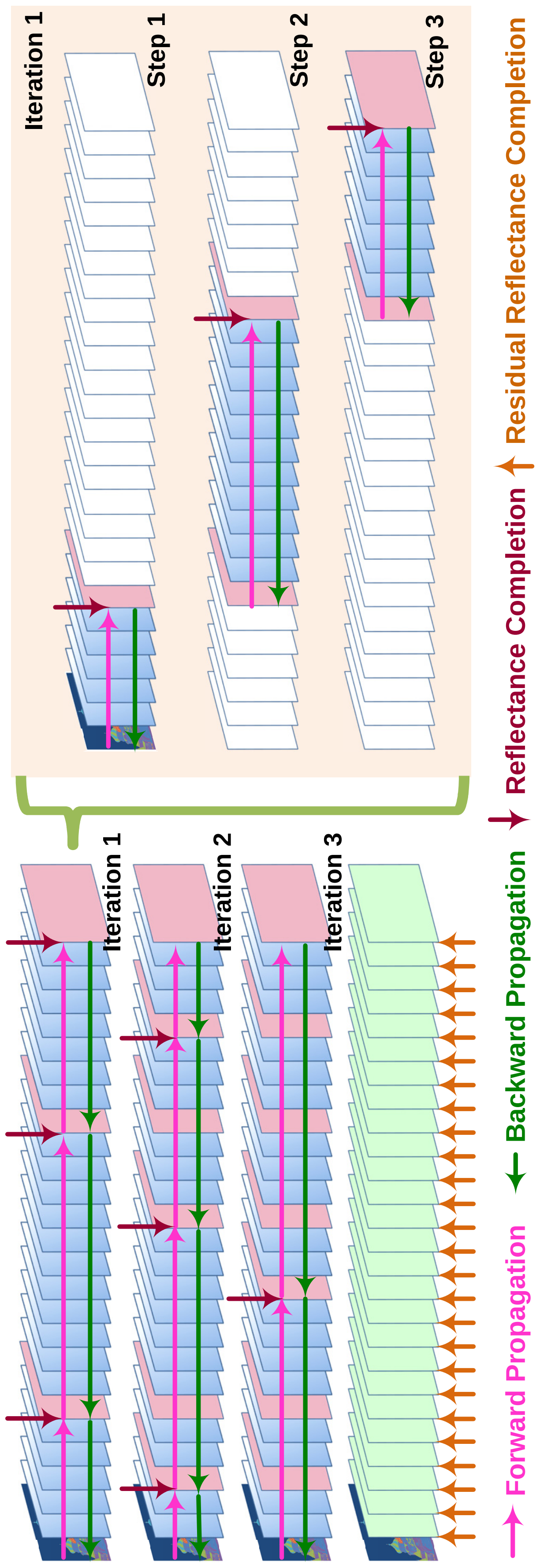


Figure 3.4: *Reflectance propagation. We propagate reflectance information over successive frames, until a stopping frame is found (depicted in pink). We then apply a reflectance completion step on the stopping frame, and proceed to propagate backwards. We run this basic cycle of forward propagation - reflectance completion - backward propagation until we reach the end of the video, and iterate three times in a coarse-to-fine approach. Last, we perform a residual reflectance completion step.*

($N = 50$) window $W_{p'}$ defined in f_{t-1} and centered at p'. Our MAP problem (Equation 3.3) is therefore only solved for all the clusters intersecting window $W_{p'}$.

likelihood $\mathcal{P}(p|C_k)$ The likelihood of a pixel p belonging to a cluster C_k is based on the probability density function of cluster C_k. We formulate a standard non-parametric density estimation problem to estimate the fitness of assigning a pixel to a cluster according to pre-clustered pixels from the previous frame, for which a Parzen window-based approximation is a convenient and effective method. We define our normalized likelihood term $\mathcal{P}(p|C_k)$ as:

$$\mathcal{P}(p|C_k) = \frac{1}{|\mathcal{I}_k|} \sum_{\mathbf{l}} \gamma(k, \mathbf{l}) G\left(\frac{\mathbf{l} - \mathbf{l}_{p,t}}{d}\right) \tag{3.4}$$

where $|\mathcal{I}_k|$ is the total number of pixels of cluster C_k, and G represents a Parzen window defined by a 3-D Gaussian kernel function, with width $d = 5$. In this form, $\mathcal{P}(p|C_k)$ represents the probability density function of cluster C_k. Figure 3.5 (c) shows the propagated reflectance at this stage.

However, we note that this definition is biased towards clusters with a large number of pixels. Since each cluster may contain several different reflectances, if we only divide Equation 3.4 by $|\mathcal{I}_k|$, reflectances with more pixels within the cluster (large $\gamma(k, \mathbf{l})$) will dominate. We thus introduce a unit function U:

$$U(\gamma) = \begin{cases} 1, & \gamma > 0 \\ 0, & \gamma = 0 \end{cases} \tag{3.5}$$

and redefine $\mathcal{P}(p|C_k)$ as:

$$\mathcal{P}(p|C_k) = \frac{1}{|U(\mathcal{I}_k)|} \sum_{\mathbf{l}} U(\gamma(k, \mathbf{l})) G\left(\frac{\mathbf{l} - \mathbf{l}_{p,t}}{d}\right) \tag{3.6}$$

Note that we also apply the unit step function on $\mathcal{I}_k$, where $|U(\mathcal{I}_k)|$ represents the number of *non-zero* entries in $\mathcal{I}_k$. Including the unit step function in our formulation makes the likelihood independent of the number of pixels in each cluster, effectively removing bias (see Figure 3.5 (d)).

prior $\mathcal{P}(C_k)$ We adopt a smoothness prior common to many MAP solutions. Given the pixel p' computed from p by optical flow, we define $D(p', q')$ as the Euclidean spatial distance between pixels p' and q', where q' belongs to the same frame as p' and satisfies the following three conditions: $q' \in W_{p'}, q' \in C_k$ and $q' \neq p'$. Our prior thus becomes:

$$\mathcal{P}(C_k) = \min_{q'} \left(D(p', q')\right)^{-\frac{1}{2}} \tag{3.7}$$

Note that $\mathcal{P}(C_k)$ is independent of the intrinsic properties of a pixel, and can be calculated after the former frame has been processed: it thus acts as a prior even if it is not explicitly formulated as a probability. Figure 3.5 (e) shows how the smoothness prior improves the reflectance propagation.

3.5.2 *Additional constraints*

Solving our MAP problem in Equation 3.3 for frame f_t, we obtain the cluster candidate $\bar{k}(p)$ indicating the best cluster match for pixel p. However, assign-

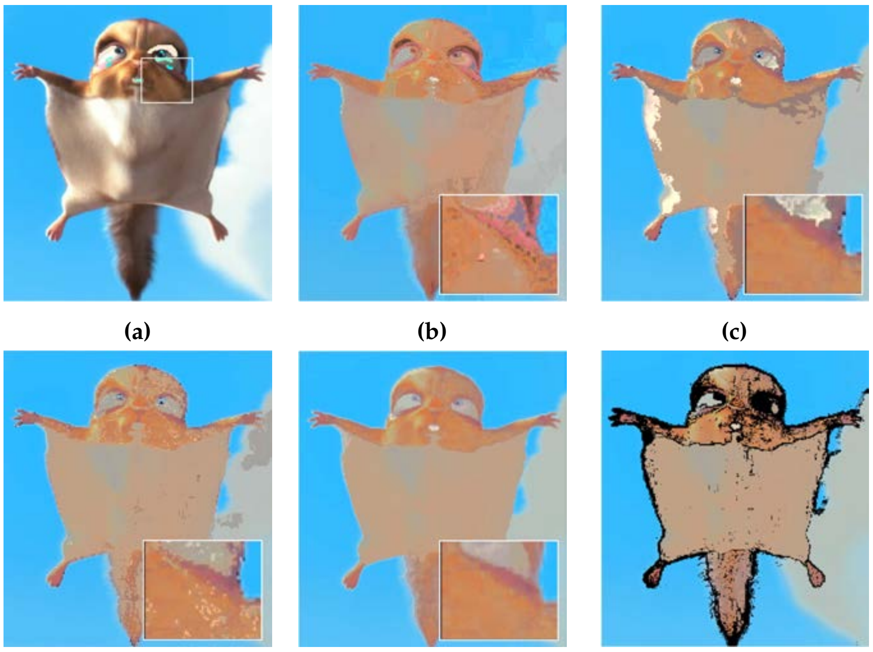


Figure 3.5: *Effect on reflectance propagation of every component of our algorithm: (a) Original color frame (including user strokes for constant-albedo regions; highlighted area enlarged in the insets). (b) Reflectance propagation without clustering. (c) Reflectance propagation using Eq. 3.4. (d) Reflectance propagation using Eq. 3.6, adding the step function U. (e) Reflectance propagation using both Eq. 3.6 and the distance term of Eq. 3.7. (f) Removing low-confidence reflectance using Section 3.5.2 on the result of (e).*

ing the corresponding cluster index to every pixel before proceeding to the next frame would accumulate errors over time (as Figure 3.11 in Section 3.8 shows). This is in part due to factors such as antialiased edges, non-rigid motion, occluded surfaces or time-varying shading, which increase the number of pixels in a given frame without a suitable cluster in the previous frame.

To avoid this, we take a conservative approach and only assign a definite cluster index to those pixels above a *high confidence* threshold. We again leverage the extra information in the temporal domain and postpone the judgment of the unresolved pixels to subsequent processing (Section 3.6). To define our high confidence threshold, we analyze two conditions: We first check whether the posterior of $\bar{k}$ satisfies $\mathcal{P}(C_{\bar{k}}|p) > \varepsilon$. We empirically set the threshold $\varepsilon = 0.8$ (the maximum value of $\mathcal{P}$ is 1.0). Additionally, we ask how close this probability is to the probability associated to the second-best cluster $C_{\bar{k}'}$, by checking whether $\mathcal{P}(C_{\bar{k}}|p) > \beta\mathcal{P}(C_{\bar{k}'}|p)$. We fix $\beta = 2$, which we found to work well for all the results shown in the chapter and the Supplementary Video[1]

If these two conditions are satisfied, pixel p is assigned to a cluster, and its reflectance value is set to $\mathbf{r}_{\bar{k},l}$. In this fashion, we successfully assign about 95% of the pixels for each frame. The reflectance of the rest of the pixels is left undefined, and will be assigned later using information from other frames. Figure 3.5 (f) shows the result after removing the pixels with low-confidence reflectance.

1 http://webdiis.unizar.es/~elenag/projects/SIG2014_intrinsicvideo/

3.5.3 *Stopping condition*

We continue propagating reflectance values on successive frames $f_{t+1}, f_{t+2}, \cdots$, following the same approach. Unassigned pixels in f_t remain undefined in subsequent frames, so the total number of such pixels increases as the propagation continues forward. As a consequence, applying our propagation algorithm over long stretches in the video sequence may lead to poor results, with too many unassigned pixels in the end.

To bound the number of unassigned pixels, we define a stopping condition for our forward reflectance propagation, and stop when a given threshold is reached. Instead of relying on the total number of unassigned pixels over the whole image, we take a local approach and stop when the ratio of unassigned pixels is greater than $\alpha = 80\%$ in any local $M \times M$ window W_M on the current frame. This allows for timely detection of growing *regions* of unassigned pixels, which would be too difficult to solve later if a global threshold had not been reached yet. We begin with a value of $M = 30$, which will be progressively reduced in subsequent iterations (see Subsection 3.6.1). Figure 3.6 (a) shows the result of the forward propagation once the stopping condition has been met. The remaining undefined pixels of all the processed frames are left to later processing, explained in the following sections.

3.6 REFLECTANCE COMPLETION

Once the propagation has stopped at frame f_s, we perform a *local* intrinsic decomposition on the sparse set of unknown pixels in that frame. We take advantage of the propagated reflectance values, and add them as constraints into our framework. We first define a mask Ω^0 containing all the unknown pixels of frame f_s, and perform a morphological expansion with a radius of four pixels, so that the new expanded mask Ω^* now also contains known reflectance values (see Figure 3.6 (b)). We denote $\Omega^1 = \{\Omega^* - \Omega^0\}$ as the set of pixels with known reflectance, which are included as constraints adding a new term $\lambda_c E_c$ to Equation 4.4 (see the Appendix), where:

$$E_c = \sum_{p \in \Omega^1} (r_p - r_p^1)^2 \tag{3.8}$$

where r_p and r_p^1 are the log-reflectance values of the unknown and known pixels respectively. We set $\lambda_c = 1000$.

To avoid discontinuities introduced by the new reflectance values (see Figure 3.6 (c)), we apply Poisson blending [146] with Dirichlet boundary conditions on the arbitrary outlines of the regions:

$$\Delta R_c = \Delta R^1 \ over\ \Omega^0,\ with\ R_c|_{\partial\Omega^0} = R^1|_{\partial\Omega^0} \tag{3.9}$$

where Δ is the Laplacian operator, and R^1 and R_c are the known and the newly obtained (completed) reflectance values on Ω^0, respectively. Figure 3.6 (d) shows the final result.

CLUSTERING UPDATE We now need to find matching clusters C_k for the pixels with new reflectance values. We rely on table $\mathcal{R}_k$ since the variation of pixel values in reflectance space is smaller than in the original color space.

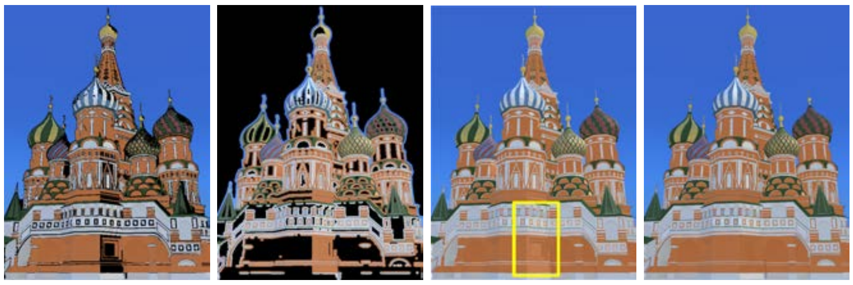


Figure 3.6: *Reflectance completion: (a) Forward reflectance stops at frame $f_s = f_{31}$. Unassigned pixels are depicted in black. (b) Partial, constrained reflectance completion over Ω^*. (c) Simply combining the forward propagation reflectance with the partial completed reflectance causes reflectance discontinuities (see for instance the yellow rectangle). (d) The result of our Poisson reflectance smoothing.*

Similar to Equation 3.6, we calculate the likelihood of pixel p belonging to cluster C_k by maximizing:

$$\mathcal{P}(p|C_k) = \frac{1}{|U(\mathcal{R}_k)|} \sum_{\mathbf{r}} U(\rho(k, \mathbf{r}))G\left(\frac{\mathbf{r} - \mathbf{r}_{p,s}}{d}\right) \tag{3.10}$$

where $\mathbf{r}_{p,s}$ is the reflectance of pixel p at frame f_s, and $|U(\mathcal{R}_k)|$ is the number of non-zero entries of $\mathcal{R}_k$ for cluster C_k. Note that in this case we cannot rely on our smoothness prior in Equation 3.7, since nearby pixels in the previous frame will likely be unassigned also. We thus do not limit the search to local windows, but search all clusters instead. Additionally, since the completion works robustly, we do not need to enforce the conservative constraints in Section 3.5.2 before assigning a cluster. Once the index $\bar{k}$ with maximum probability has been found, we update the corresponding $\mathcal{I}_{\bar{k}}$, $\mathcal{R}_{\bar{k}}$ tables to ensure an effective subsequent propagation.

Note that thanks to this updating process, our reflectance propagation of frame k is not entirely determined by its adjacent frame only. In our probability formulation (Equation 3.3), the first term (Equation 3.6) is based on the global histogram $\mathcal{I}_k$, which is constantly updated based on information from all the frames that have already been traversed. The smoothness term in Equation 3.7 does depend only on the previous frame.

3.6.1 *Coarse-to-Fine Propagation*

Our next steps complete the remaining unassigned pixels, following a coarse-to-fine approach. From the (partially) completed f_s we first launch a backward propagation towards f_t. We follow the same basic approach as the forward propagation, based on the color table $\mathcal{I}_k$ and using again our Bayesian framework defined in Equation 3.3. Note that at each step, we only propagate the reflectance to pixels that have not been assigned yet, leaving the rest unchanged. Intuitively, this backward propagation helps assign correct reflectance values to occluded objects in the forward propagation: working backwards in time, these are visible from the beginning.

This forward-backward propagation scheme is then iterated to complete the rest of the unassigned pixels. At each iteration, we decrease the threshold for the stopping condition: specifically, we maintain the threshold ratio $\alpha = 80\%$, but progressively reduce the size of the window W_M. We adopt a three-pass iteration scheme, using $M = 30, 20$, and 10 respectively. This offers a good trade-off to complete unassigned pixels effectively without need-

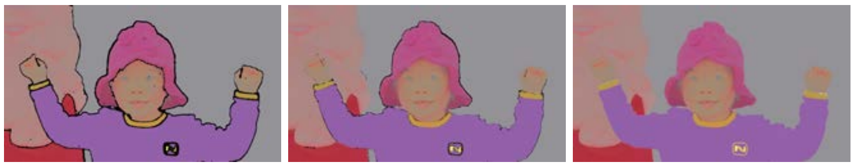


Figure 3.7: *Results of the iterative propagation and final completion. Left: Unassigned (black) pixels after the first iteration. Middle: Reduced number of unassigned pixels after the three iterations. Right: Final result after reflectance completion.*

ing to run our reflectance completion algorithm at each frame. Figure 3.7 (left and middle) shows an example.

RESIDUAL REFLECTANCE COMPLETION While we could continue the iteration until all pixels are assigned a reflectance value, this may be impractical or error-prone in some difficult cases. Instead, we rely on Retinex theory and leverage *shading* information. We assume C^1 shading continuity on the unassigned regions, an assumption that has been used in previous Retinex-based works [51, 164, 59]; in our case this works particularly well given the small size of the remaining unassigned areas (less than 1% in our experiments).

Similar to our reflectance completion step, we impose smooth interpolation on the domain Ω^0 of unassigned pixels by applying the following Poisson equation:

$$\Delta S_c = 0 \; over \; \Omega^0, \; with \; S_c|_{\partial\Omega^0} = S^1|_{\partial\Omega^0} \tag{3.11}$$

where S^1 and S_c are the known and the new completed shading values respectively. The final reflectance values are simply obtained by a pixel-wise division of color over shading (Figure 3.7, right).

3.7 EVALUATION

We first evaluate our algorithm by comparing it to a ground truth, synthetic example. We have rendered a 3D model of *St. Basil* with dynamic lighting and a rotating camera, using *Maya*. The ground truth reflectance and shading layers are obtained assigning a constant and a white Lambertian shader, respectively. Figure 3.8 (left), shows the comparison between the rendered ground truth sequence and our algorithm; the entire animation is included in the video. Figure 3.8 (top-right) shows a representative graph of the evolution of the Local Mean Squared Error (LMSE) with the number of initial reflectance clusters. We have found that the error is consistently minimized with 5-10 clusters in all our sequences. With too few clusters, the algorithm cannot differentiate reflectances properly, whereas a larger number increases error since the second condition in our high confidence threshold (Section 3.5.2) can hardly be met. Moreover, in Figure 3.8 (bottom-right) we compare our reflectance results for *chicken* against its chromaticity channels. Our reflectance represents the underlying materials more truthfully, while being able to separate black features (no chromaticity) like the pupils. Figure 3.9 shows some representative clusters of the scenes included in this work.

Although ours is the first intrinsic decomposition method devised to work with video sequences, we objectively compare against four other alternatives, by plotting the LMSE with respect to the ground truth for the *St.*

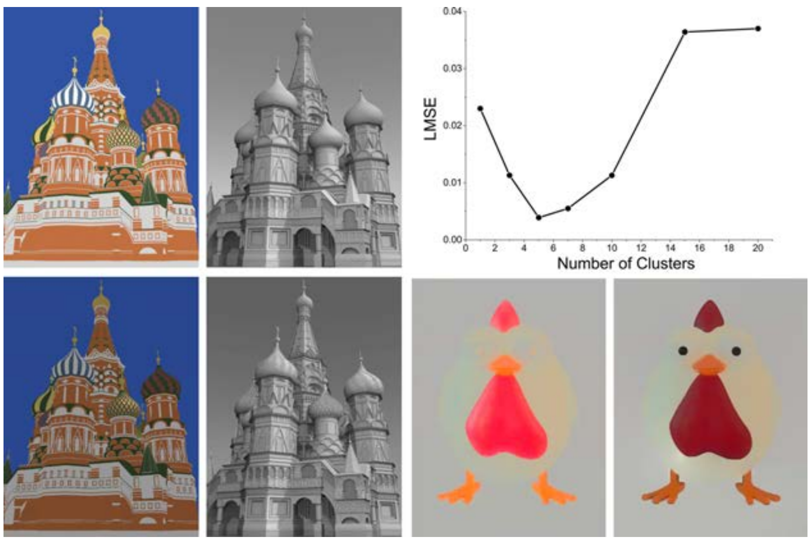


Figure 3.8: *Left: Comparison between synthetic ground truth for reflectance and shading (top) and our corresponding results (bottom). Top-right: The evolution of the Local Mean Squared Error (LMSE) with the number of initial reflectance clusters. Bottom-right: Comparison between chrominance (left) and our reflectance layer (right). Note how the reflectance of the materials is much better depicted with our method.*

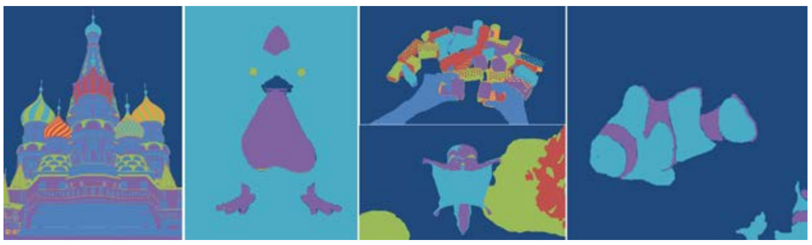


Figure 3.9: *Representative clusters of the videos included in this work. The parameters of the segmentation are the same for all the sequences (details in Section 6.5).*

Basil sequence. In particular, we first compare against two state-of-the-art methods devised for single images [197, 57], decomposing each frame of the video individually. Figure 3.11, left (black and blue lines) shows the results. Because of motion-induced occlusions and time varying shading, the results of a frame-by-frame approach are inevitably unstable, with large, sudden changes between frames. These spikes in error translate into disturbing flickering artifacts even for the best of the two algorithms, as the accompanying video shows. Our method yields very stable results, with the lowest LMSE (red line).

The green line (also in Figure 3.11, left) shows the result of applying the recent method by Lang et al. [103] over the frames returned by Zhao's method. The resulting error curve is somewhat smoother than frames processed individually, indicating overall improved temporal stability (although some of the largest error peaks remain). However, the overall error increases compared to per-frame decomposition by Zhao's method. The reason is that Lang's method is not well suited for the particular case of intrinsic images since, as we have seen, this initial per-frame decomposition shows very large differences between frames, which in turn forces a very aggressive smoothing in the temporal domain. The spatial smoothing that the method imposes along with its temporal filtering, leads to clear ghosting artifacts and over-

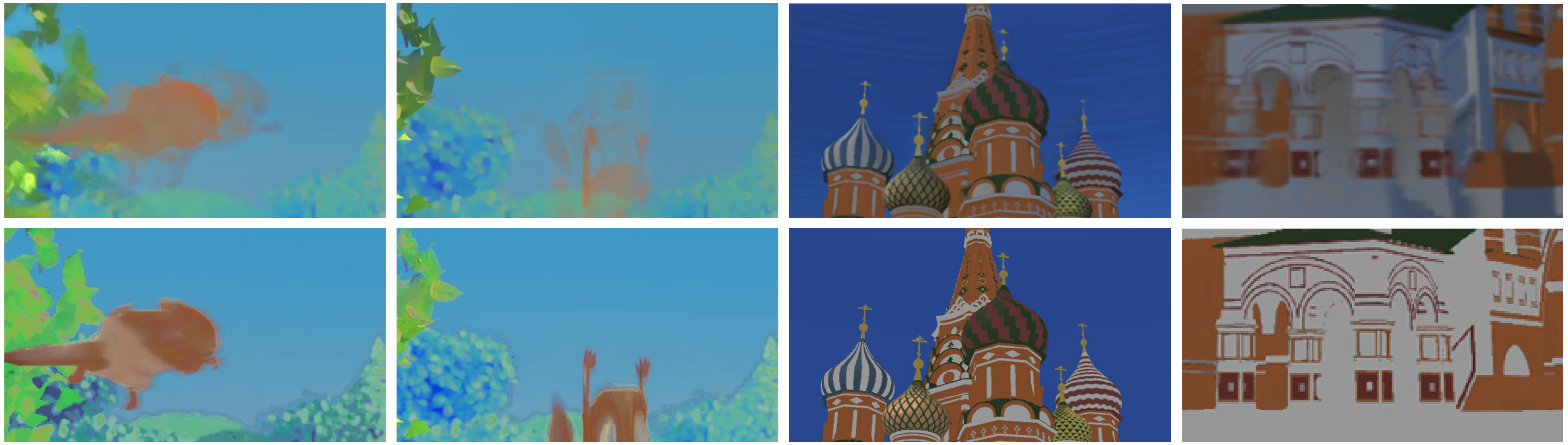


Figure 3.10: *Comparison of Lang's method [103], imposing temporal coherency to the decomposition of each individual frame, (top) and our result (bottom) for the squirrel and St. Basil sequences. The built-in spatial filtering in Lang's method, coupled with the large variability across frames of per-frame decompositions, cause some clear ghosting artifacts, more exaggerated in the presence of quick motion (squirrel). Additionally, fine features become blurry (see the rightmost comparison for St. Basil).*

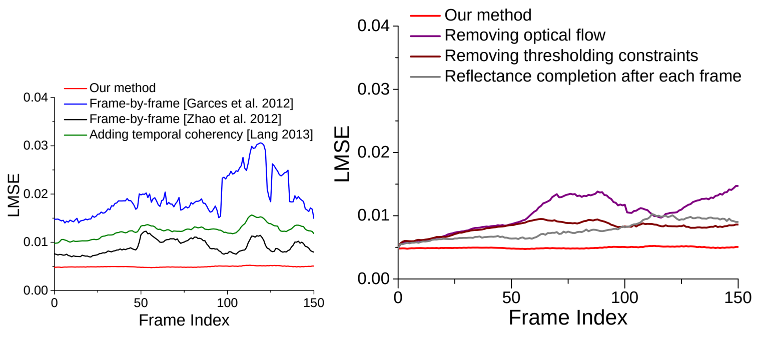


Figure 3.11: *Left: Comparison of LMSE of our algorithm (red line) and three other different approaches, including two per frame decompositions [197, 57] and the recent method by Lang et al [103] which enforces temporal stability over Zhao's per frame result. Our algorithm yields the most stable results, avoiding flickering, and the lowest LMSE. Right: Different variations of our algorithm; removing or altering some of its key components leads to temporal instability and larger error.*

blurred edges. Figure 3.10 shows some direct comparisons with our method for two different sequences.

Figure 3.11, right, shows the LMSE of different variations of our algorithm to highlight the influence of its most relevant components. It can be seen how performing reflectance completion after each frame (gray line), removing our conservative high-confidence threshold defined in Section 3.5.2 (deep red) or removing optical flow (purple line) increase both the overall error and the temporal instability of the results.

3.8 results and applications

Figure 3.12 shows additional results (we refer the reader to the accompanying video for the complete set). All our results have been produced automatically, unless user scribbles are shown on the initial frame. Their frame length varies between 100 and 500 frames, which is in accordance to the average shot in modern TV and movies [103]. The initial decomposition runs at interactive rates. Since we only propagate frames until the local error threshold is surpassed, we have experienced no error accumulation or temporal drift in any of the videos tested. Note also how in the *chicken* sequence parts of the body disappear and appear again, which our algorithm handles gracefully thanks to its forward-backward structure. In *objects*, the camera loops around the scene; notice the temporal coherence of the reflectance images, including the first and last frames of the sequence (leftmost and rightmost frames in the image). In the video, *Squirrel* presents a particularly challenging case including a furry animal, moving shadows and motion blur, but our algorithm still yields good results.

For one frame in a video sequence at 800 $\times$ 600 spatial resolution, our algorithm takes slightly over one minute using a four-threaded unoptimized implementation on a standard PC. Automatic decomposition and reflectance clustering of the first frame requires about 30 seconds, plus 20 seconds for the forward propagation in the first iteration and 5 seconds for each reflectance completion and residual reflectance completion steps. Since the propagation for each pixel is neighbor-independent, the algorithm is suitable for parallel processing and could be significantly accelerated on a GPU.

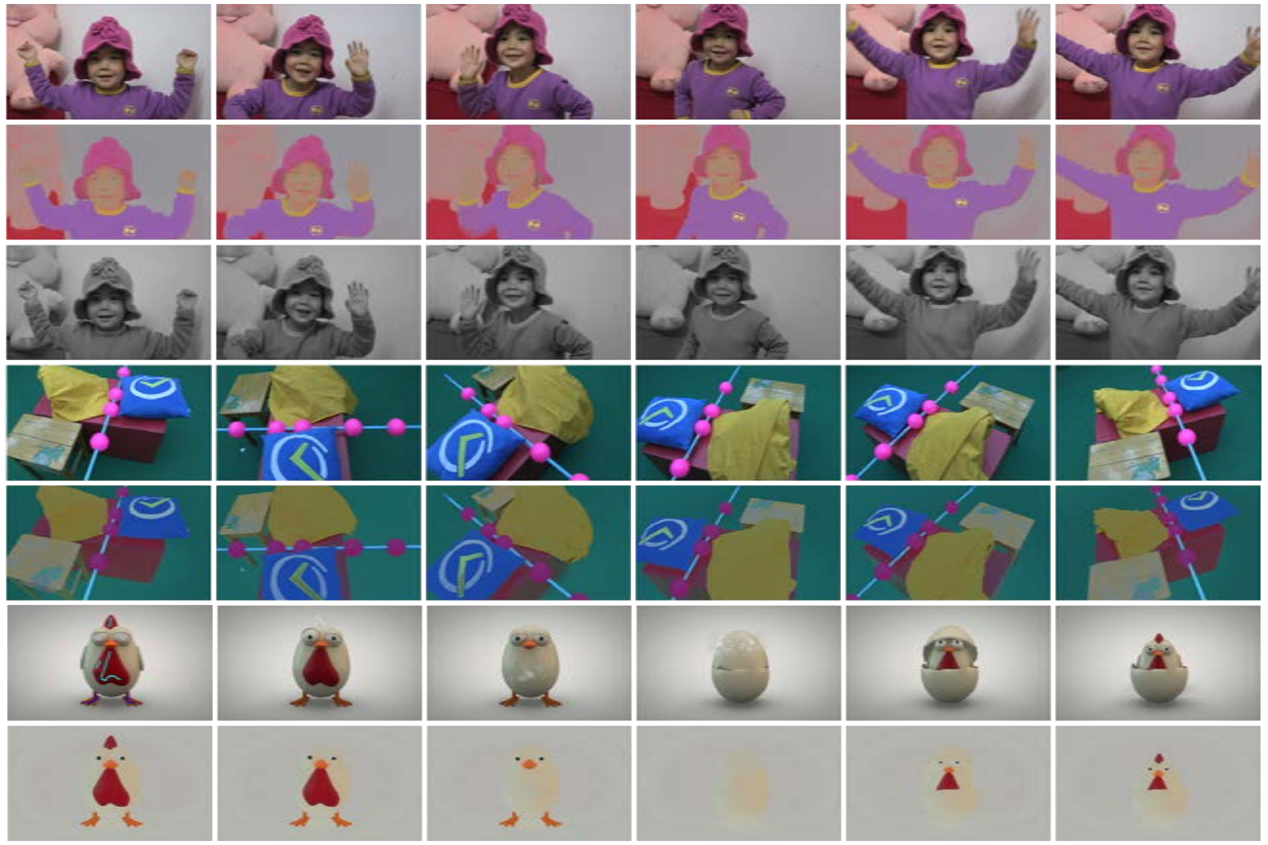


Figure 3.12: *Results of our intrinsic video decomposition for the dancing baby, objects and chicken sequences (including strokes for chicken for constant-shading regions in gray and for constant-albedo in color). Please refer to the video for these and more complete examples.*

Moreover, our intrinsic video decomposition method can be used as a basic platform for video editing applications. Here we show some examples, and refer the reader again to the Supplementary Material[2].

Video segmentation Available video segmentation methods rely heavily on local color and shape information; therefore such methods usually fail to track an object in natural videos where illumination or shape may change. Our algorithm can be used for object segmentation based on propagated *clustered reflectance* information. Since the reflectance propagation effectively removes the shading component, it is more robust in case of changes in illumination. Moreover, our method does not rely on local statistics, which makes it also robust in case of fast topology changes of the segmented objects. Figure 3.14 shows the results for five frames of the *nemo* sequence, equally spaced 225 frames apart, compared with the result using the popular video SnapCut [7] and the online efficient hierarchical graph-based video segmentation (EHGV) tool [64]. For a fair comparison with these methods, in the SnapCut example the first frame has been manually labeled as foreground and background, in order to provide visually comparable results, while for EHGV we use the automatic cluster result directly. It can be seen how both SnapCut and EHGV progressively accumulate error and completely miss at least one of the fishes in the end. In contrast, our method improves performance in the presence of fast motion, dramatic changes in shape, and similar color between foreground and background objects, even over a large number of frames, without error drift.

Material editing We can re-render the surface materials by manipulating the shading layer, defining a simple mapping function between the original and the new shading. The user only needs to define a sparse set of control point in the shading grayscale space, and the mapping function is obtained by cubic interpolation. Figure 3.13, top-left, shows how diffuse surfaces can be made to appear shiny by applying the function shown.

Color transfer Given a source image of different scene, our intrinsic decomposition allows us to efficiently transfer its color and tone to a target video, by simply performing histogram matching on the reflectance layers of both the source image and the first frame of the video. Our algorithm efficiently propagates this new reflectance information; multiplying by the corresponding shading images yields the final color-transferred video. Given our temporally consistent cluster information, we can further composite the original foreground object onto the new background (Figure 3.13, bottom-left).

Recolorization By adding simple scribbles on the input image (Figure 3.13, top-right), users can define a reflectance transfer function between the input and the target reflectance values. In our implementation, the target reflectance values are defined by the color of the scribbles. For each pixel in any subsequent frame, this transfer function is applied to the original reflectance. Note that similar effects have been achieved using propagation techniques [110, 185], although here we directly manipulate the reflectance layer.

Stylization We can easily achieve interesting non-photorealistic results by manipulating each intrinsic layer separately. Figure 3.13, bottom-right, shows two different depictions of the same video, increasing and decreasing the saturation of the reflectance layer, respectively, while flattening the shading layer. An edge layer has been subsequently added to enhance the effect.

2 http://webdiis.unizar.es/~elenag/projects/SIG2014_intrinsicvideo/

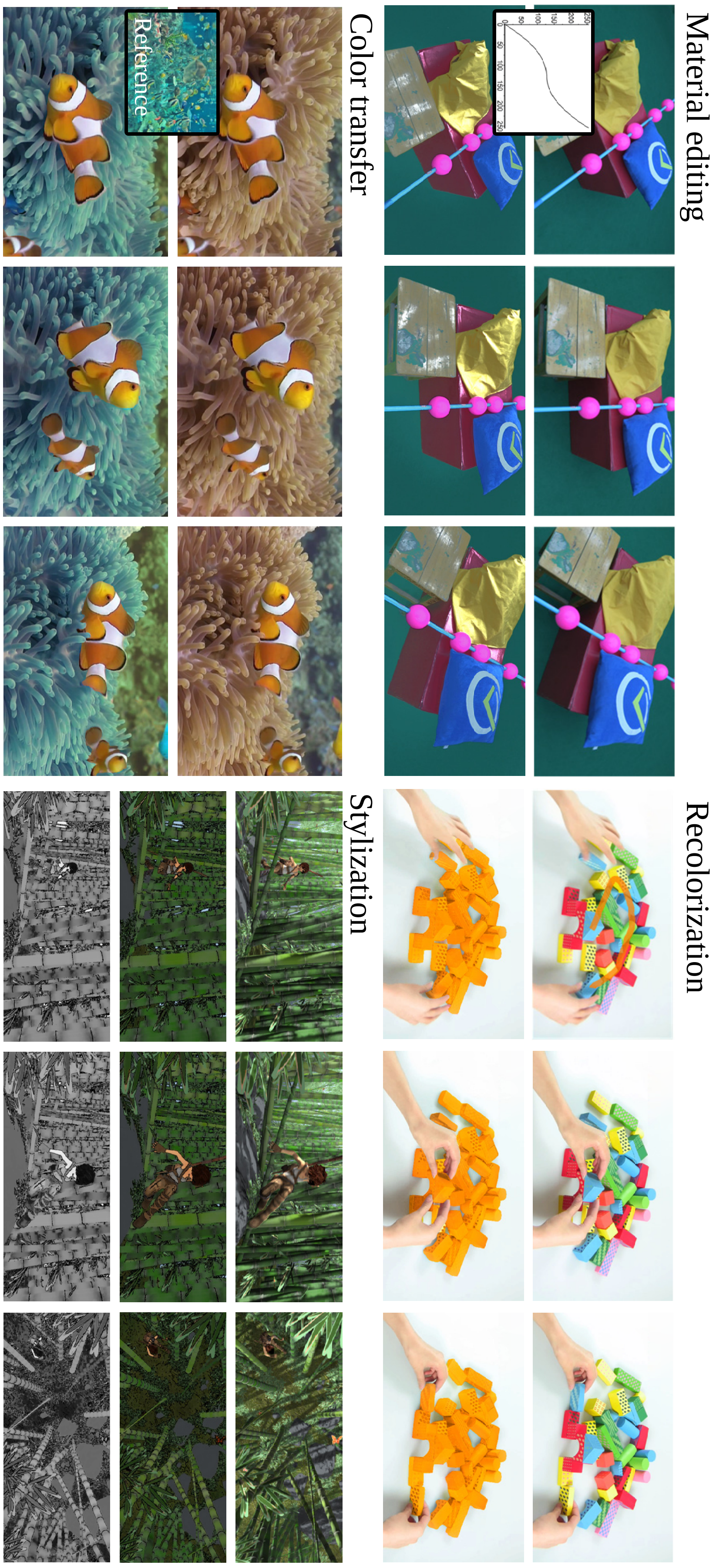


Figure 3.13: *More example applications of our intrinsic video decomposition. The two small insets show the function mapping input and output shading (material editing), and the source image (color transfer). In the recolorization example, the target reflectance value is defined by the color of the scribble.*

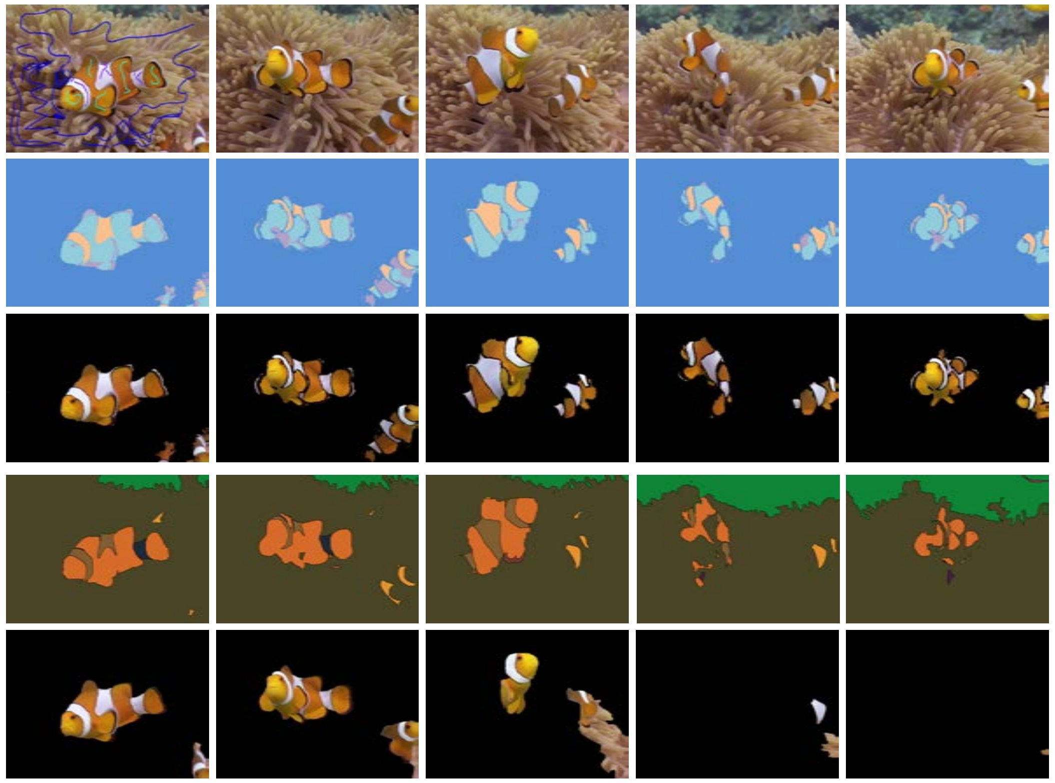


Figure 3.14: *Example and comparison of automatic video segmentation. First row: Input image frames (first column input strokes for constant-albedo regions). Second row: Our automatic clustered reflectance. Third row: Our result labeling foreground/background. Fourth row: Result of automatic EHGV segmentation [64]. Fifth row: Result of using SnapCut [7] labeling foreground/background.*

3.9 DISCUSSION

In summary, we have presented a novel approach for the challenging problem of intrinsic video decomposition, as well as several example applications in video editing. Our method is temporally coherent, does not impose a large memory footprint, and does not require heavy user interaction. We believe this is the first work successfully addressing this problem, without imposing any restrictions on the input videos.

Our approach is not free of limitations. Working on a per-frame basis allows our method to be very light on memory requirements and processing time; however it is currently not optimized for speed, and it still does not work in real time. Similar to other video editing approaches, different shots need to be processed separately: if the content between shots varies drastically (i.e., two completely different scenes in a movie), our propagation scheme cannot guarantee good results. Last, if the initial frame is very dark or very saturated, it may be hard to find meaningful reflectance clusters. Despite this, we hope our work inspires both future research on intrinsic decomposition of video sequences, as well as novel video editing techniques that take advantage of the individual manipulation of reflectance and shading information.

APPENDICES

3.A RETINEX-BASED OPTIMIZATION

We summarize here the main aspects of the paper by Zhao and colleagues [197] relevant to our work. The problem of intrinsic decomposition of a single image is posed as an optimization. Working in log-space, the problem is defined as $i_p = s_p + r_p$, where $i_p = \log(I_p), r_p = \log(R_p)$ and $s_p = \log(S_p)$ (I_p, R_p and S_p are the image, reflectance and shading pixel values, according to our Equation 3.1). The following function is then minimized:

$$\arg\min_s E(s) = \lambda_l E_l(s) + \lambda_r E_r(s) + \lambda_a E_a(s) \tag{3.12}$$

where λ_l, λ_r and λ_a are positive weights (set to 1, 10000 and 1000 respectively), $E_l(s)$ represents the common Retinex constraint minimizing the differences in shading and reflectance between adjacent pixels, $E_r(s)$ is a non-local albedo constraint and $E_a(s)$ is a normalization factor. Explicitly:

$$E_l(s) = \sum_{(p,q)\in N} \left[(s_p - s_q)^2 + w_{(p,q)}(r_p - r_q)^2\right] \tag{3.13}$$

where N denotes the set of all neighboring pairs of pixels and $w_{(p,q)}$ is the balance factor introduced in Equation 3.2. The next term is:

$$E_r(s) = \sum_{G_r^i \in \Gamma_r} \sum_{(p,q)\in G_r^i} \left(r_p - r_q\right)^2 \tag{3.14}$$

where G_r^i is a set of pixels with similar albedo, and Γ_r is the set of pixel groups previously detected to share the same albedo. Last, the normalization term is:

$$E_a(s) = \sum_{p\in G_a} (s_p - 1)^2 \tag{3.15}$$

where G_a contains the brightest pixel(s).

4 INTRINSIC LIGHT FIELD DECOMPOSITION

In this chapter, we present the first method to automatically decompose a *light field* into its intrinsic shading and albedo components. Contrary to previous work targeted to 2D single images and videos, a light field is a 4D structure that captures non-integrated incoming radiance over a discrete angular domain. This higher dimensionality of the problem renders previous state-of-the-art algorithms impractical either due to their cost of processing a single 2D slice, or their inability to enforce proper coherence in additional dimensions. We propose a new decomposition algorithm that jointly optimizes the whole light field data for proper angular coherency. For efficiency, we extend Retinex theory, working on the gradient domain where new albedo and occlusion terms are introduced. Results show our method provides 4D intrinsic decompositions difficult to achieve with previous state-of-the-art algorithms.

E. Garces, J. I. Echevarria, W. Zhang, H. Wu, K. Zhou & D. Gutierrez
INTRINSIC LIGHT FIELDS
eprint arXiv:1608.04342, 2016.

4.1 INTRODUCTION

Intrinsic scene decomposition is the problem of separating the integrated radiance from a captured scene, into a physically-based and more meaningful reflectance and shading components, so that *Scene = Albedo* $\times$ *Shading*; enabling quick and intuitive edits of the materials or lighting of a scene.

However, this decomposition is a very challenging, ill-posed problem. Given the interplay between the illumination, geometry and materials of the scene, there are more unknowns than equations for each pixel of the captured scene. To mitigate this uncertainty, existing *intrinsic decomposition* works assume that some additional properties of the scene are known. However, the prevailing goal is always the same: the gradients of the depicted scene need to be classified as coming from a variation in albedo, shading, or both. In this work, we build on classical theories of Retinex to obtain better predictors of these variations leveraging information from the light field data.

On the other hand, light field photography is becoming more popular, as multi-view capabilities are progressively introduced in commercial cameras [121, 149], including mobile devices [178]. Such captured light fields are 4D structures that store both spatial and angular information of the radiance that reach the sensor of the camera. This means a correct intrinsic decomposition has to be coherent in the angular domain, which increases the complexity with respect to 2D single images and 3D videos (x, y, t). Not only because of the number of additional information to be processed, but also because of the kind of coherence required. To the best of our knowledge, no intrinsic decomposition method exists suited to handle these particularities.

A naïve solution to intrinsic light field decomposition would be to apply any state-of-the-art single image algorithm to each view of the light field independently. However, due to the extensive computational time that these algorithms require (Bell et al. [13] takes around 10 minutes for a 400×400 image), this is not an option given a typical light field contains around 9×9 views. And even then, angular coherence across views would not be guaranteed. Another approach could be to extend intrinsic video decompositions to 4D light field volumes, as these techniques rely on providing an initial solution for a 2D frame (usually the first), which is then propagated along the temporal dimension. However, the propagation mechanism is designed to ensure smooth temporal transitions in a single dimension, assuming a normal playback of the video. While the 4D structure of a light field would allow its different views to be "stacked" and processed as a single video sequence, its angular dimensions cannot be assumed to be explored in a single way. Thus, all possible sequences should be taken into account, making this approach highly inefficient.

Therefore, we propose a new decomposition algorithm that jointly optimizes for the whole light field data efficiently, is scalable, and maintains proper angular coherency. With this work, our goal is not to obtain the best decomposition for each single view, but the most coherent one, in a practical way. This not only keeps adding to the limited set of tools for light field editing [80, 194], but paves the way for other applications that require robust segmentation or selection of the same object and areas across all the different views. Our algorithm has the following characteristics:

- It is devised with 4D light field data in mind, enforcing angular coherence in the results.
- It is computationally efficient, while working with the whole 4D data volume.
- It builds on existing Retinex approaches, extending them with additional terms on the formulation.

We show results both with synthetic light fields (with ground truth references for reflectance, shading and depth), and real world light fields captured with a Lytro camera. We show how our method outperforms previous approaches when used with light field data, and is robust even in the presence of imperfect depth information.

4.2 RELATED WORK

Intrinsic decomposition of the shading and albedo components of an image is a long-standing problem in computer vision and graphics since it was formulated by Barrow and Tenembaum in the 70s [10]. We review previous intrinsic decomposition algorithms based on their input, and then briefly cover related light field processing.

SINGLE IMAGE. Several works rely on the original Retinex theory [102] to estimate the *shading* component. By assuming that shading varies smoothly, either a pixel-wise [173, 197] or cluster-based [57] optimization is performed. Clustering strategies have also been used to obtain the *reflectance* component, e.g. assuming a sparse number of reflectances [59, 164], using a dictionary of learned reflectances from crowd-sourcing experiments [13], or flattening the image to remove shading variations [17]. Alternative methods require user interaction [22], jointly optimize the shape, albedo and illumination [8], incorporate priors from data driven statistics [198], train a Convolutional Neural Network (CNN) with synthetic datasets [133], or use depth maps acquired with a depth camera to help disambiguate shading from reflectance [9, 28, 104]. For efficiency, our work is also based on the Retinex theory, with 2D and 4D scene-based heuristics to classify reflectance gradients.

multiple images and video. Several works leverage information from multiple images of the same scene from a fixed viewpoint under varying illumination [182, 70, 98, 168]. Laffont et al. [97] coarsely estimate a 3D point cloud of the scene from non-structured image collections. Pixels with similar chromaticity and orientation in the point cloud will be used as reflectance constraints within an optimization. Assuming outdoor environments, the work of Duchene et al. [38] estimates sunlight position and orientation and reconstructs a 3D model of the scene, taking as input several captures of the same scene under constant illumination. Although a light field can be seen as a structured collection of images, we avoid the additional work required to build and process such proxy 3D models, by directly leveraging the 4D structure for a more effective and efficient approach.

video. A few methods dealing with intrinsic video have been recently presented. Ye et al. [191] propose a probabilistic solution based a casual-anticasual, coarse-to-fine iterative reflectance propagation. Bonneel et al. [21] present an efficient gradient-based solver which allows interactive decompositions. Kong et al. [94] rely on optical flow to estimate surface boundaries to guide the decomposition. Recently, Meka et al. [127] present a novel variational approach suitable for real-time processing, based on a hierarchical coarse-to-fine optimization. While these methods work in a 3D domain, they are devised to keep a smooth coherency in the temporal dimension, which is assumed to be navigated in a continuous way. 4D Light fields, however, need to be consistent in all angular dimensions, without any assumption about the way they are going to be explored.

light field editing. Our work is also related to papers that extend common tools and operations for 2D images to 4D light fields. This is not a trivial task, given again the higher dimensionality of light fields. Jarabo et al. [80] present a first study to evaluate different light field editing interfaces, tools and workflows, this study is further analyzed by Masia et al. [122], providing a detailed description of subjects' performance and preferences for a number of different editing tasks. Global propagation of user strokes has also been proposed, using a a voxel-based representation [155], a multi-dimensional downsampling approach [81], or preserving view coherence by reparameterizing the light field [2], while other works focus on deformations and warping of the light field data [18, 27, 196]. Cho et al. [31] utilize the epipolar plane image to extract consistent alpha mattes of a light field. Guo et al. [65] stitch multiple light fields via multi-resolution, high dimensional graph cuts. There are also considerable interests in recovering depths from a light field. Existing techniques exploit defocus and correspondence depth cues [172], carefully handle occlusions [179], or use variational methods [180]. As most of these works, we also rely on the epipolar plane image for implicit and efficient multi-view correspondences and processing.

4.3 FORMULATION

To represent a light field, we use the two-plane parametrization on ray space $L(x, y, q, t)$, which captures a light ray passing through two parallel planes: the sensor plane Π_{qt}, and the virtual camera plane or image plane Ω_{xy}. Analogous to its 2D image counterpart, the problem of intrinsic light field decomposition can be formulated as follows: for each ray of the light field L, we aim to find its corresponding reflectance and shading components R and S, respectively.

$$L(x, y, q, t) = R(x, y, q, t) \times S(x, y, q, t) \tag{4.1}$$

Instead of solving for single rays directly, the problem can be formulated in the gradient domain for the image plane Ω_{xy}:

$$\nabla L(x, y, q^*, t^*) = \nabla R(x, y, q^*, t^*) + \nabla S(x, y, q^*, t^*) \tag{4.2}$$

more compactly $\nabla l = \nabla r + \nabla s$. Where l, r and s denote the single views for each $\{q^*, t^*\} \in \Pi_{qt}$ for each input view l, its reflectance r and shading s in log spaces. Note that we denote single views computed in log domain with lowercase, while uppercase letters denote the whole light field in the original domain.

The classic Retinex approach [102] proposes a solution to this formulation by classifying each gradient as either shading or albedo. As seen before, different heuristics have been proposed over the years, with the simplest one associating changes in albedo with changes in *chromaticity*. Although this provides compelling results for some scenes, it still has the following limitations: chromatic changes do not always correspond to albedo changes; the solution is very sensitive to high frequency texture; and more importantly it does not take into account the effects of occlusion boundaries, where shading and albedo vary at the same time.

4.4 OUR METHOD

4.4.1 *Overview*

Our approach to the problem of intrinsic light field decomposition is based on a multi-level solution detailed in Algorithm 2: In a first step, we perform a global 4-dimensional l_1 filtering operation, which generates a new version of the light field with reduced high frequency textures and noise, to promote relevant gradients and edges, as well as improved angular coherency. The resulting light field, which we call $\hat{L}$, will serve to initialize a first estimation of the reflectance R_0 and shading components S_0 (Section 4.4.2). These initial estimations will then be used to compute the albedo and occlusion cues needed for the actual intrinsic decomposition, which is done locally per view (Sections 4.4.3.1 and 4.4.4), benefiting from the previous global processing of the whole light field volume. A final global 4D l_1 filtering operation (Section 4.4.5) performed over the reflectance finishes promoting angular coherency and stability, as can be seen in the results section and the Supplementary Material.

4.4.2 *Initialization*

Inspired by the work of Bi et al. [17], we noticed that better predictions of the albedo discontinuities can be done by performing an initial l_1 filtering of

Algorithm 2 Intrinsic Light Field Decomposition

```
1: Input: Light field $L(x,y,q,t)$
2:                                    ▷ Initialization (Section 4.4.2)
3: $\hat{L} \leftarrow \mathrm{TV}L_1(L, \beta = 0.05)$
4: $S_0 \leftarrow ||\hat{L}||_2$
5: $R_0 \leftarrow \hat{L}/S_0$
6:                      ▷ Global Analysis (Sections 4.4.3.1 and 4.4.4)
7: $\omega_a \leftarrow \mathrm{getAlbedoTh}(\hat{L}, R_0)$
8: $\omega_{occ} \leftarrow \mathrm{getOcclusionGradient}(L_{depth})$
9:                                    ▷ Local intrinsic decomposition
10: $R_{g1}, S_1 \leftarrow \mathcal{G}(\hat{L}, \omega_a, \omega_{occc})$ ▷ Note that $R_{g1}$ and $S_1$ are both single channel
11:                                   ▷ Global coherency (Section 4.4.5)
12: $\hat{R_1} \leftarrow \mathrm{TV}L_1(R_{g1}, \beta = 0.05)$
13: $S_f \leftarrow ||\hat{L}||_2/\hat{R_1}$
14: $R_f \leftarrow L/S_f$
15: Result: $R = R_f(x,y,q,t)$, $S = S_f(x,y,q,t)$
```

the light field volume, since it enhances edges and removes noise that could introduce errors in the estimation of gradients. In particular, we regularize the total variation (TV-l_1):

$$\min_{\hat{L}} \frac{1}{2}||\hat{L} - L||_2^2 + \beta||\hat{L}||_1 \tag{4.3}$$

As a result, from the original light field L, we obtain a filtered version $\hat{L}$, close to the original input but with sharper edges due to the use of l_1 norm on the second term. Additionally, the use of this norm effectively removes noise while prevents smoothing out other important features. The regularization factor β controls the degree of smoothing, where in our experiments $\beta = 0.05$.

Working with light fields means that we need to solve this multidimensional total variation problem in $4D$. Since efficiency is key for our method to be practical, we use the ADMM solver proposed by Yang et al. [189]. ADMM combines the benefits of augmented Lagrangian and dual decomposition methods. It decomposes the original large global problem into a set of independent and small problems, which can be solved exactly and efficiently in parallel. Then it coordinates the local solutions to compute the globally optimal solution.

Figure 4.1, shows the difference in angular coherency and noise between the input L, a filtered version obtained from processing each single view independently, and our $\hat{L}$ obtained from the described global filtering. From $\hat{L}$, we compute the initial shading as, $S_0 = ||\hat{L}||_2$. This is a convenient step to obtain a single-channel version of the input image, with other common transformations like the RGB average or the luminance channel from CIELab [57] providing similar performance. Taking S_0 as baseline, we compute the initial RGB reflectance R_0 simply from $\hat{L}/S_0$. It is important to note that S_0 and R_0 serve only as the basis over which our heuristics are applied to obtain the final cues to solve for the actual intrinsic decomposition (Equation 4.4). Figure 4.2 shows the impact of this l_1 regularization on the detection of albedo variations.

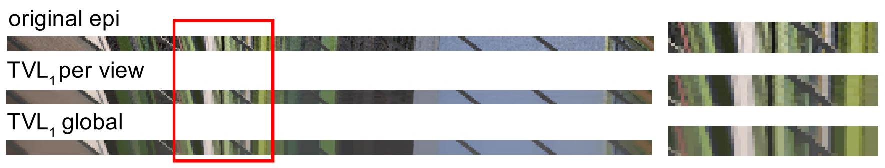


Figure 4.1: *Visualization of the horizonal epi view for the red scanline in Figure 4.2 (a). From top to bottom: the epi from the original light field; the epi after applying TVL_1 filter to each view separately; the same epi after applying a 4D TVL_1 filter to the whole light field volume using our approach. We can observe (by zooming in the digital version), areas with very similar colors are flattened, while sharp discontinuities are preserved, effectively removing noise and promoting angular coherence.*

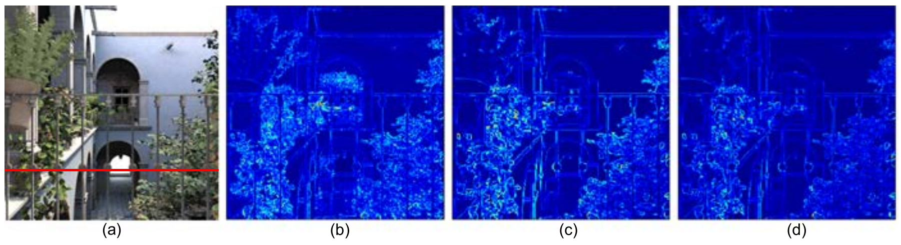


Figure 4.2: *(a) Central view of an input light field. (b) Albedo variations computed as the angle between RGB vectors for neighboring pixels $L^i, L^j \widehat{L^i, L^j}$, from the original light field L. (c) Albedo variations obtained from our initial reflectance estimation, $R^i_0, R^j_0 \widehat{R^i_0, R^j_0}$. (d) Albedo variation from the chromaticity norm, $||\hat{L}^i - \hat{L}^j||$, used by Zhao et al [197]. Our approach (c) yields cleaner gradients than (b), and captures more subtleties than (d). Note for example the green leaves at the right of the image. Every image is normalized to its maximum value.*

4.4.3 *Intrinsic Estimation*

As motivated before, for our efficiency requirements we follow a Retinex approach. We build on Zhao's closed-form formulation, extending it to take into account our albedo and occlusion cues obtained from the 4D light field volume. For each view l of the light field, the system computes the shading component s by minimizing the following equation:

$$\min_s \lambda_1 f_1(s) + \lambda_2 f_2(s) + \lambda_3 f_3(s) \tag{4.4}$$

where f_1 is the Retinex constraint, f_2 is an absolute scale constraint, and f_3 is a non-local texture cue; and λ_1, λ_2, and λ_3 are the weights which control the influence of each term, set to $\lambda_1 = 1$, $\lambda_2 = 1$ and $\lambda_3 = 1000$. In this work we extend f_1, so please refer to the original paper [197] for the full details of f_2 and f_3.

4.4.3.1 *Retinex-Based Constraint*

The original Retinex formulation assumes that while shading varies smoothly, reflectance tends to cause sharp discontinuities, which can be expressed as:

$$f_1(s) = \sum_{i,j \in \mathcal{N}_{xy}} (\nabla s^2_{ij} + \omega^a_{ij} \nabla r^2_{ij}) \tag{4.5}$$

where $\mathcal{N}_{xy}$ is the set of pairs of pixels that can be connected in a four-connected neighborhood defined in the image plane Ω_{xy}, and ω^a_{ij} is commonly defined as a threshold on the variations in the chromatic channels

(Section 4.4.4). Following Equation 4.2, we define the following transformation, needed to solve Equation 4.4.

$$\nabla r = \nabla \hat{l} - \nabla s \tag{4.6}$$

However, we found that this equation ignores the particular case of occlusion boundaries, where shading and reflectance may vary at the same time. In order to handle such cases, we introduce a new additional term ω_{ij}^{occ}, which has a very low value when an occlusion is detected, so it does not penalize the corresponding gradients (more details in Section 4.4.4):

$$f_1(s) = \sum_{i,j \in \mathcal{N}_{xy}} \omega_{ij}^{occ} (\nabla s_{ij}^2 + \omega_{ij}^a \nabla r_{ij}^2) \tag{4.7}$$

We define as $\mathcal{G}$ the function that takes the whole light field and the global cues to obtain the corresponding shading and reflectance layers:

$$\mathcal{G}(\hat{L}, \omega^a, \omega^{occ}) = (S_1, R_{g1}) \tag{4.8}$$

It is important to note that s has a single channel (an interesting future work would be to lift this restriction to allow colored illumination), so Equation 4.6 is also a single channel operation, where $\hat{l}$ is $||\hat{l}||_2$. Therefore, Equation 4.4 yields single channel shading s, and reflectance $r_g = ||l||_2 - s$ in log-spaces. Then, S_1 and R_{g1} are:

$$\forall q, t \in \Pi_{qt} \quad \begin{array}{rcl} S_1(x,y,q,t) & = & e^s \\ R_{g1}(x,y,q,t) & = & e^{r_g} \end{array} \tag{4.9}$$

4.4.4 *Gradient Labeling*

In the following, we describe our extensions to the classic Retinex formulation: the albedo and occlusion terms in Equation 4.7. Note that this labeling is independent from solving the actual system (Equation 4.4), so each cue is computed in the most suitable color space, or additional available dimensions like depth.

4.4.4.1 *Albedo Gradient (ω^a)*

Albedo gradients are usually computed based on the chromatic information in CIELab color space. However, as we have shown, our initial RGB reflectance R_0 is better suited for this purpose, since it shows more relevant albedo variations. Staying in RGB space, we choose to analyze albedo based on Omer and Werman's color lines model [141], which states that if the RGB vectors of two neighboring pixels $\{i, j\}$ are co-linear, their albedo is assumed to be constant. We thus compute our weights as:

$$\omega_{ij}^a = \begin{cases} 0, & \text{if } R_0^i, R_0^j \widehat{R_0^i, R_0^j} > 0.04 \\ 1, & \text{otherwise} \end{cases} \tag{4.10}$$

Setting $\omega_{ij}^a = 0$ in Equation 4.7, means that such gradient comes from albedo, so the gradient of the shading should be smooth. We found a difference of 0.04 radians works well in general, producing good results. We can see an example in Figure 4.2, where our measure is compared to the original Zhao's estimator, which only used Euclidean distances.

Our proposed heuristic works reasonably well when there is color information available, however it fails when colors are close to pure black or white. Thus, we choose to detect them independently and use them as similar cues as for regular albedo, so the final shading is not affected. We propose an approach based on the distance from a color to the black and white references in CIELab space (given its better perceptual uniformity than RGB), which gives a measure of the probability of a color being one of them.

From the light field $\hat{L}$, we compute the perceptual distance of each pixel to the white color as $\mathcal{D}_i^w = \|\hat{L}_i - w\|_2^2$, and analogously the distance to black $\mathcal{D}_i^b$; where w and b may change depending on the implementation. With that, we compute the probability of a pixel of being white or black as $\mathcal{P}_i^w = \exp(-\mathcal{D}_i^w / \mathcal{D}_b^w)$, with $\mathcal{D}_b^w$ being the maximum distance in CIELab space (see Figure 4.3). Then, we label the gradients as:

$$g_{ij}^w = \begin{cases} 0, & \text{if } (\mathcal{P}_i^w \geq \tau \| \mathcal{P}_j^w \geq \tau_1) \wedge (|\mathcal{D}_i^w - \mathcal{D}_j^w| > \tau_2) \\ 1, & \text{otherwise} \end{cases} \tag{4.11}$$

where $\tau_1 = 0.85$ and $\tau_2 = 0.05$. And we impose the additional condition that it must be a real gradient, so $|\mathcal{D}_i^w - \mathcal{D}_j^w| > \tau_2$ avoids marking pixels inside uniform areas. The black albedo labeling g_{ij}^b is analogously formulated. τ_1 and τ_2 were set empirically, but work well for all tested scenes. Then, we compute the final albedo threshold for each gradient as $\omega_{ij}^a = \max(\omega_{ij}^a, g_{ij}^w, g_{ij}^b)$. The result of this step is a binary labeling, where each gradient is labeled as albedo or shading change (Figure 4.3).

4.4.4.2 *Occlusion Gradient (ω^{occ})*

Previous work assume that discontinuities come from changes in albedo or changes in shading, but not both. However, we found they can actually occur simultaneously at occlusion boundaries, becoming an important factor in the intrinsic decomposition problem. Our key idea then is to detect the corresponding gradients and assign them a low weight ω_{ij}^{occ} in Equation 4.7, so larger changes are allowed in shading and albedo at the same time. Contrary to single 2D images, 4D light fields provide several ways to detect occlusions, like analyzing the epipolar planes [**?** 180] or using defocus cues [179]. In the following, we describe a simple heuristic assuming an available depth map [171], although it can be easily adjusted if only occlusion boundaries are available:

$$\omega_{ij}^{occ} = \begin{cases} 0.01, & \text{if } |D_i - D_j| > 0.02 \\ 1, & \text{otherwise} \end{cases} \tag{4.12}$$

where the depth map D is normalized between 0 and 1. Note that we cannot set $\omega_{ij}^{occ} = 0$ because it would cause instabilities in the optimization. Figure 4.4 (c), show the effect of including this new term.

4.4.5 *Global Coherency*

After solving Equation 4.8 we get S_1 and R_{g1}. Given the way normalization of shading values is performed in Equation 4.4, we found some views may become a bit unstable, affecting the angular coherence of the results.

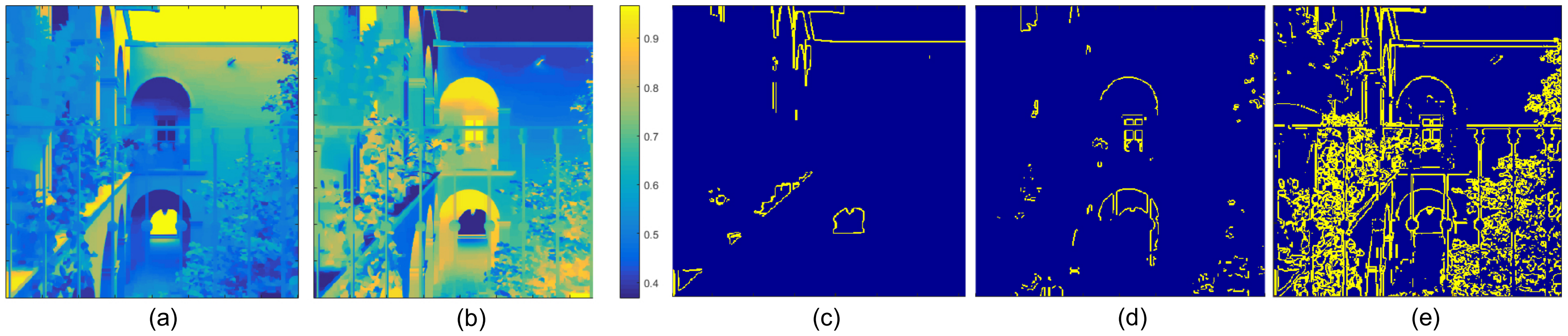


Figure 4.3: *(a) Probability of being white, $\mathcal{P}_i^w$ (b) Probability of being black, $\mathcal{P}_i^b$ (c) White pixels masked after g_{ij}^w (d) Black pixels masked after g_{ij}^b (e) Final albedo weights ω_{ij}^a taking into account color, white, and black information.*

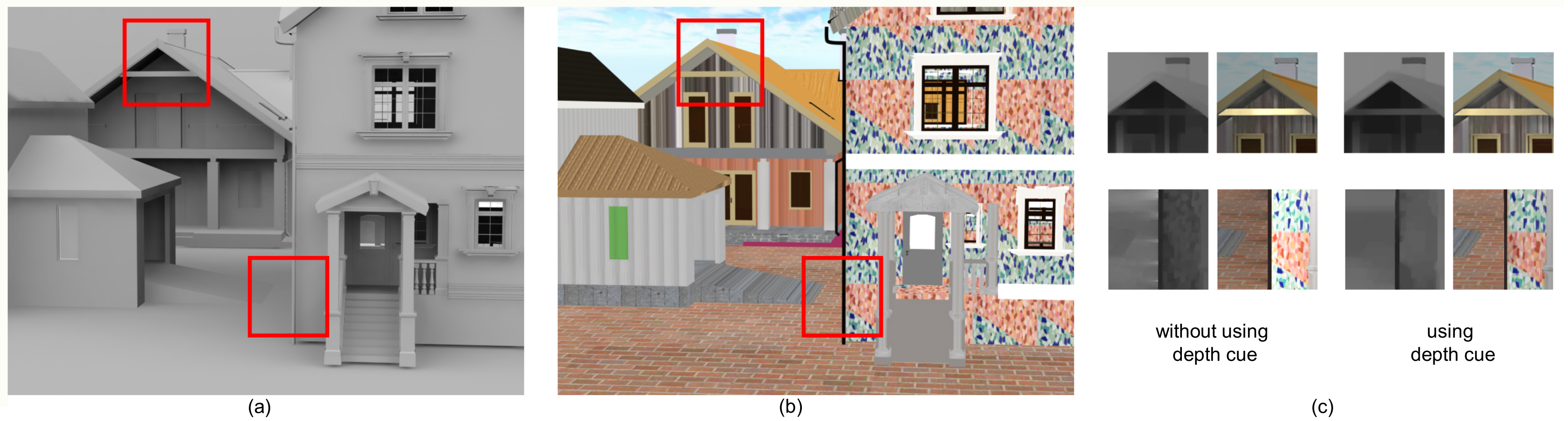


Figure 4.4: *(a) Ground truth shading. (b) Ground truth reflectance. (c) Without ω^{occ}, the algorithm classifies some prominent gradients as albedo, so it enforces continuous shading, causing artifacts. Taking occlusions into account fixes this limitation, producing results closer to the reference.*

A straightforward approach could be to apply another 4D l_1 filter (Equation 4.3) over S_1. But, this tends to remove details, wrongly transferring them to the reflectance producing an over-smoothed shading layer and a noisier reflectance one.

We found filtering R_{g1} provides better results. Because R_{g1} already features uniform regions of color, the 4D l_1 filter finishes flattening them for enhanced angular coherence, obtaining $\hat{R}_1$. Again, we use $\beta = 0.05$. From there, we compute our final smooth and coherent shading S_f as $||\hat{L}||_2/\hat{R}_1$. And the final RGB reflectance as $R_f = L/S_f$.

4.5 RESULTS AND EVALUATION

We show the whole pipeline in Figure 4.5. The central view is shown after each step of the Algorithm 2, plus the whole light field is shown in the Supplementary Material [1]. The input light field L, the filtered version $\hat{L}$ and the normalized version $||\hat{L}_2||$ are shown in Figures (a) to (c). We observe that the variation between the original light field L and the filtered one $\hat{L}$ is very subtle. In particular, in this figure, it is more noticeable in very dark regions where black gradients become grayish. This is favorable to the gradient-based solver we use to solve Equation 4.4, which is very sensitive to very dark areas. The output from Equation 4.8 is shown in Figures (d) and (e), and, although the shading looks pretty consistent in one view, it lacks of angular consistency when the whole volume is visualized (as shown in the Supplementary). Finally, from the filtered reflectance $\hat{R}_{g1}$ (f) and the original light field L, we are able to recover the coherent shading S_f (g) and reflectance layers R_f (h). Note that the initial filtering operation also removes small details in shadows and texture, which are recovered in the reflectance layer. This is favorable if the details removed are high frequency texture, as we can see in Figure 4.6 (top), but may also cause small remnants of shading in the reflectance, as we can see in Figure 4.5 (h).

Figures 4.6 and 4.7 show several results using our method. We refer the reader to the Supplementary Material [1] to visualize the whole set of light fields with ground truth data and comparisons, where the critical improvement in angular coherence over state-of-art methods can be fully appreciated. All our results have been generated automatically with the fixed parameters given in the text.

Figure 4.6 shows two synthetic scenes of our dataset along with comparisons with the works of Zhao et al. [197] and Bell et al. [13]. In the case of Bell's method, it is based on a fully connected conditional random field, which produces a dense graph connecting all pixels in the image. This strategy does not converge to a coherent solution even for adjacent views of the light field, which translates into obvious flickering artifacts. Moreover, the reported computational cost of the method is of the order of 10 minutes for a moderate-size image, whereas our method, which is based on a much simpler linear system, performs in about 10 seconds. Since all the light fields shown here have 9×9 views, the efficiency of our method becomes crucial to make the method practical. Zhao's method, as can be observed in the accompanying videos, is not free of angular flickering, which is eliminated almost completely with our new albedo and occlusion gradients, plus 4D filtering.

Intrinsic light field decomposition extends the range of edits that can be performed to a light field with available tools [80, 122]. Figure 4.8 shows two examples, where simple albedo and shading edits allow to change the

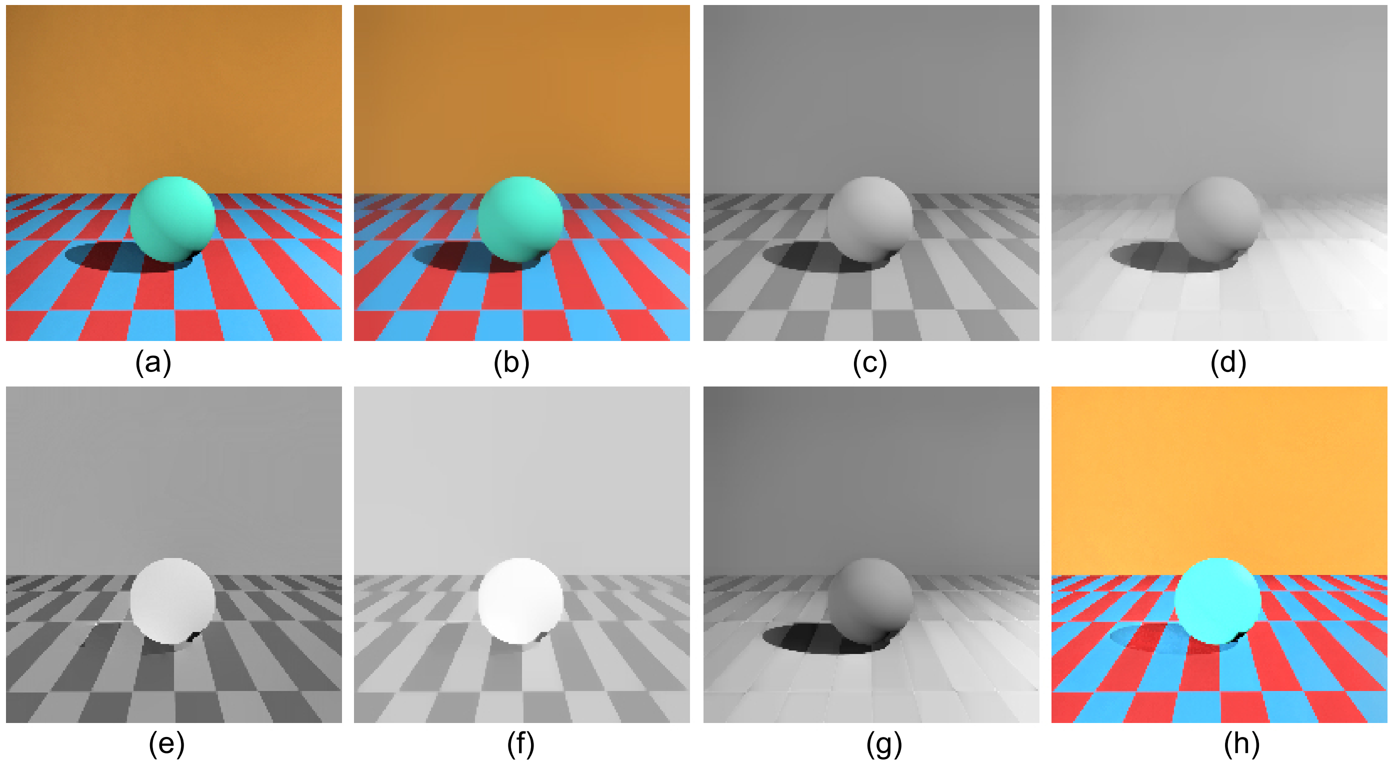


Figure 4.5: *Complete pipeline with a simple scene [2]. The central view is shown here and the whole light field is shown in the Supplementary Material [1]. (a) Input light field L. (b) Filtered light field $\hat{L}$. (c) Normalized input $||\hat{L}||_2$. (d) Resulting shading S_1 from Equation 4.8; note that although it looks consistent in one view, the global coherency is not guaranteed as shown in the Supplementary Material videos. (e) Resulting reflectance R_{g1} from Equation 4.8. (f) Filtered reflectance $\hat{R}_{g1}$. (g) Final shading S_f. (h) Final reflectance R_f.*

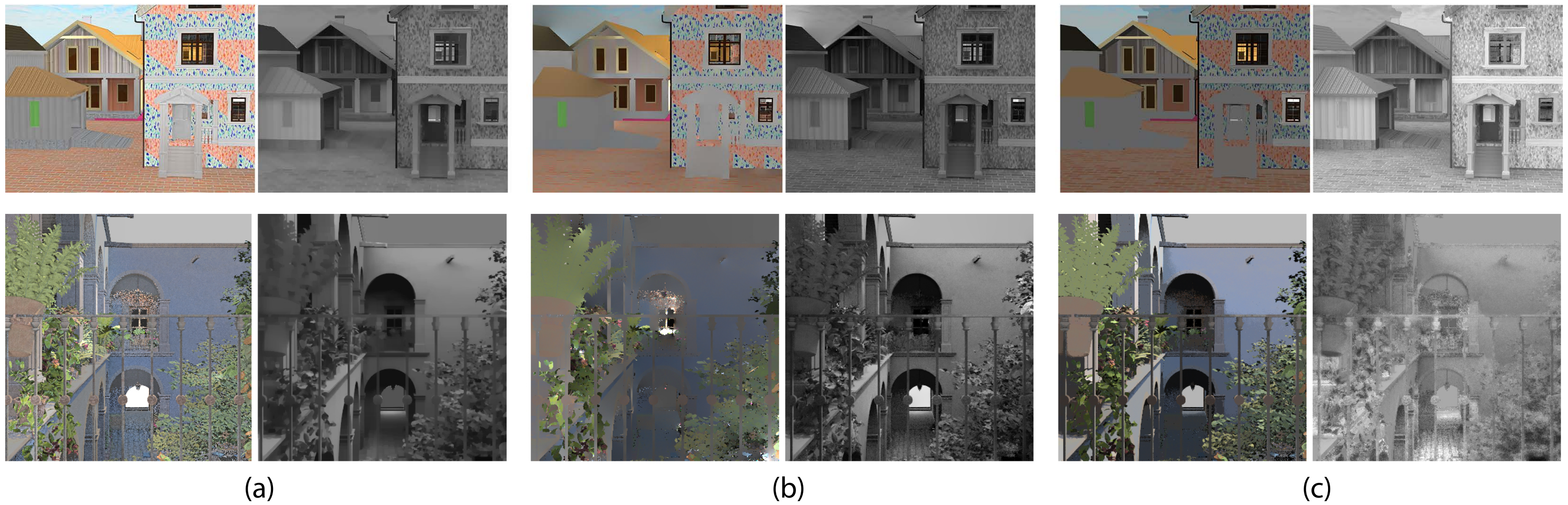


Figure 4.6: *Comparison with previous work, using synthetic datasets [179]. (a) Our proposed method. (b) Zhao et al. [197]. (c) Bell et al. [13]. Single view differences can be appreciated in these representative frames, where all the results look similar. However, the critical aspect to evaluate in light fields is the angular coherency and the total processing time. Please refer to the videos in the Supplemental Material [1] for a visual comparison of the angular coherence in the solutions.*

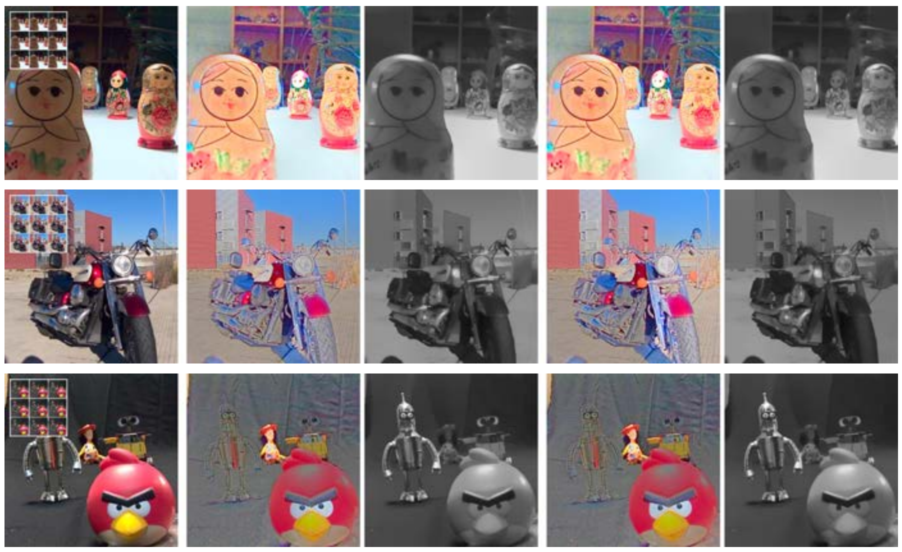


Figure 4.7: *Results of our method on three real light fields taken with the Lytro*^{TM} *camera [80]. We show the decomposition for two opposite views. Please refer to the videos in the Supplementary Material [1].*

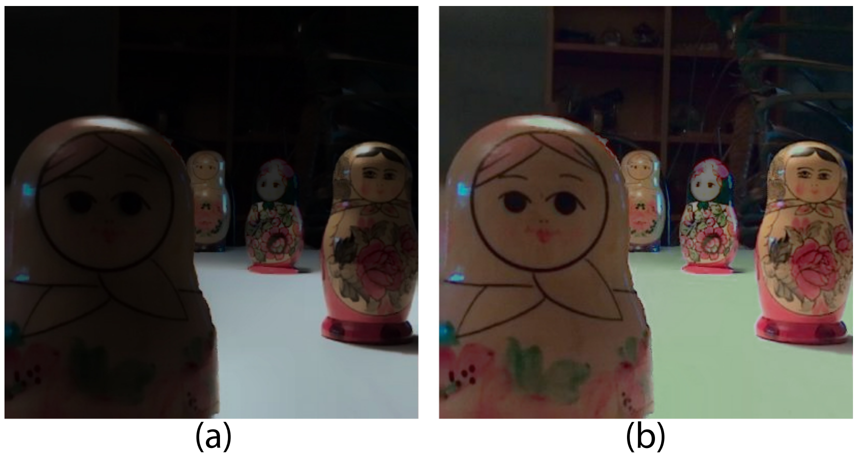


Figure 4.8: *Editing operations performed by modifying the shading (a) and the albedo (b) layers independently. Check the accompanying videos to see the complete edited light field.*

appearance coherently across the angular domain, something very difficult to achieve in RGB space.

4.6 CONCLUSIONS AND FUTURE WORK

We have presented the first method for intrinsic *light field* decomposition, which follows up existing approaches for single images and video, enabling practical and intuitive edits in 4D. Our method is based on Retinex formulation, reviewed and extended to take into account the particularities and requirements of 4D light field data. We have shown results on both synthetic and real datasets, which compare favorably against existing state-of-the-art methods, as shown by the accompanying videos in the supplemental material. Our method is efficient (a crucial aspect given the higher dimensionality of light fields), and our formulation still produces high quality results even in the absence of accurate depth information.

We have shown a straightforward editing example, but it would be interesting to see other applications enabled by our enhanced angular coherency, which will help in quick selection of objects or propagation of edits across the whole set of views.

For our albedo and occlusion cues, we currently rely on simple thresholds. A more sophisticated solution could make use of multidimensional Conditional Random Fields [79]. Despite the flexibility of our formulation with respect to depth data, a current limitation is that its quality can directly affect the final results. More sophisticated occlusion heuristics could combine information from the epipolar planes to make this term more robust.

Finally, to reduce the complexity of the intrinsic decomposition problem, some simplifying assumptions are usually made, with the most relevant ones about the color of the lighting (white light) and the material properties of the objects in the scene (non-specular lambertian surfaces). In this work we focused on coherency over subtleties in the single-view decompositions. However, we believe light field data captures rich radiometric scene information that will help lifting such limiting assumptions in the future.

Part III

STYLE IN ARTISTIC SCENES

In this part we focus on illustration art, clip art in particular. We present a feature-based representation of style for this kind of visual data, and propose a similarity metric which allows us to compare two pieces of vector art based on style. This metric is computed based on the human perception of style leveraging crowdsourcing experiments. We show several applications of this metric such as style-based image retrieval or mash-up creation; and present an exploratory interface which allows to efficiently navigate through massive amounts of this kind of datasets taking into account both semantic labeling and style.

5 A SIMILARITY MEASURE FOR ILLUSTRATION STYLE

In this chapter we present a method for measuring the similarity in style between two pieces of vector art, independent of content. Similarity is measured by the differences between four types of features: color, shading, texture, and stroke. Feature weightings are learned from crowdsourced experiments. This perceptual similarity enables style-based search. Using our style-based search feature, we demonstrate an application that allows users to create stylistically-coherent clip art mash-ups.

This work is published in *ACM Transactions on Graphics* and presented at *SIGGRAPH 2014*. It was partially developed during a three-month internship at Adobe, Seattle (USA).

E. Garces, A. Agarwala, D. Gutierrez & A. Hertzmann
A Similarity Measure for Illustration Style
ACM Transactions on Graphics (SIGGRAPH), Vol.33 (4), 2014

5.1 INTRODUCTION

Vector art is one of the most common forms of two-dimensional computer graphics. Clip art libraries contain hundreds of thousands of pieces of vector art designed to be copied into documents and illustrations. These collections are typically tagged by object categories; searches for common objects (e.g., "dog") yield huge numbers of results. However, there is another aspect of vector art that is currently much harder to search for: *style*. Clip art comes from many artists and many sources, in a vast range of visual styles, including sketches, woodcuts, cartoon drawings, and gradient-shading; some are very cartoony and whimsical, whereas others are more professional-looking. Because clip art comes from heterogeneous sources with very inconsistent tagging, these datasets lack any reliable annotation of artist or style.

While simulation of depiction style has long been a focus of non-photorealistic rendering [60], little attention has been paid to understanding style, and no good tools exist for stylistic search or analysis in clip art datasets. Indeed, it is fundamentally difficult to define a simple function that describes these different styles. But, with the recent dramatic growth in the quantity of visual content available online and the rising popularity of remixed and mashup art [105], stylistic search could be valuable for many design applications.

This chapter presents a *style similarity function* for clip art. That is, given two pieces of clip art, our function computes a real-valued measure of their style similarity, independent of content. We demonstrate style-based search, where clip art search results are sorted by similarity to a query artwork. We describe a clip art mashup application that uses style-based search to help users combine multiple pieces of stylistically-coherent clip art into a composition. For example, if a user has already placed a house and tree in a sketchy pen style onto the canvas, and then searches for a dog, our application re-orders search results so that dogs of a similarly sketchy pen style are shown first.

We compute our style distance function using a combination of crowdsourcing and machine learning. We gathered a stylistically-diverse collection of clip art. Then, for a large selection of clip art triplets, we gathered

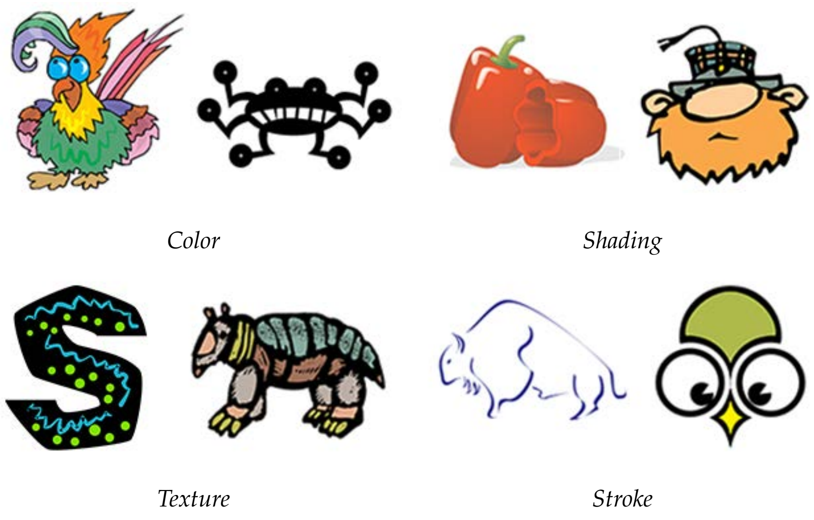


Figure 5.1: *For each feature category we show two pieces of clip art whose style is very different. The color example contrasts a colorful illustration from a monochrome one. The shading example shows a gradient-shaded illustration next to one with regions of constant color. The texture example shows an image with artistic patterns next to a woodcut illustration with more stochastic patterns. Finally, the stroke example pairs a sketchy illustration with lines of varying width next to smooth contours of constant width.*

Mechanical Turk (MTurk) ratings of the form "Is clip art A more similar in style to clip art B or C?" We then learned a model of stylistic similarity from these ratings. The model is based on a set of features that we observe to be descriptive of the style of vector art. In total we compute 169 features in four categories: color, shading, texture, and stroke. The similarity function is a weighted L2 distance of the feature vectors; the weights are learned by maximizing the probability of the MTurk ratings. Learning with a sparsity prior produces a final weighting with 78 non-zero weights. We numerically evaluate the performance of our distance function on separate test data. We also perform user studies in which workers create mash-ups with and without style-based search. We find that raters judge mash-ups created with style-based search to be more stylistically coherent and generally preferable.

5.2 RELATED WORK

Though there has been considerable effort in computer graphics on stylistic rendering, there has been less work in the analysis of artistic style. Willats and Durand [184] provide an overview of the elements of pictorial style found in 2D illustrations. One approach to the algorithmic analysis of style is to learn a *generative* model; that is, a model that learns to create new examples of a style from scratch, such as generating new typefaces [175], and learning 3D hatching styles from examples [86]. A second class of approaches *transfer styles* from examples, such as transferring painting styles [73], photographic styles [6], or curve styles of 2D shapes [109]. Our work focuses on a different problem, namely, perceptual measures of stylistic similarity, rather than synthesis. To our knowledge, there is no previous work on perceptual similarity of vector art. Moreover, the above methods are not directly applicable. A generative model could theoretically be used to compute stylistic similarity; however, creating a generative model of clip art style from examples would be extraordinarily difficult.

Though image search and retrieval is a standard problem in vision and image analysis [33], recognition based on style is rare. Murray et al. [129] classify photographs according to a few photographic styles. There are more examples in other domains; style similarity functions are used to recommend music [3] and films [11] based on examples of preferences. Shamir et al. [160] describe style recognition for paintings. Doersch et al. [36] recognize the style of street scenes that visually distinguish different cities. Our method instead focuses on styles of vector art.

The most related work to ours is a method for retrieval of sketches of similar style from an art dataset [77]. They propose features computed from stroke contours which are first extracted from the image to describe the style of line drawings. Their method only applies to black and white line drawings, whereas our method can also measure differences in color, shading, and texture. Also, our technical approach is different in that we collect data on the human perception of style, and fit our style similarity function to this data.

Finally, our mashup application is similar in motivation to recent work that supports search of photo databases for visually consistent content that can be combined into composites. Photo Clip Art [101] find objects in photos whose lighting and perspective are consistent with a target scene, while Sketch2Photo [29] generates photo composites from sketches while ensuring that the photo elements are visually consistent.

5.3 CLIP ART STYLE FEATURES

The first step in building a style similarity function is to define the numerical features that identify and distinguish clip art style. Although it is difficult to specify the exact characteristics that define style, there are a series of pictorial cues that can be used to differentiate one style from another. These cues include basic visual attributes like color, shading, and texture, as well as the actual marks such as lines, strokes, and regions [39, 184]. Note that we do not need to decide a priori how these features differentiate style, or their relative weights; our goal is to create an overcomplete set of features that can be used by the learning algorithm to fit our similarity function to data. Later, we will see that some features are completely removed by L1 regularization (Section 5.6.1).

Our features are computed on bitmaps rather than vector descriptions (e.g., SVG) of clip art for two reasons. First, computing on bitmaps gives us the flexibility to include clip art whether or not a vector version is available. Second, we observed a surprising variety of vector descriptions of similar content. For example, a simple black stroke could be defined with line, path, or polygon primitives, or even worse, be the result of adding a smaller foreground region to a black background region. Converting these representations into a consistent vector format is a research challenge by itself.

We identify four main aspects which we believe best characterize styles in clip art: **color**, **shading**, **texture**, and **strokes**. Together these form a 169-dimensional feature vector $\mathbf{x}$ for any individual piece of clip art. Figure 5.1 shows some representative examples of clip art whose styles are very different along each of the identified aspects. We now describe the list of features that form our feature vector. To compute features, we render each clip art image to 400x400 pixels. For each image, we define a mask Ω that approximately covers the clip art. We select all non-white pixels, perform a morphological expand operation of ten pixels, and then fill the remaining holes. All the statistics are computed only on the domain of Ω.

COLOR. These features distinguish between different styles of color use; some styles we observe include black-and-white, monochrome, colorful, muted, and bright/saturated colors. Note that these statistics reflect styles of color usage rather than the individual colors used.

The first color features are scalar values, defined as follows: Standard deviation of hue; average saturation; standard deviation of saturation; average luminance; standard deviation of luminance; entropy of the luminance histogram, after quantizing it to 256 bins; entropy of the RGB histogram, after quantizing it to 512 bins; colorfulness, computed by the measure of Hasler and Susstrunk [68]; colorfulness, computed as:

$$\frac{1}{|\Omega|}\sum_{p\in\Omega}|R_p - G_p|+|G_p - B_p|+|B_p - R_p| \; ; \tag{5.1}$$

percentage of pixels that are black; and percentage of pixels that belong to the most dominant color.

Additionally, we define a few features in terms of a 20-bin histogram $C(h)$ of hue h, omitting pixels with saturation less than 0.1, similar to Li et al. [108]. Then, we include a feature for the frequency of the most common hue ($\max_h C(h)$), and a feature for the number of dominant hues:

$$\#(h \mid C(h) > 0.05 \max_h C(h)) \tag{5.2}$$

We also include the same two features above applied to a quantized RGB histogram; that is, the number of pixels in the most frequent color, and the number of dominant colors.

SHADING. These features distinguish types of shading. Some styles have a very cartoonish look, with sharp, simple color transitions; others have more realistic materials with smooth gradients. We describe shading with histograms of both color and grayscale gradients. The former captures the overall appearance and materials of colored images, while the latter captures transitions between shading and stroke lines, if any.

We concatenate eight-bin gradient magnitude histograms at two resolutions of the image, 1x and 0.5x. The histograms are normalized by $|\Omega|$ for the relative figure size. The resulting bins are concatenated to form the corresponding features in $\mathbf{x}$.

The first pair of histograms measure smooth gradients, while ignoring zero gradients and sharp transitions. We define the region Φ as the region Ω minus all black pixels and pixels with zero gradients. Then, we histogram the following values for all $p \in \Phi$:

$$g(p) = \min(\max(|\nabla R_p|, |\nabla G_p|, |\nabla B_p|), 1.1) \tag{5.3}$$

The natural range of the above values is $[0, \sqrt{2}]$; however, we truncate any values over 1.1 to minimize the influence of strong edges.

The second pair of histograms is computed as above, but only at black pixels, and without truncation of 1.1. This histogram is meant to quantify the number of sharp edges (e.g., ink edges).

texture. These features capture the presence of repeated patterns over the image. Texture defines the look of a depicted object at a small scale, and gives an intuition of how an object in an image would feel when touched.

Our first texture descriptor are Local Binary Patterns (LBP) [140, 199]. An LBP feature vector is represented as a series of patterns; each bin in our feature vector contains the number of times a certain pattern occurs in the image. The number of patterns is specified by the radius R and the sampling points P. We used rotational invariant patterns [140] at three resolutions $\text{LBP}_{P,R}$, in particular $\text{LBP}_{8,1}$, $\text{LBP}_{12,2}$ and $\text{LBP}_{16,4}$ yielding a total of 10, 14 and 18 patterns (bins), respectively.

We use two different sampling spaces: all edges of the figure, and only external contour lines. In total this descriptor yields 84 features.

Our second texture feature are Haralick texture features [67]. We compute all 22 Haralick texture features, which are obtained from co-occurrence matrices capturing the frequency of different combinations of grayscale pixel values.

strokes. The types of strokes used are a significant element of clip art style [77]. Clip art strokes typically vary in texture, thickness, and weight. Some of these characteristics are already captured by the LBP features, like thickness or uniformity of the stroke lines. We add additional stroke features using the Stroke Width Transform (SWT) [42], which was originally developed to recognize text in natural images. The SWT is a local operator which approximates the width of the stroke at each pixel.

We compute SWT separately on outer edges of the figure on the silhouette, and inner edges, since their appearance is often different; the result is two arrays of likely stroke width values per pixel. We then take the mean and standard deviation of these two images. Finally, to avoid scale sensitivity we compute SWT at four different resolutions of the image: 1x, 0.5x, 0.25x and 0.12x. The result is 16 feature values.

5.4 collecting similarity information

We use two sources of clip art to train our models: clip art from Art Explosion[1], a commercial collection of over 200,000 pieces of clip art, and a collection of 3,600 clip art pieces that is included with Microsoft Office. For the former collection we used crowdsourcing to collect data on stylistic similarity. In contrast, the latter, smaller collection is already organized into groups of stylistic similarity.

We manually selected 1000 images from the Art Explosion dataset that cover a wide range of styles and subjects. We used Mechanical Turk (MTurk) raters to collect style information. Each test (a HIT in Mechanical Turk terminology) consisted of fifty questions. We gathered data in the form of relative comparisons [126, 154] since they are much easier for human raters to provide than numerical distances. Each question showed three pieces of clip art A, B, and C, and the MTurk rater is asked: "Is A more similar to B or to C?" (Figure 5.2.) For the Microsoft data, we automatically generated relative comparisons through random sampling constrained so that two of the three samples in each relative comparison come from the same style group.

Each HIT was preceded by a short training session that included a few trial relative comparisons with obvious answers; the users could only access the real test once they correctly answered all the trial questions. A total of

1 www.novadevelopment.com

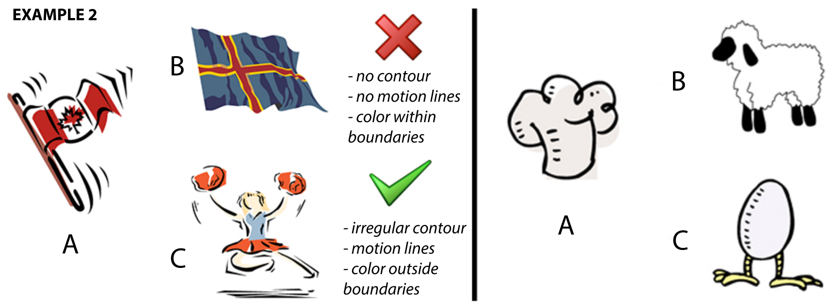


Figure 5.2: *Screenshots of our MTurk similarity collection interface. Left: An example given to users at the beginning of the test to make sure they understood the question asked. Right: An actual comparison triplet.*

313 users took part, 51.4% female. 56.5% declared some artistic experience, and an additional 7.3% claimed some professional design experience. The duration of each HIT was approximately ten minutes, for which we paid $0.30. Five control questions were included in each HIT; HITs with two errors in the control questions were rejected, with a rejection rate of 21.5%.

5.5 LEARNING SIMILARITY

This section describes our approach for learning stylistic similarity based on the feature vector defined in Section 5.3 and the training data from Section 5.4. Our learning approach uses a combination of previous techniques that works well for our application. Let $\mathbf{x}$ and $\mathbf{y}$ be the feature vectors for two pieces of clip art. We aim to learn a Euclidean distance metric

$$d(\mathbf{x},\mathbf{y}) = \parallel \mathbf{x}-\mathbf{y} \parallel_{\mathbf{W}} = \sqrt{(\mathbf{x}-\mathbf{y})^T\mathbf{W}(\mathbf{x}-\mathbf{y})} \tag{5.4}$$

parameterized by a diagonal matrix $\mathbf{W}$. This problem is well-studied and known as *metric learning* [96, 154]. Our problem is further complicated by the fact that crowdsourced relative comparisons are not always reliable; there are several approaches to modeling worker reliability both in classification [183] and search ranking [30]. We minimize this reliability problem in two ways. First, we use a number of control and training questions to reject bad workers. Second, we use a logistic formulation of the probability of each rating [170] that expects more noise for relative comparisons with less clear answers.

Specifically, we use the metric learning approach of Donovan et al. [138], who adapt the logistic formulation of Tamuz et al. [170] to the scenario of learning from features: Given clip arts A, B, and C, we define $q = 1$ if the rater states that A and B are more similar, and $q = 0$ if the rater states A and C are more similar. We parameterize the model by the diagonal of the weight matrix: $\mathbf{w} = \text{diag}(\mathbf{W})$. We model the probability that a user rates $q = 1$ given the tuple as:

$$P^A_{BC}(q=1) = \sigma(d(\mathbf{x}_A,\mathbf{x}_C) - d(\mathbf{x}_A,\mathbf{x}_B)) \tag{5.5}$$

$$\sigma(x) = 1/(1+\exp(-x)) \tag{5.6}$$

In addition to this model, we aim to regularize and sparsify the weight vector. We thus assume a Laplacian prior on the weights:

$$p(\mathbf{w}) \propto \exp(-\lambda||\mathbf{w}||_1) \tag{5.7}$$

where λ is a regularization weight. Given a set of Turker ratings $\mathcal{D} = \{(A_i, B_i, C_i, q_i)\}$, we learn the weights $\mathbf{w}$ by Maximum A Posteriori estimation, which entails minimizing the following objective function:

$$E(\mathbf{w}) \quad = \quad -\sum_{i=1}^{|\mathcal{D}|} \log\left(P_{B_i C_i}^{A_i}(q_i)\right) + \lambda||\mathbf{w}||_1 \tag{5.8}$$

where i indexes over the training tuples. We perform optimization using L-BFGS [200].

We set the regularization weight λ by five-fold cross-validation on the training set. After training, all weights with $w < 0.02$ are set to zero.

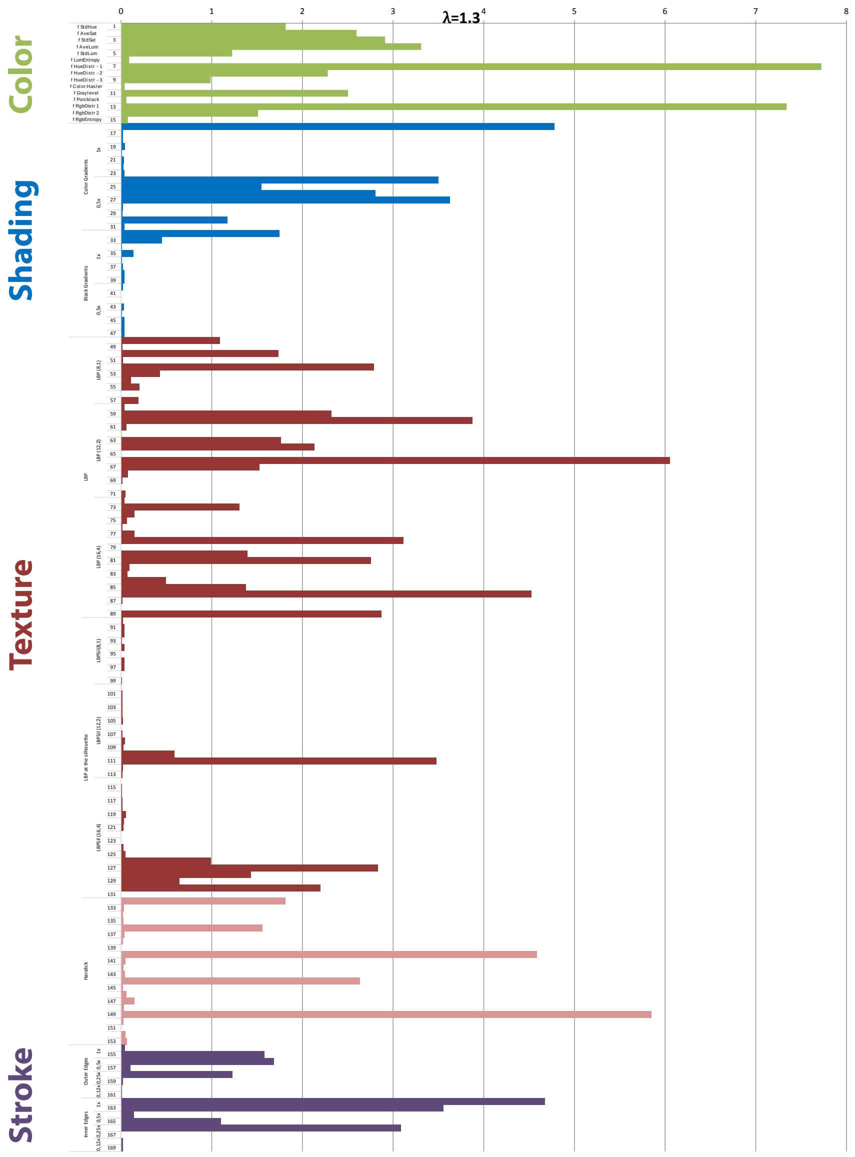


Figure 5.3: *Learned weights* $\mathbf{w}$

	MTurk		MS
	Raw	Majority	
Learned Weights	0.72	0.81	0.95
Uniform Weights	0.68	0.75	0.94
Humans	0.68	0.74	N/A
Oracle	0.83	1	N/A

Table 5.1: *Accuracy of our method (with and without training) and two baselines, on both the MTurk and Microsoft testing data. Higher values are better (see text for details).*

	MTurk		MS
	Raw	Majority	
Learned Weights (MTurk + MS)	1.75	1.57	1.18
Learned Weights (MTurk only)	1.76	1.64	1.30
Learned Weights (MS only)	2.52	1.77	1.11
Uniform Weights	1.83	1.73	1.39

Table 5.2: *Perplexity of our method on both the MTurk and Microsoft testing data. Lower values are better (see text for details).*

5.6 SIMILARITY FUNCTION EVALUATION

We now evaluate the influence of the regularization term, the performance of the learning process, and the quality of the training data. The training set includes 25,540 tuples gathered via Amazon Mechanical Turk and 25,000 tuples generated from the labeled data from Microsoft.

5.6.1 *Feature Selection*

The use of L1 regularization encourages a sparse set of weights. Through cross-validation, the regularization weight was set to $\lambda = 1.3$. After the thresholding step, we reach a final weight vector with 78 non-zero weights; the 91 zero-weight features can be ignored. The learned weights are shown in Figure 5.3. Some features are deemed irrelevant (e.g., the entropy of both luminance and color), while others can be covered with a smaller set of features (e.g., by using a few bins of LBP instead of all). The highest weighted feature is the number of unique hues; the number of RGB colors is the second highest. Several bins of the LBP, Haralick, Color Gradient, and Stroke width features are also highly weighted. The weights show that these high-dimensional features can be simplified to lower-dimensional combinations for our application.

5.6.2 *Evaluation on a Testing Set*

We gathered a new set of relative comparisons to form a separate testing set to evaluate our model. We sampled 1,000 new tuples from the Art Explosion collection, and 10,000 tuples from the Microsoft labeled data. We used MTurk to obtain 10 ratings per tuple for the Art Explosion data; this redundancy helps us to understand which tuples have clear answers. We removed tuples with high disagreement, i.e., tuples with MTurkers split 5-5 or 6-4 in their judgment of which pair is more similar. This left 633 reliable comparisons, each with 70% or more agreement. Disjoint training tuples were used between the training and test sets, though both sets of tuples used clip art pieces drawn randomly from the same clip art collection.

We evaluate performance by two metrics on the test set: *accuracy* and *perplexity*. Accuracy is the percentage of testing tuples correctly predicted by our method. For MTurk tuples, accuracy can be computed in two ways: *raw* and *majority*. Raw accuracy counts each of the 10 opinions per tuple separately; majority assumes the majority opinion is correct and assigns all votes to the winner. So, an ideal predictor which always chooses the majority opinion would have a majority accuracy of 1, but a raw accuracy of less than 1 assuming there is disagreement between human raters. In either case, a completely random predictor would have an accuracy of 0.5.

Perplexity Q is a standard measure of how well a probability model predicts a sample; it takes into account the uncertainty in the model, giving higher weight to predictions where the model outputs higher confidence (i.e., P^A_{BC} close to 1 or 0). It is given by

$$Q = 2^{-(\ln P(\mathcal{T}))/|\mathcal{T}|} \tag{5.9}$$

where $P(\mathcal{T})$ is the probability of the test set according to a given model, computed by Equation 5.6 over all test tuples. The perplexity is 1 for a model that makes perfect, confident predictions at all times. The perpexity is 2 for a model that outputs 0.5 (total uncertainty) for all evaluations. The perplexity is worse for a model that makes highly confident but wrong predictions. Perplexity can also be computed with raw and majority data.

The influence of varying the parameter λ in the error measures *accuracy* and *perplexity* is shown in Figure 5.4.

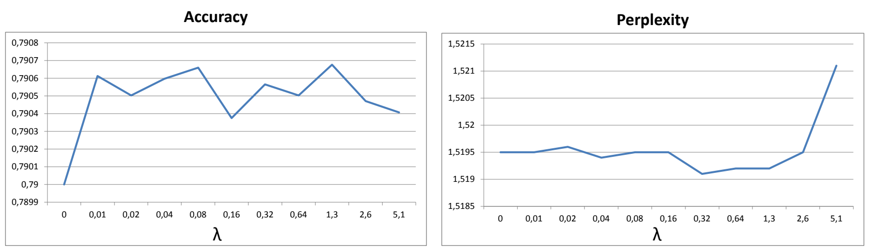


Figure 5.4: *L1 Regularization: accuracy and perplexity for different values of lambda.*

We show accuracy data in Table 5.1 and perplexity data in Table 5.2, both for MTurk relative comparisons and Microsoft data. We show the results of our model both with uniform weights $\mathbf{w} = [1...1]^T$, and with weights learned from training data. Note that there is no need to separate the Microsoft data into raw and majority, since there is no disagreement. We compare our results with two baselines in addition to a random baseline. The *oracle* predictor has access to all the human ratings and always gives the majority

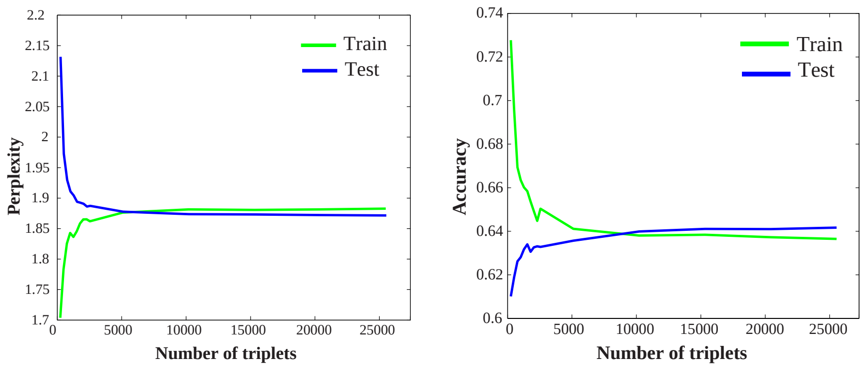


Figure 5.5: *Perplexity (left) and accuracy (right) on the test data as a function of the number of MTurk tuples used during training.*

opinion; note that its raw accuracy is not 1 due to human rater disagreement. The *human* accuracy is a measure of how well the average individual human performs relative to the majority; it is computed as an average over each human rater's percentage of agreement with other raters on the same tuples. Note that perplexity cannot be computed for the oracle and human models since they are not probability distributions.

For the MTurk data, the accuracy of our model is roughly equal to average human accuracy without training; with trained weights, our model performs better than human accuracy. Our model is able to predict the majority opinion 81% of the time. Not surprisingly, the oracle predictor is still significantly better than our model. Our model is 95% accurate on the Microsoft data. This data is easier, since in each tuple two of the clip art pieces have the same style; tuples with three different styles (as is common in the MTurk data) are more subjective.

We also experimented with only training on the Microsoft or MTurk datasets. Training on only the Microsoft data performed poorly on the MTurk data, while training on only the MTurk data performed reasonably on the Microsoft data. The combination of both datasets during training performs the best or nearly the best on both testing sets (Table 5.2).

We can check whether we have collected enough tuples by holding back some of the training data and observing accuracy and perplexity (Figure 5.5). We can see that we have collected more than enough randomly sampled data; improvements stop at around 10,000 triplets relative to the 25,000 we collected. However, our triples are randomly sampled; it may be that sampling triples closer in style could add further discriminative power [170].

5.6.3 *Failure Cases*

Our similarity measure disagrees with the MTurk majority 19% of the time; we include the 25 worst examples where our probability is most inconsistent with the MTurk opinions in the Appendix 5.A. We show two typical examples in Figure 5.6. In the left example, the style of all three clip arts is very different, and it is surprising the MTurk opinion is so consistent either way. The right example shows another common error; our metric generally believes two clip arts with color are more similar with each other than to a black and white clip art. However, in this case, the iconic nature of the examples overwhelms other differences for humans.

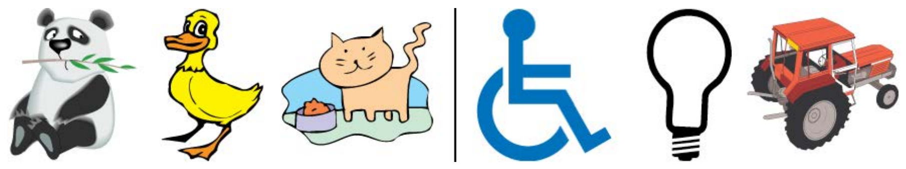


Figure 5.6: *Two tuples incorrectly labeled by our similarity function. In both cases, Turkers agreed (by a 9-1 margin) that the first clip art is more similar to the second than to the third, whereas our algorithm scores the first and third as more similar.*

5.7 APPLICATIONS

We apply our style similarity metric in three ways: to cluster and visualize the space of clip art styles, to perform search, and to support a mashup application for creating compositions of clip art.

5.7.1 *Clip Art Style Visualization*

We use our style distance function as a basis for visualizing the diverse styles of clip art in our dataset. In particular, we use the popular t-SNE [177]; this technique maps a high dimensional feature vector to a 2D space where similar styles are located close to each other. To create the visualization in Figure 5.7, we reduce the entire dataset to 100 examples by k-means on the $\mathbf{Wx}$ values, select only dogs, and then perform t-SNE. We can observe a clear separation of style; colorfulness increases from top to bottom, while stroke complexity varies left to right.

5.7.2 *Search*

Figure 5.12 shows typical results of search queries using our method; we show an additional 500 examples in the Supplemental Material[2]. Each image in the left column shows a query image. The next column shows the results from the dataset that our method judges to be most similar. In each case, the algorithm appears to have recovered at least some artwork from the exact same artist and style, after searching in the entire dataset of 194,663 pieces. The other six columns show the top 3 results each for keyword queries, i.e., the three most similar *cat* images, the three most similar *fish* images, and so on. The amount of clip art in each category ranges from 598 (*sky*) to 1836 (*man*). In many cases, the algorithm does not find matches by the same artist, but finds excellent matches nonetheless, with similar stroke styles, similar fill styles, and so on. For example, the dinosaur query is drawn in a woodcut style with solid color fills. The first two *cat* results appear to be from the same artist, whereas the third is a tiger that is not a woodcut but is similar nonetheless. The other examples show cases where woodcut styles are found, or, when no more woodcuts can be found, similar non-woodcuts are returned.

Search is typically evaluated with precision-recall, where the goal is to return search results that are in the same category as the query. In our case, most of our data is not cleanly separable into relevant and irrelevant categories. The Microsoft data is separated into groups; however, this represents only 1% of our overall data. Also, we found that many of the groups had

2 `http://webdiis.unizar.es/~elenag/projects/SIG2014_styleSim/downloads/search1.html`

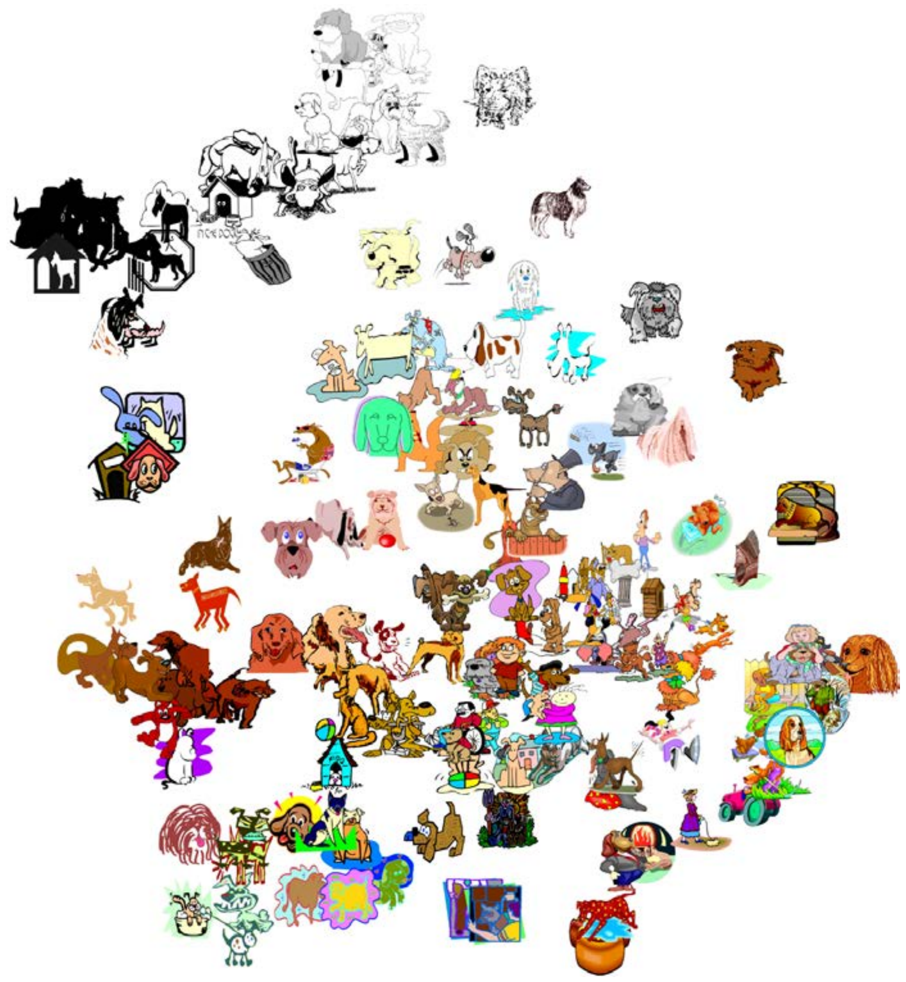


Figure 5.7: *A 2D embedding of clip art styles, computed using t-SNE, shown with "dog" examples.*

similar styles to each other, which means that clip art from one group is often relevant to queries from another group.

5.7.3 *Clip-Art Mash-ups*

We also demonstrate the usefulness of our similarity metric with a simple clip-art mashup application (Figure 5.9). The application allows users to search for and combine multiple pieces of clip art into a composition. Our clip art library is organized into 13 common categories (e.g., *dog, tree, house*); the user can also search by keyword to find objects not in common categories, or to add a modifier (e.g., *running dog*). We provide 11 pre-made backgrounds that the user can select from, including a blank white background. Clip art is added to the composition by dragging and dropping from the search results; it can be resized or rotated, and the layer order can be modified. The app is implemented in HTML5. We use Apache Lucene to extract keywords from clip art file names and tags. We show several examples of mash-ups created with our app in Figure 5.7.3 and the Supplemental Video [3].

3 `https://www.youtube.com/watch?v=GBWrRHeAxuM`

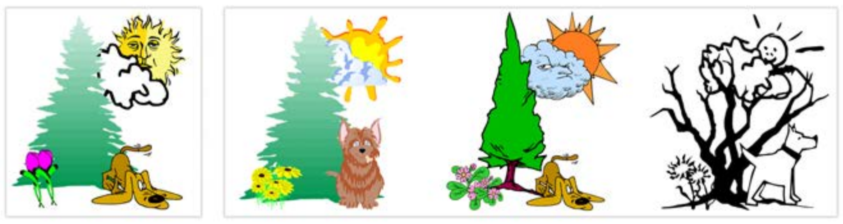


Figure 5.8: *The leftmost composition is generated by selecting from a dataset of 200K clip art searched with keywords: dog, tree, sun, cloud, and flower. Unfortunately, the styles of the clip art are inconsistent. Our style similarity function can be used to re-order the results of a search by style. The next three scenes are generated by fixing one element, and then searching for stylistically similar clip art with the above keywords. In each case, the additional clip art were chosen from the top twelve returned results (out of thousands).*

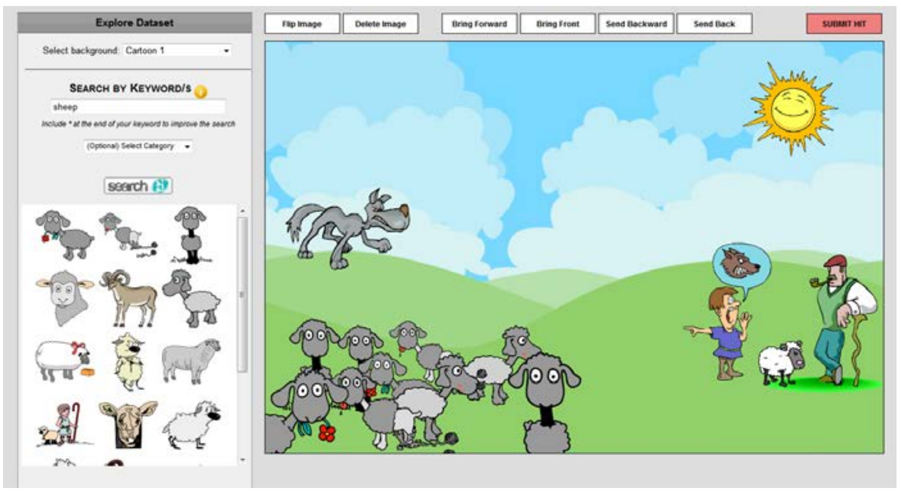


Figure 5.9: *Our mash-up interface. The user searches for clip art in the left panel by typing keywords and/or by selecting categories from the drop-down menu. Results are shown below the search button, sorted by stylistic similarity to already-selected clip art.*

During search, clip art that matches the keyword and category query are sorted by their style similarity to all currently-used clip art (averaged over the existing clip art). The sort order automatically updates whenever clip art is added or removed from the canvas.

When there is no clip art in the current canvas, a naïve approach would be to order the search results randomly. However, some styles are more common in the dataset than others, leading to some styles not being represented in the results. As in other search problems [147], it is important to produce results with diverse styles. We produce diversity as follows: In advance, we produce a two-level hierarchical clustering; the entire dataset is clustered into 100 clusters by k-means on $\mathbf{Wx}$ values, and then these cluster centers are clustered again to produce seven top-level clusters. Then, at search time, the first seven search results are sampled sequentially from the seven top-level clusters. This process repeats to generate the next seven results, and so on. Within a top-level cluster, the second-level clusters are also sampled sequentially, to avoid repetition within the cluster.

EVALUATION. To evaluate the impact of our similarity measure on mash-up creation, we performed two different MTurk studies. In each study, users were asked to create high-quality compositions. Some users were provided

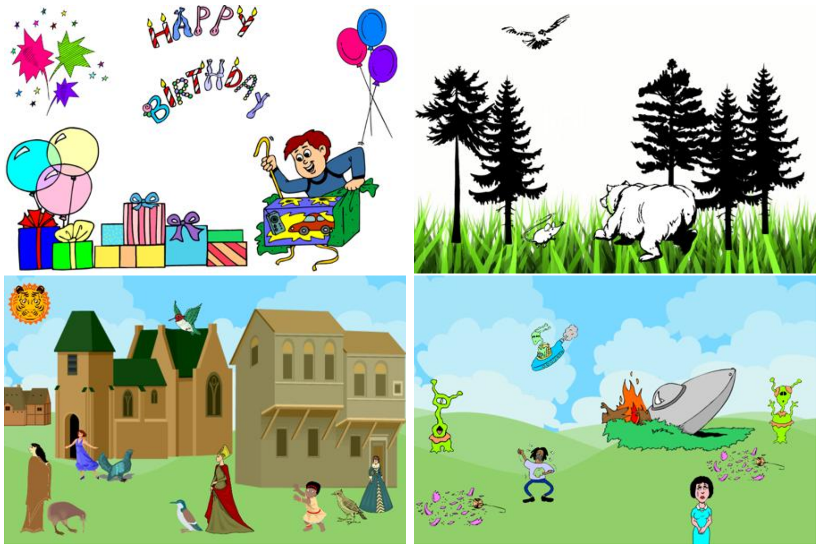


Figure 5.10: *Typical mash-ups created by Turkers using our similarity.*

with a version of the application in which search results are sorted by similarity to the existing composition, and others received an interface which sorts search results randomly. Users were not told of this difference.

In the first study, users were given an open-ended task: they were free to choose the topic of the composition, the background, and the number of images to use. In the second study, we asked for a specific story, fixed the background, and required the users to include at least four different pieces of clip art and perform at least four different searches. Study details are in the Appendix 5.B.

We gathered 38 compositions for the first study (19 with our metric on), and 95 compositions for the second (47 with our metric on). We show several typical examples of compositions both with our metric and without in Figures 5.10 and 5.11, respectively (we include all compositions in the Sup-

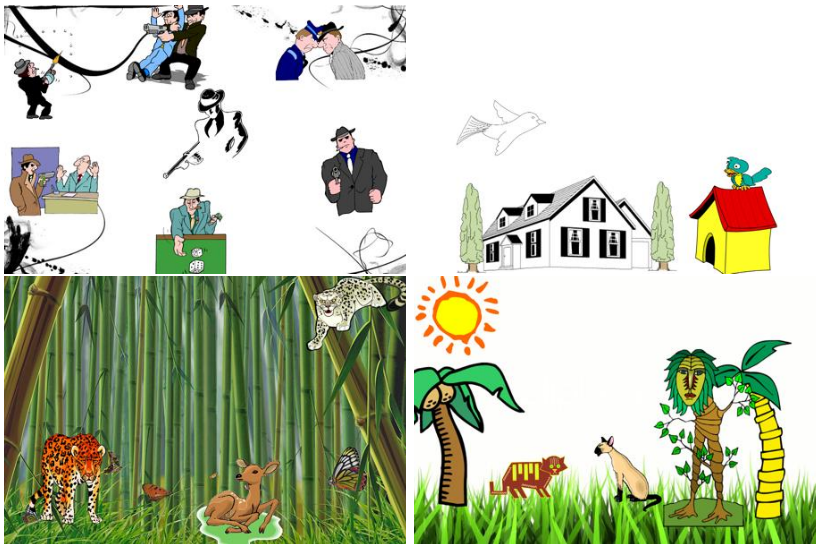


Figure 5.11: *Typical mash-ups created by Turkers without our similarity metric. All results are included in the Appendix 5.B.*

plemental Material). We then asked a separate pool of MTurk workers to evaluate the compositions on both style coherence and general preference. Specifically, we performed a 2AFC test between randomly sampled compositions with and without our metric. We asked the questions: *which composition has a more coherent style?*, and *which compositions do you like better?* For both tests, we perform a one-sample, one-sided t-test comparing the mean of users preferences against the null hypothesis (people have no preference, $\mu_0 = 0.5$). Compositions created with our metric were perceived as having a more coherent style (67% of agreement for the first and 69% for the second, with $p < 0.01$), and participants liked them more (63% for the first and 66% for the second, with $p < 0.01$). The effect sizes for style coherence were 1.4 and 2.4 for the first and second experiments, and 1.4 for general preference for both experiments; these effects are large.

When using the similarity-based interface, the clip art pieces used in the mash-ups had a mean position of 24 in the search results (confidence interval: ± 4.4), whereas without our metric, the mean position was 40 ± 7.1. This indicates that, although users need to look through many results in order to find content that matches their goals, our interface cuts the length of the search nearly in half.

5.8 CONCLUSION

We have presented a similarity function for illustration style that is trained from crowdsourced data on the human perception of similarity. We also demonstrate a mash-up application for combining clip art of a consistent style into a scene. There are a number of ways we could improve our system. We could collect data on how humans name illustration styles, so that a user could ask for a "sketchy dog." We could learn relative attributes of style [143], so that a user could ask for a dog similar to the current one but "more colorful." Our metric learning technique is fairly simple and there may be other, possibly non-linear methods that work as well or better; we have made our data public for further experimentation. Finally, a more fundamental problem is to understand and parse the elements that form an illustration, such as outlines, fills, object identity, and so on. (Even identifying the outline strokes in the vector art in our libraries is non-trivial.) This analysis could lead to richer and more accurate analysis of illustration style and similarity.

Beyond our application of clip art style, we believe that recognizing the style of visual content will become increasingly important as the amount of online content increases and remixing becomes more and more common. Design-focused social networks such as Pinterest can also benefit from the ability to search by style, without relying on manual tagging. While visual recognition of semantics is a mature field, recognition of style is less well explored and perhaps more challenging, since style perception is more subjective. Given the appropriate domain-specific features, our approach could easily generalize to other kinds of style similarity, e.g., pictures, fonts, graphic designs, architectural elements, etc. We believe that training from data on the human perception of style is a promising and general approach to this problem.

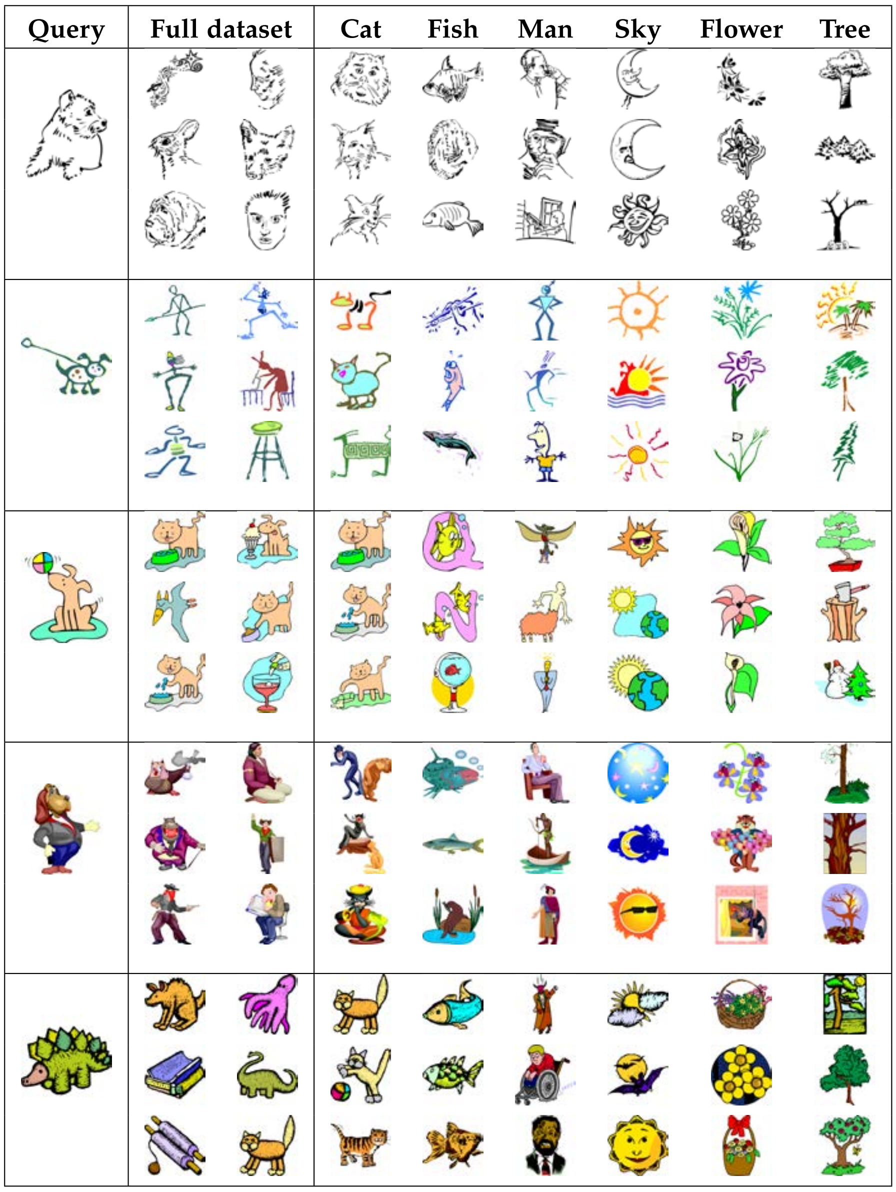


Figure 5.12: *Style-based search. The leftmost artwork is the query image. The next column shows the most similar images in the full dataset of 200k images. The remaining columns show the top three search results in six different categories.*

APPENDICES

5.A ADDITIONAL DETAILS ON FAILURE CASES

24 triplets where our metric disagree with Turkers (Figures 5.13 and 5.14).

5.B ADDITIONAL DETAILS AND RESULTS FROM THE MASH-UP EVALUATION

(Pages 6-8) We performed two tests: the first one was an open-ended task, where users were free to select background, the story and the number of images to use (see Figure 5.15). The second one was guided: we required at least 4 pieces of clip art in each composition, and we prevented the same user from working on the same story twice (see Figures 5.16 and 5.17).

Our pool of stories were the following:

- Show summer season at the lake. Consider including animals, plants, weather elements and/or human activities
- Show the world after an alien invasion
- You are looking at the sky through your window. Include elements you might see in the sky
- Create a poster for a gangster movie, or a scene of the movie
- Create the poster for a scary movie, or a scene of the movie
- Wildlife: Picture the jungle or a forest, and some interaction between animals
- Choose any character (keyword: "face") and show him/her experiencing one or more desires or feelings
- Illustrate the following situation: it is your birthday, and you are feeling very happy with all the presents

In both studies: time was limited to eight minutes; we paid $0.30 per task; and we offered a bonus of $1 for the best compositions.

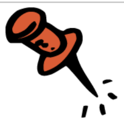
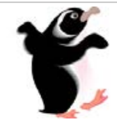
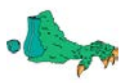
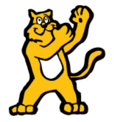
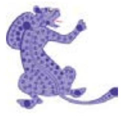
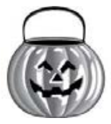
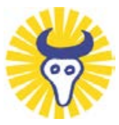
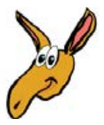
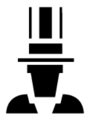
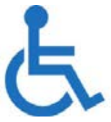
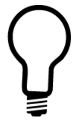
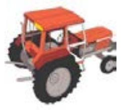
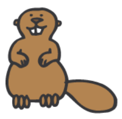
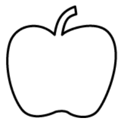
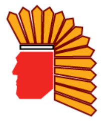
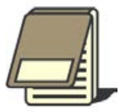
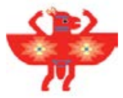
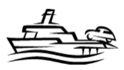
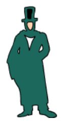
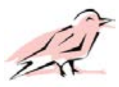
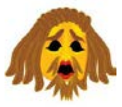
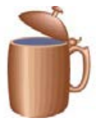
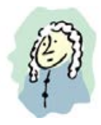
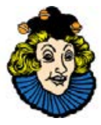
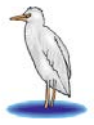
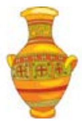
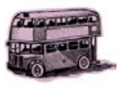
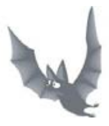
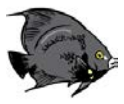
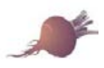
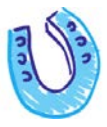
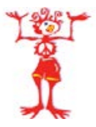
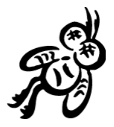
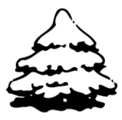
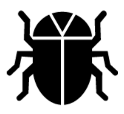
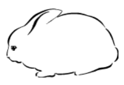
A	B	C	#AB	#AC	Agreement	Probability
			3	8	0.189	0.895
			10	1	0.19	0.164
			5	6	0.196	0.947
			9	1	0.201	0.172
			6	5	0.204	0.0569
			1	9	0.21	0.82
			5	6	0.214	0.937
			2	9	0.219	0.837
			2	9	0.221	0.835
			6	5	0.236	0.0756
			3	7	0.237	0.863
			4	6	0.246	0.896

Figure 5.13: *Testing triplets with greatest disagreement between Turkers and our learned similarity. The triplets are sorted by "agreement," which is equal to the probability of the Turk scores according to the learned model; agreement of 1 would be perfect agreement. #AB is the number of Turkers who rated image A as more similar to B than to C; #AC is the number of Turkers who rated A as more similar to C. The "probability" column shows the model's probability that A and B are more similar than B and C.*

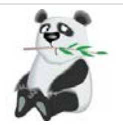
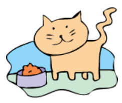
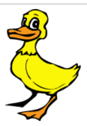
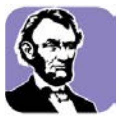
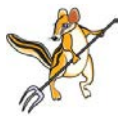
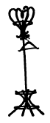
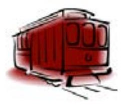
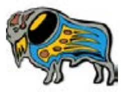
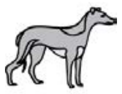
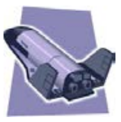
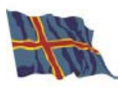
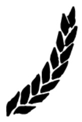
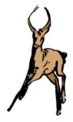
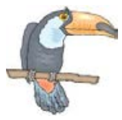
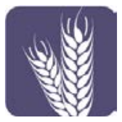
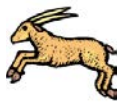
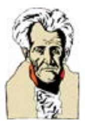
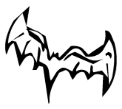
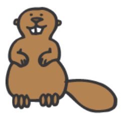
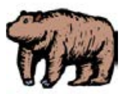
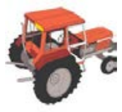
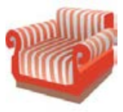
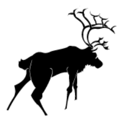
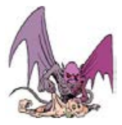
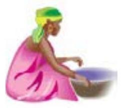
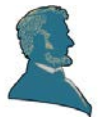
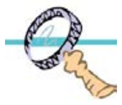
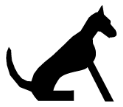
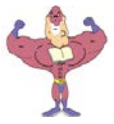
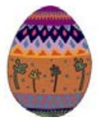
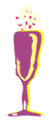
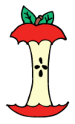
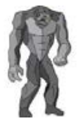
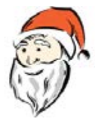
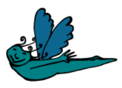
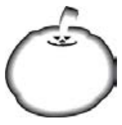
A	B	C	#AB	#AC	Agreement	Probability
			1	9	0.252	0.777
			3	7	0.259	0.843
			3	7	0.262	0.841
			6	5	0.262	0.944
			12	0	0.269	0.269
			4	7	0.281	0.851
			9	2	0.281	0.224
			7	4	0.282	0.15
			8	2	0.284	0.22
			2	9	0.291	0.766
			5	6	0.292	0.884
			3	8	0.302	0.79

Figure 5.14: *The next 12 testing triplets with greatest disagreement between Turkers and our learned similarity. The triplets are sorted by "agreement," which is equal to the probability of the Turk scores according to the learned model; agreement of 1 would be perfect agreement. #AB is the number of Turkers who rated image A as more similar to B than to C; #AC is the number of Turkers who rated A as more similar to C. The "probability" column shows the model's probability that A and B are more similar than B and C.*

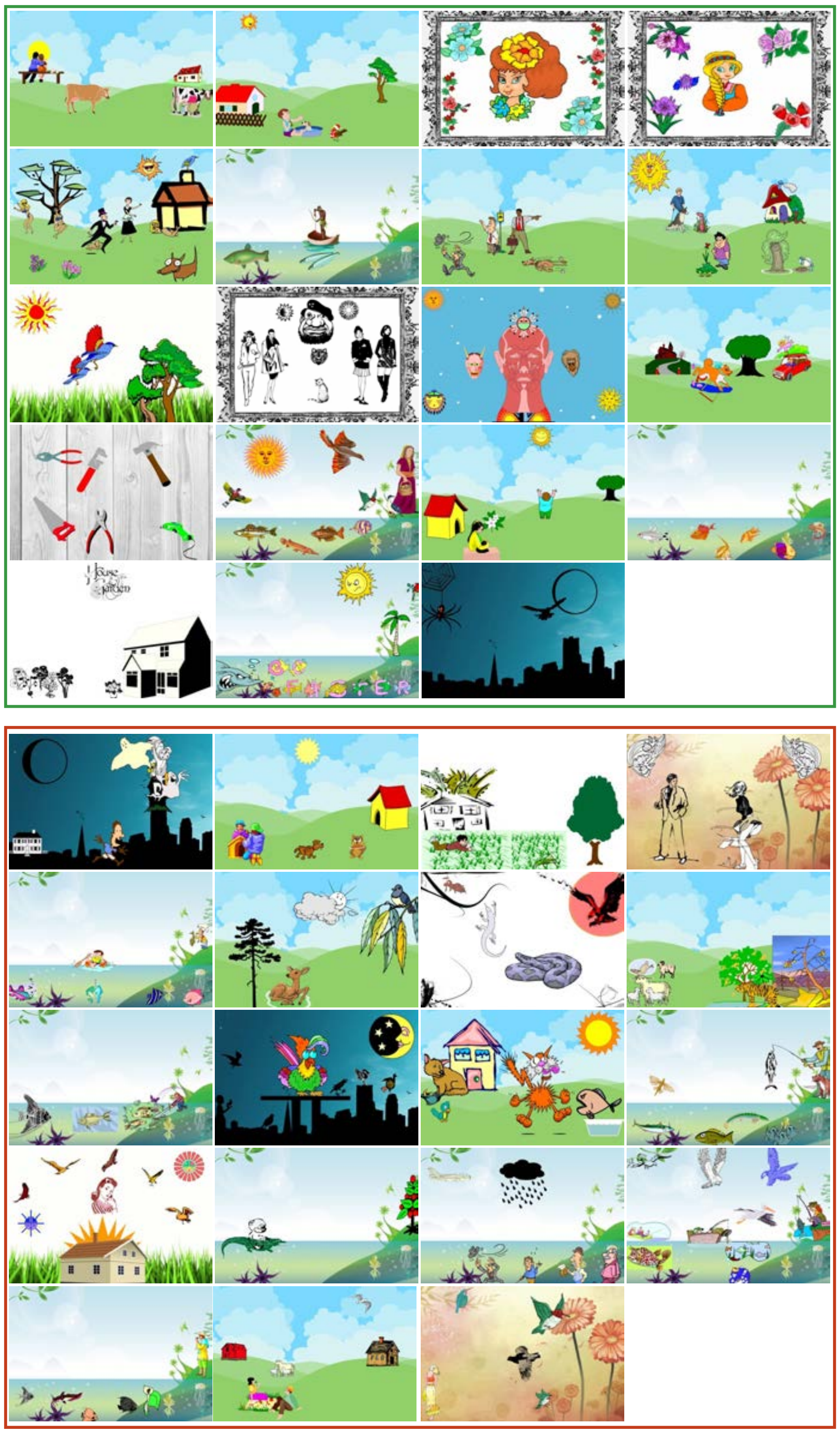


Figure 5.15: ***Open-ended task***: *The upper block shows compositions created using the similarity metric. The lower block shows compositions created without it.*

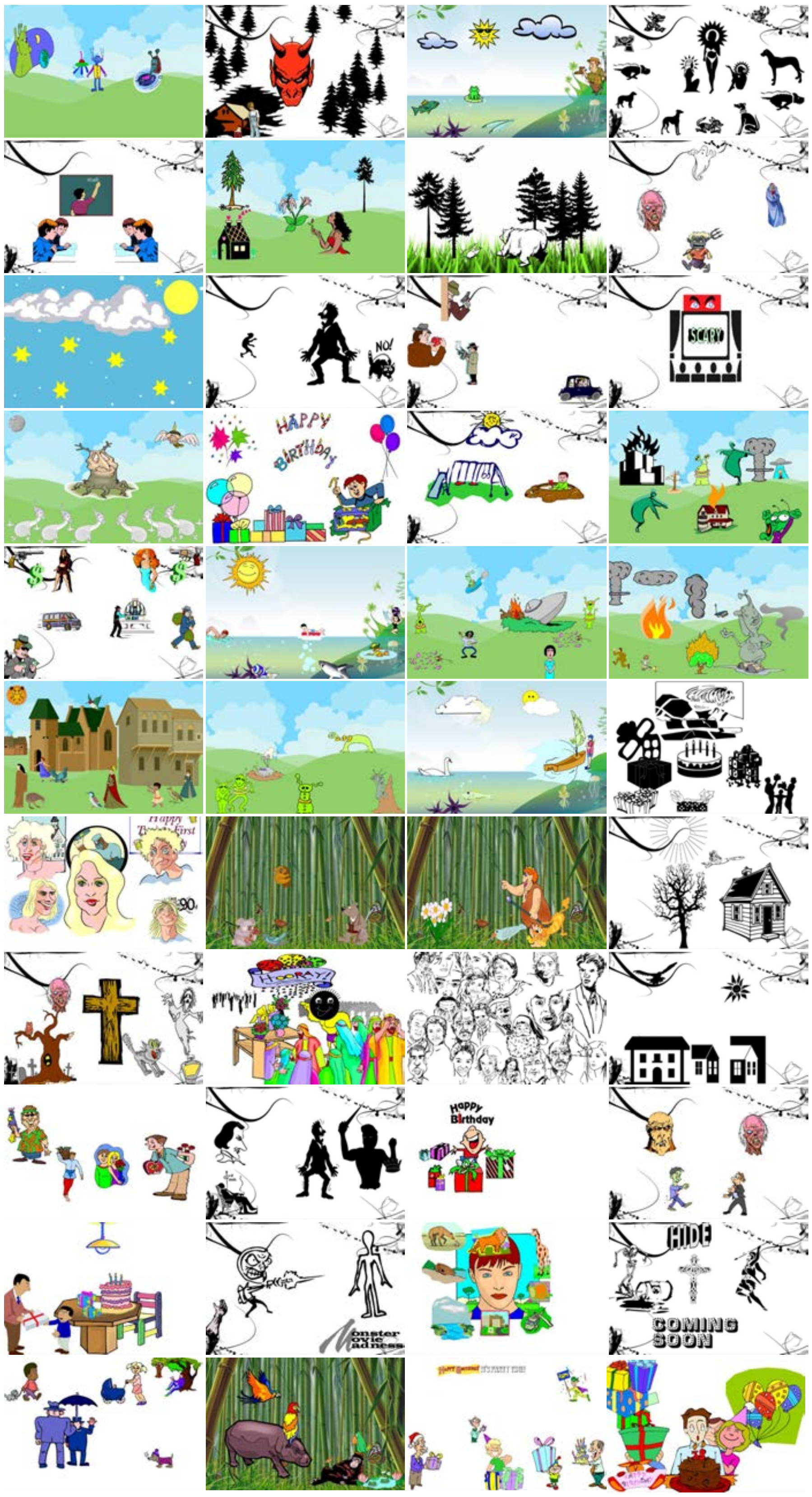


Figure 5.16: ***Guided task***: *Compositions created with the similarity metric turned on.*

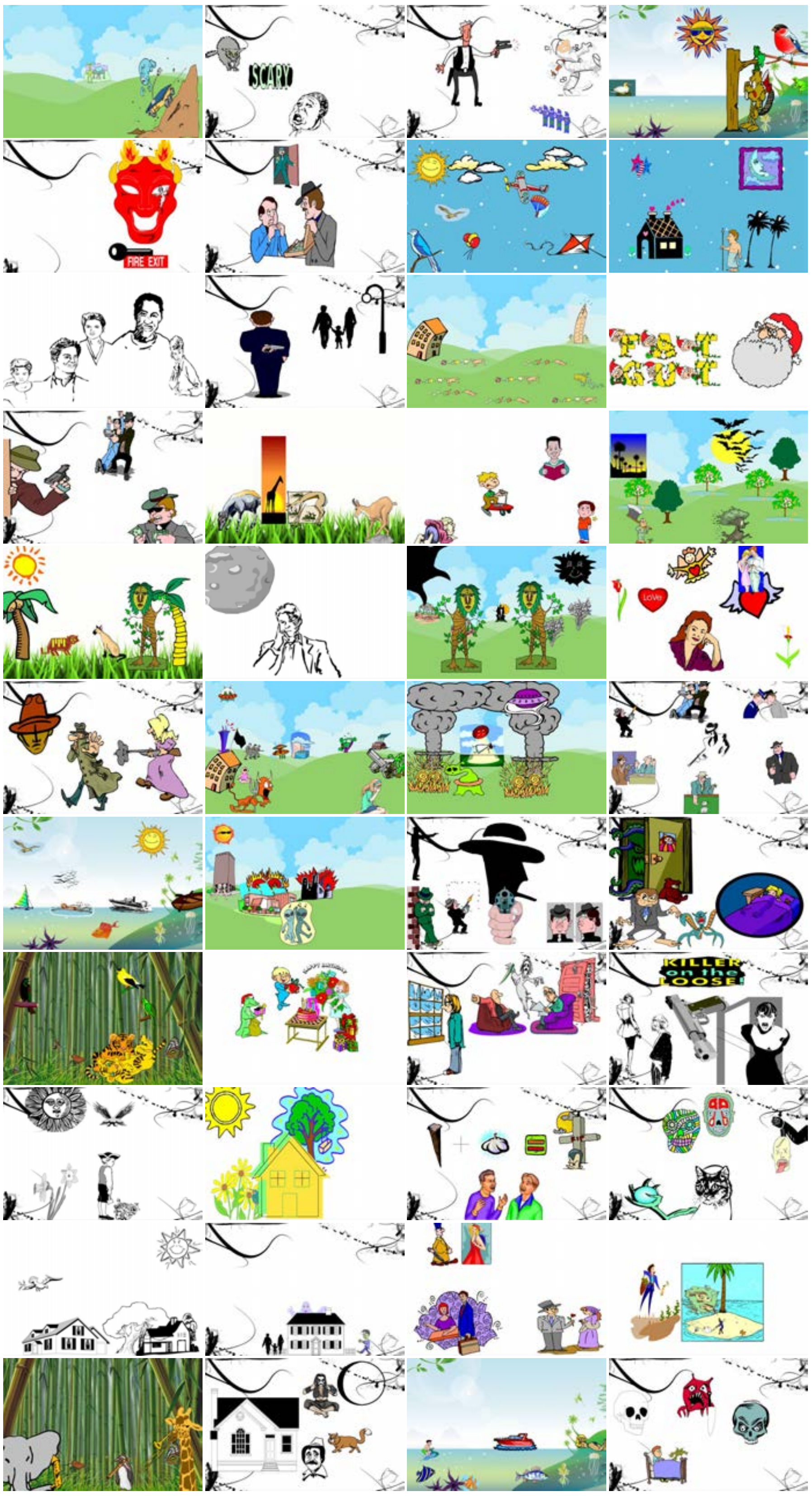


Figure 5.17: ***Guided task***: *Compositions created with the similarity metric turned off.*

6 STYLE-BASED EXPLORATION OF ILLUSTRATION DATASETS

In this chapter, we propose several contributions towards a better comprehension of illustration style and its usefulness for data exploration and retrieval. First, we provide new insights about how we perceive style in illustration. Second, we evaluate a handmade style clustering of clip art pieces with an existing style metric to analyze how this metric aligns with expert knowledge. Finally, we propose a method for efficient navigation and exploration of large clip art data sets which takes into account both semantic labeling of the data and its style. Our approach combines hierarchical clustering with dimensionality reduction techniques, and strategic sampling to obtain intuitive visualizations and useful visualizations.

This work is published in *Multimedia Tools and Applications* 2016.

E. Garces, A. Agarwala, A. Hertzmann & D. Gutierrez
Style-Based Exploration of Illustration Datasets
Multimedia Tools and Applications (Springer), 2016

6.1 INTRODUCTION

The amount of visual information available online has increased dramatically during the last years. In particular, clip art collections contain massive amounts of images which are usually classified by content. While a semantic classification is undoubtedly needed for searching tasks, a style-based exploration might result extremely helpful in certain situations, for example, to create visually coherent presentations, web content, or any kind of graphic design. Occasionally, we find these images labeled by the designer, although the number of images in each group is usually very small and they cover a specific subject e.g. animals, food,... Having unambiguous style labels would be of great help in situations where we need to explore hundreds of images from multiple collections. The problem is that the subjectivity in the perception of style makes finding this labeling a very difficult task. The *style metric* of Garces et al. [53] contributed to quantify certain properties of style, such as line properties or shading, however there is little knowledge about how we distinguish between different styles and how these metrics can be used to guide users in style-based interactions.

In this work, we aim to obtain an efficient method to explore large collection of clip art data sets by style. First, with the purpose of better understanding the perception of illustration style and its correlation with the metric, we perform a user study in which we gain new insights on how do people identify similar styles; then, we evaluate the metric using a hand labeled data set. Finally, we propose an exploratory interface which allows users to combine images from multiple collections and to identify the most common styles at first glance. To do so, we leverage the existing metric to automatically discover the implicit style hierarchies existing in such data sets by means of hierarchical clustering and dimensionality reduction techniques.

The chapter is structured as follows: in Section 6.2 we present the related work; in Section 6.3 we briefly describe the style metric of Garces et al. [53] and present our study about how do people identify similar styles. In Section 6.4 we perform the analysis of a labeled data set. The exploratory interface and results are described in Sections 6.5 and 6.6.

6.2 RELATED WORK

Style Analysis. The analysis of artistic style has received much less attention than stylistic rendering. A common way to algorithmically capture the style elements is to learn a *generative* model; that is, a model that learns to create new examples of a style from scratch, such as generating new typefaces [175, 24], and learning 3D hatching styles from examples [86]. Another type of approaches *transfer styles* from examples, such as transferring painting styles [73], photographic styles [6], or curve styles of 2D shapes [109].

Willats and Durand [184] provide an overview of the elements of pictorial style in 2D illustrations. Recently, a computational set of low level features which capture style has been developed for illustration [53] and fonts [138]. Several methods provide style similarity metrics for 3D shapes [66, 112, 119], clothes [186] or infographics [152]. These works provide a distance metric which allow content retrieval based on style, and that can be used in conjunction with any existing algorithm which requires a similarity measure. In particular, in this work we leverage the work of Garces et al. [53], to obtain aesthetically coherent clusters which are exploited to provide meaningfully visualizations of clip art datasets.

Exploration and Visualization of Data Collections. The huge amount of datasets available online has pointed out the necessity of new tools to explore its content in intuitive ways. In particular, the problem of exploring and browsing 3D shapes is currently a hot topic of study. Kleiman et al. [92] introduce the idea of dynamic maps to provide smooth navigation between the elements of 3D datasets. Averkiou et al. [4] propose a combined approach between exploration and synthesis of 3D shapes. A related approach to ours is the work of Huang et al. [76], which provide a method for organizing heterogeneous 3D shape collections based on different distance measures between shapes. They additionally study several tree-based hierarchies to present the data. However, none of these approaches explore the dimensionality of style.

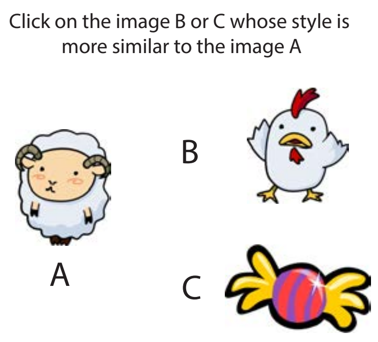
Artistic visualizations of illustration collections have been proposed in the context of packing layouts [150], although unlike us they do not take into account semantic labeling, and just focus on optimal arrangements for fixed layouts. Modeling and navigation through color spaces was introduced by Shapira et al. [161], which fit a Gaussian Mixture Model to the pixel colors of the image. Visualizing high dimensional spaces in two dimensions has been done before for sketches [41], color palettes [137] and 3D models [165].

6.3 ANALYSIS OF STYLE IN ILLUSTRATION

In this section we first briefly review the style similarity metric of Garces et al. [53], and then we present our study of how users identify style in illustration.

6.3.1 *Style Similarity Metric*

Thanks to the *style similarity* metric learned by Garces et al. [53], we can compute a real-valued measure of the similarity in style of two input pieces of clip art, independent of semantics or content. In the original work, the


Click on the image B or C whose style is more similar to the image A
A
B
C

(a)

Attribute	Reason	ID
Contour	No contour	1
	Thick line	2
	Thin line	3
	Irregular/broken line	4
Color	Black&White	5
	Monochromatic or very few colors	6
	Colourful	7
	Similar color saturation	8
Shading	Smooth	9
	Sharp transitions between colors	10
Shape	3D	11
	Flat or two-dimensional appearance	12
	Simple form: few detail	13
	Complex form: lot of detail	14
None	Other reasons	15

(b)

Figure 6.1: *(a) Example of question used to capture style similarity data. Each test (HIT) on Amazon Mechanical Turk contained fifty questions of this kind. (b) List of the available reasons offered to people to choose from.*

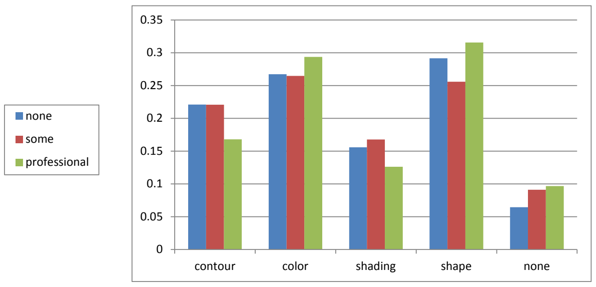
authors demonstrate the usefulness of the metric in search-by-style operations: given a particular query clip art, the search results appear sorted by style similarity. The metric was computed by combining crowdsourcing and machine learning. First, they modeled each clip art image as a high-dimensional feature vector which captures four aspects of style: *color*, *shading*, *texture* and *stroke* [39]. Then, through Amazon Mechanical Turk (MTurk), a set of users were presented with clip art triplets, and were asked the question: "Is clip art A more similar in style to clip art B or C?". Figure 6.1 (a) shows an example of question of such a type. Last, using the feature vector and the relative comparisons collected via MTurk, the style similarity metric, d_S, was learned by computing the Euclidean distance metric [96, 154] (please refer to the original paper for extended details).

6.3.2 *How do people identify similar styles?*

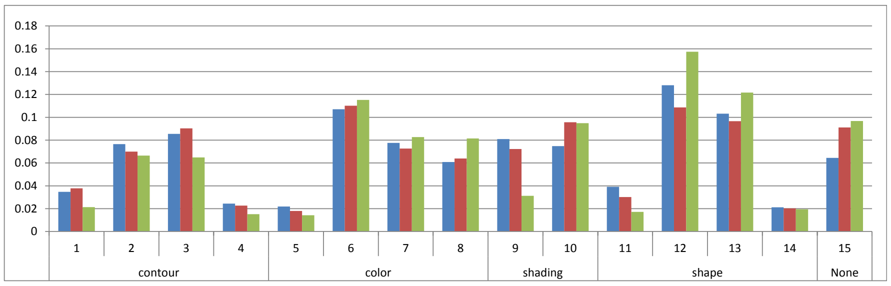
While the *style metric* $d_S$works reasonably well for style-based image retrieval operations, the authors do not offer any insights about what attributes people consider more important when judging if two styles are similar, and thus, the metric could be skipping relevant properties. In this work, we directly asked people what they look at when comparing two pieces by style.

We followed the same MTurk-based methodology as the original work to obtain relative comparisons (see Figure 6.1 (a)), and additionally, we gathered information on the reasons to select one result over another in the performed comparisons. Thus, during the test, each participant would occasionally be asked to choose one or several items out of a proposed set of reasons for picking a result [151]. This questionnaire appeared randomly with a probability of 1 in 10 in each test composed by 50 relative comparisons. Figure 6.1 (b) shows the complete list of proposed reasons grouped by style attribute. In total, 294 people took part on the experiments, where 83 had none artistic experience, 190 had some experience, and 21 were professional artists. We collected 2654 questionnaires of this kind, and grouped the responses by user. We count the number of times a user selects a reason and normalize that number by the total amount of answers per user.

The analysis of the answers is shown in Figure 6.2. It can be seen that the dominant high level attributes that people notice are color and shape. Among color criteria, the most frequent reason is the presence of a domi-



Frecuency of selection of each attribute w.r.t. the total number of answers



Frecuency of selection of each reason w.r.t. the choice of each attribute

Figure 6.2: *Summary of the results from the MTurk questionnaire. Top, cummulative number of selections per attribute, normalized with respect to the total number of answers. Bottom, frequency of occurrence of each reason normalized with respect to the total number of answers per attribute. In both plots, users are separated by artistic experience.*

nant single color, over other reasons like color saturation. Many users identified *shape* as a very important attribute, which is interesting because the style metric $d_S$do not include *shape* in the computed properties. Given that distinguishing between a 3D and a 2D shape is a high level vision task [49], this finding opens new avenues of future work and a big insight about how our visual system discards other style inputs such as shading to favor shape instead. Finally, we did not find any correlation between the artistic background of the users and the particular choices they made, suggesting that we may not need training to perceive style in this kind of artworks. The reason might be that the style of these pieces of art is very well defined and thus, our visual system finds it easy to discern styles with such a high level of stylization.

6.4 analyzing a labeled dataset

In order to analyze how the style metric $d_S$aligns with expert knowledge, we use the data set labelled by style from the Microsoft Office library (now discontinued) which, to our knowledge, is the only free clip art data set labeled according to style. We collected a total of 3024 clip art images of 220 different styles, where the average number of images per style, or style-cluster, is sixteen. We can see in Figure 6.3 a few examples of the initial classification as labeled in the data set, which shows that in principle the labeling makes a good job at discerning styles, with style variations within the same cluster being minimal. An alternative option to use the metric d_S, could have been to use this labeled data set of 3024 images to train a new metric [188, 84, 120]. However, this subset of data only represent a 1.5% of

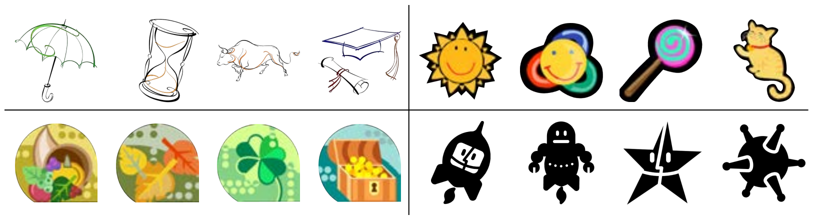


Figure 6.3: *Example of labeled styles from the MS data set. Top-left style is defined by sketchy strokes without fill. Top-right and bottom-left are more colorful styles, the first with black stroke and the second without it. Bottom-right is defined by rounded shapes with black fill and no colors.*

the total amount of data used to train d_S, which was 200k images. Thus, it is expected that $d_S$will generalize better for any other set of images and styles.

6.4.1 *Ranking Evaluation*

In order to evaluate the metric using the labeled data set, and, since the style metric was originally designed for searching purposes, we decided to evaluate the quality of the *ranking* returned after a search operation. That is, for each image of the dataset, which we call the query, we let the metric compute the distances to the rest of the clip art images and rank them from low to high values. At the top of the resulting ranking we expect to find elements with the same style label as the query (true positives) and almost no elements with different labels (false positives); as we traverse the ranked list further down, we expect an increasing number of false positives. To do this, we employ several ranking metrics, which have been adapted to handle labeled data [125]:

AUC Measures the Area Under the ROC Curve. For binary classification problems, the ROC curve measures the amount of false positives against true positives. In our case, this value is computed counting the number of items returned. This metric is position independent, so, an incorrect item at the bottom of the list counts as much as an incorrect item at the beginning.

Precision-at-k (Prec@k) Measures the fraction of relevant results out of the first k returned. This measure is specially relevant when only the first few results matter, as in web browsing or clip art search applications.

Mean Average Precision (MAP) Precision-at-k score of a ranking, averaged over all positions k of relevant items.

Mean Reciprocal Rank (MRR) Inverse position of the first relevant item in the ranking.

Normalized Discounted Cumulative Gain (NDCG) Extension of the MRR metric. In this case all of the top k items are scored at a decaying discount factor.

Figure 6.4 shows the results of these ranking metrics on the initial feature set of Garces el al. [53] with uniform weights (*baseline*), and with the *style metric* d_S. A value of one means a perfect score. As expected, the results are always better for the learned weights and improve when the ordering of the result does not matter, as is the case of AUC and MRR. We can also observe that increasing the value of k for Prec@k and NDCG@k decrease the quality of the results, introducing more false positives. In general, these values are relatively low except for AUC, which does not take into account the relative ordering of the elements and thus, performs quite well. As we

	BASELINE	LEARNED WEIGHTS
		[Garces 2014]
AUC	0.936	0.944
MAP	0.365	0.399
MRR	0.765	0.779
Prec@k=3	0.605	0.628
Prec@k=10	0.470	0.499
NDCG@k=3	0.614	0.638
NDCG@k=10	0.518	0.545

Figure 6.4: *Ranking measures between the unweighted features (Baseline) and the style similarity metric [53]*

will see in the following, the huge similarity between the styles of the hand-made labels has heavily penalized these metrics.

Figure 6.5 shows some of the ranking results where the first item on the left is the query image and the other nine items are the most similar images according to the *style metric*. The images that do not belong to the same cluster as the query (as originally labeled) are highlighted in red. At first glance, we could say that the metric is performing poorly, however, a closer visual inspection of these elements reveals that the styles of the retrieved images are quite coherent. This indicates that many of the images initially labeled as different styles are actually very close in the feature space learned by the metric; and thus, they are *perceived* as similar styles. Although the *intra-cluster* distance is very small, which is a desired property, the *inter-cluster* distance is also sometimes very small. A disjoint labeling like this one do not capture the overall styles available in the data set and thus, would hinder style-based navigation task. In the following section, we propose a exploratory interface based on clustering which will allow to overcome this limitation by showing the most frequent styles available in the data set.

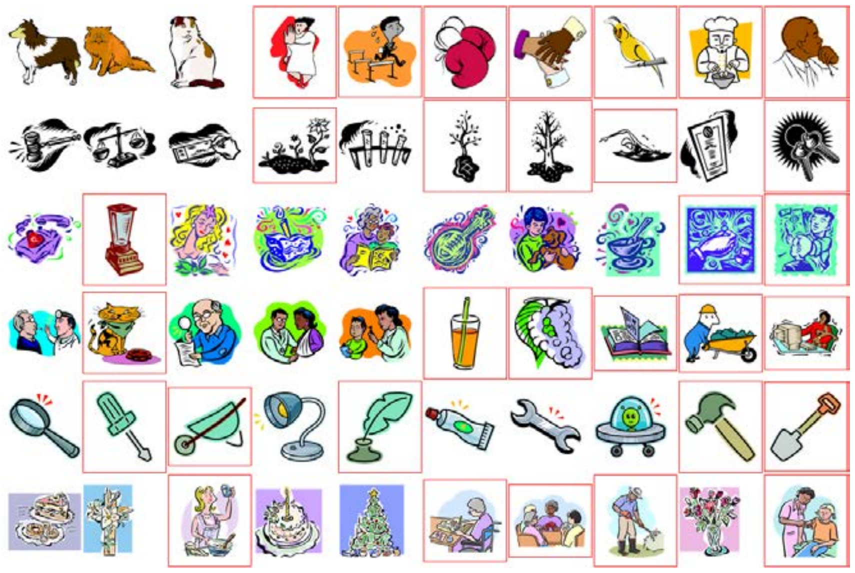


Figure 6.5: *Ranking results. The first image on the left is the query image, the remaining images are sorted by lower to high distance to it. We have used the style metric of Garces et al. [53] to obtain the distances.*

6.5 style-based exploration of illustration datasets

As we have seen in the previous section, we may find several occasions where images of the same style do not necessarily have the same style label, either because they come from different collections/artists or because the labeling is not accurate. Therefore, one of the problems that we face when trying to find elements in these collections is the prior lack of awareness of the available styles in the data set. This is specially problematic when we need to select more than one images stylistically coherent i.e. we may find that the requested images are not depicted in the desired style.

Suppose a user needs to find in certain data set several images that match in style. Each image has to belong to a different semantic category, e.g., a dog, a cat and a tree. We denote as $\mathcal{L}$ the total set of categories or semantic *labels* requested by the user, where in this example $\mathcal{L} = \{dog, cat, tree\}$. In a typical workflow, the user would start querying the data set sequentially by category watching that the style of all the images matches. We can make this task easier by sorting the retrieved results by style similarity to the previously selected items using the metric d_S. In this second scenario, the more delicate step is to select the first image, since its style will define the order in which the remaining queries are sorted. The problem is that if the style chosen in the first place is unusual in the data set, the user may end up disappointed for not having enough stylistically similar images of all the requested categories. We propose a solution to this problem in which the user knows in advance the amount of feasible visual styles which are available in the data set. Our approach combines unsupervised clustering techniques along with adaptive dimensionality reduction and mapping to sample and visualize the images of these huge collections in an efficient and practical interface.

Given a collection of semantically labeled images, which was previously filtered and labeled by the set of semantic categories $\mathcal{L}$ specified by the user, we first perform hierarchical clustering over the filtered collection. Since visualizing the resulting tree may be impractical, we propose two complementary visualizations. On the one hand, we visualize only the top level nodes of the resulting hierarchy in the form of a taxonomy of styles (Section 6.5.1). This gives us the most frequent styles in the set. On the other hand, we use dimensionality reduction techniques with adaptive relocation to visualize in two dimensions the high dimensional space of the style features, allowing the user to navigate through such space (Section 6.5.2).

6.5.1 *Creating a Style Taxonomy*

In order to facilitate data exploration and obtain meaningful taxonomies of the style elements, we have chosen an approach based on hierarchical clustering. Following a bottom up approach, each element initially starts on its own cluster, and, on successive iterations, clusters are merged according to a chosen criteria. In this work, we have chosen Ward's criteria [181], for which, at each step of the algorithm, the pair of clusters with the minimum variance within-cluster are merged. A typical problem with clustering is to choose the appropriate distance metric that captures the underlying relationships between the elements. In our case, we rely on the style metric d_S, which comprises users' knowledge for style discrimination, and has been extensively evaluated.

From this step we obtain a hierarchical tree, which has two types of nodes: intermediate nodes and leaf nodes. Leaf nodes are the lower level elements of the hierarchy and are represented by their corresponding images. Intermediate nodes define *branches* of style and may contain leaf nodes and other intermediate nodes. To select the representative image of an intermediate node, we choose the leaf-node image with the closer distance to its centroid in the metric space of d_s.

REPRESENTATIVE STYLES In huge data sets it may not be possible to show all the hierarchy in one view. Therefore, we need to develop a strategy to sample the hierarchy and select a representative number of intermediate nodes so that we show as much information as possible about the available styles. Our solution consists of pruning the branches that do not satisfy the following conditions: 1) the total number of images of the branch is above a certain number κ; 2) the number of images of each semantic category is greater than a value τ . The first criteria balances the taxonomy so that all the nodes contain the same number of images, the second criteria allows to discriminate feasible paths. If a certain node does not meet these two criteria, we remove the node and its branch. We then take the resulting leaf nodes of the pruned hierarchy as representative styles. By changing the values of the thresholds we control the size of the resulting hierarchy and guarantee to capture with a few nodes the most frequent styles of the data set.

To illustrate the process, we applied it to a set of 3024 images of a wide variety of styles from the Microsoft Office online libraries. We show in Figure 6.6 the top level nodes of the resulting hierarchy after the pruning and the representative styles. We observe that the most dominant style has colorful images without contour -five representative nodes in the middle capture this style. On the contrary, the less frequent style is compound by black and white images- only one node capture this style. Depending on the amount of data that we want to visualize, we vary this threshold. In Figure 6.6, we want every representative style to have 10% of the total data ignoring semantic labeling, so $\kappa = 0.1 \cdot 3024$ and $\tau = \kappa$.

6.5.2 *2D Exploration: Adaptive arrangement*

The style taxonomy resulting from the previous step makes it possible to visualize at high level the diversity of styles available in the data set. Each leaf node of the taxonomy represents a *style branch*, which the user can further explore to find the desired icon sets. Since each of these branches might contain too many images to visualize as is, we propose a two dimensional grid-based visualization where Euclidean distances between the images in the grid are equivalent to distances in the perceptual metric space of d_s. The optimal arrangement needs to satisfy the following conditions:

1) Fixed layout and number of slots. If the selected branch has more images than available slots in the grid, some images will need to remain hidden in the main view. According to this condition, interface designers can select the optimal size of the grid to fit their interface and data requirements.

2) Balanced number of categories. If the user requests more than one semantic category or label, the amount of *visible* images for each label should be approximately the same. The task of finding the requested

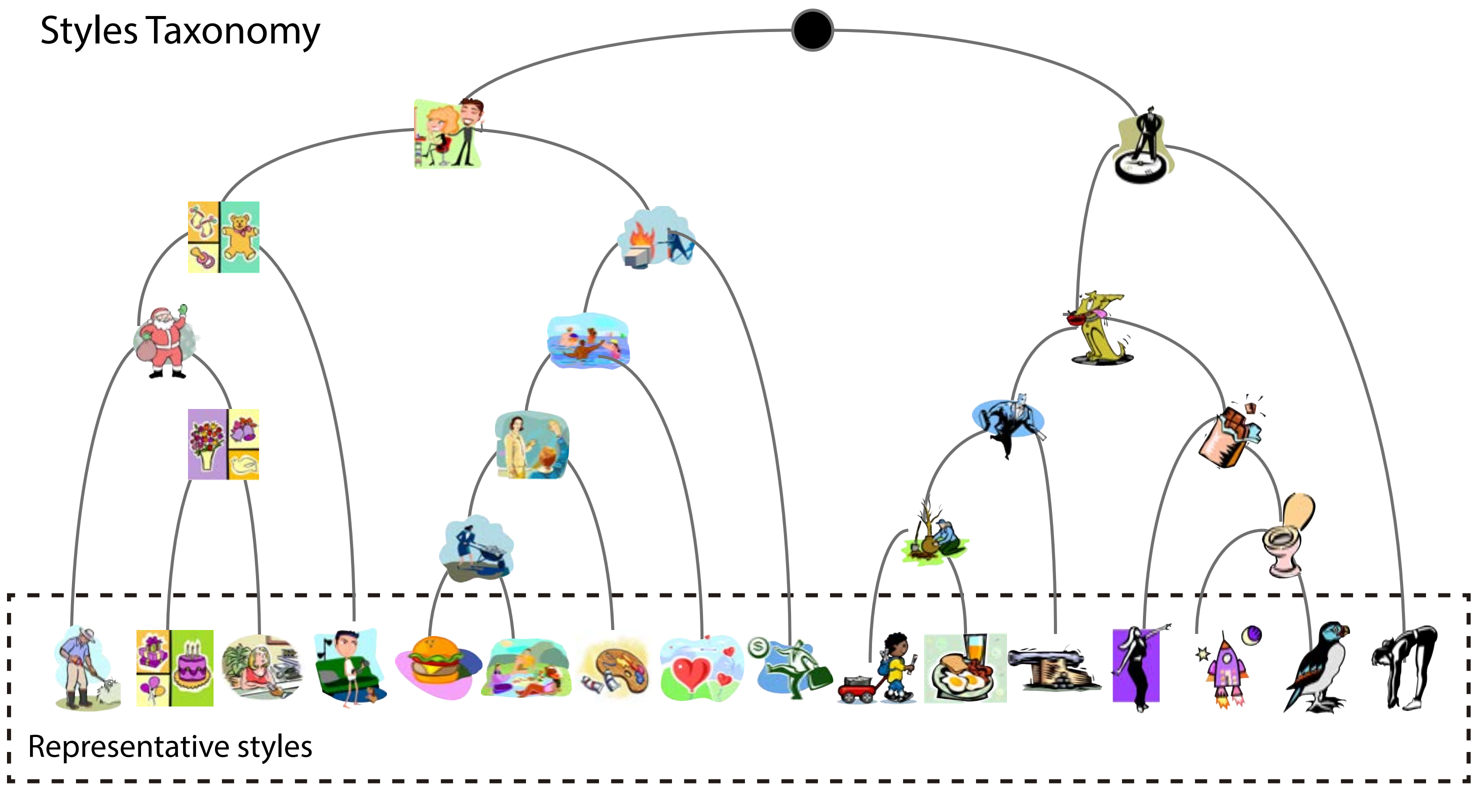


Figure 6.6: *Top level nodes of the style taxonomy with* $\kappa = 0.1 \cdot 3024$. *Each of the representative style branch contains around 10% of the total data. Note that black and white style (bottom-right node) is less frequent in the data set than colorful, stroke-less style (nodes five to nine in the middle).*

set of images is easier if the user can visualize most of the images at first glance.

3) Perceptual distances. Images with similar style should remain close to each other in the two-dimensional arrangement. This kind of arrangement makes it easier the exploration of big sets of images which may contain multiple style variations.

With the above conditions, it is unsuitable to use existing methods of layout generation [48, 135]. They do not provide support for fixed layouts, where there are more images than slots (condition 1), and do not take into account the balance of the labels (condition 2). Thus, our goal is to optimally place in a grid the set of images from a style branch according to the three previous conditions. We propose a greedy algorithm which keeps balanced the number of displayed labels, maintains the perceptual distances in style in the grid, and takes into account special situations where there are more images than positions in the grid, i.e., situations in which one position in the grid may contain multiple images.

PROBLEM FORMULATION We consider a regular grid $G = \{X\}$ of size $M \times N$, the set of images of a branch $\mathcal{I}$ which we want to arrange in G, and a set of semantic labels $\mathcal{L}$. Each image $i \in \mathcal{I}$ has a label from $\mathcal{L}$ associated, and the function label(i) returns it. Each node $x \in G$ defines a tuple $x = (i, l, \mathbf{h}, s)$, where: $x^i \in \mathcal{I}$ is the image chosen to represent the node; $x^l \in \mathcal{L}$ is the semantic label of such image; $x^{\mathbf{h}}$ is the list of candidate images assigned to the node (which excludes x^i) and $x^{\mathbf{h}(k)}$ denote the kth image in such vector; x^s is the state of the node, where $x^s \in \{single, multi, empty\}$. We define a function $o(x) = |x^{\mathbf{h}}|$ that counts the total number of candidate images per node, and a function $o_h(x, a) = |x^{\mathbf{h}_a}|$ that counts the number of candidate images per node x with a given label a. Each state x^s is defined as follows:

- $x^s = \text{single} \iff i \neq \emptyset, l \neq \emptyset, o(x) \geq 0$
- $x^s = \text{multi} \iff i = \emptyset, l = \emptyset, o(x) > 0$
- $x^s = \text{empty} \iff i \neq \emptyset, l \neq \emptyset, o(x) = 0$

where single and multi nodes differ in that single nodes have their representative image defined, and multi nodes do not. Note that *multi* nodes just define transitory states where there are multiple candidates for the node but a representative image has not been selected. A single node where $o(x) > 0$ contains *hidden* images that can be shown, for example, in auxiliary panels or pop-ups.

INITIALIZATION We initialize G with the output of SOM (Self-Organizing Map) algorithm [93] applied to the set of images $\mathcal{I}$ with a grid size $M \times N$. This method finds a mapping between the n-dimensional feature space of the images and the 2D grid. In our case, we use the 169-dimensional style feature vector [53]. The SOM algorithm performs unsupervised training of a neural network to obtain a weak classification of the data. This method is suitable for grid structures and has been successfully used to display continuous color palettes [135]. As a result, we have a mapping of each image on the grid where the perceptual distances in the feature space are well preserved. However, this mapping has three problems: 1) it may contain unassigned nodes (x^{empty}); 2) the mapping of the labels might not meet the balance criteria, i.e. the amount of images of each label in *single* nodes is not the same; and 3) it may have multiple images assigned to the same position of the grid. When this third situation happens, we set the node to *multi* state, and leave the representative image and label empty. Our goal is to find the optimal arrangement of G, so that it meets this three requested conditions, or similarly, the following energy is minimal:

$$\min_{G} |EN| + \sum_{a,b \in \mathcal{L}, a \neq b} |\gamma(a) - \gamma(b)| + |MN| \tag{6.1}$$

where $\gamma(a) = |\{x | x \in G \wedge x^s = \text{single} \wedge x^l = a\}|$ is the number of *single* nodes in G with label a ; $EN = \{x | x \in G \wedge x^s = \text{empty}\}$ is the set of empty nodes, and $MN = \{x | x \in G \wedge x^s = \text{multi}\}$ is the set of *multi* nodes. Solving the optimal arrangement can be shown that is an NP-hard problem, therefore, we propose a greedy efficient solution which works sufficiently well for our purposes. In a first step, we turn all the multi nodes into single nodes ($x^{\text{multi}} \longrightarrow x^{\text{single}}$) by selecting the optimal representative image among its candidate list. In a second step, we turn the *empty* nodes into *single* nodes ($x^{\text{empty}} \longrightarrow x^{\text{single}}$) by taking *hidden* images from neighbor nodes in the grid.

STEP 1: ARRANGEMENT OF MULTI NODES ($x^{\text{multi}} \longrightarrow x^{\text{single}}$)

The naïve solution to turn every *multi* node into *single* node is to randomly choose one of the candidate images in $x^{\mathbf{h}}$. This is an optimal option if we only have one label in $\mathcal{L}$, however if we have more than one label, we need to choose the image in a more principled way to guarantee the balance condition. Our strategy is the following: first, find the label with minor occurrence in *single* nodes; second, find a suitable *multi* node which contains that label in the list of candidates $x^{\mathbf{h}}$; and third, set as representative image any appropriate image from the candidate list $x^{\mathbf{h}}$ of the chosen node. In more detail, in each iteration, we find the set of minority labels in *single* nodes (line 4); then, sequentially for each of these labels, we find a suitable *multi* node which contains such label in $x^{\mathbf{h}}$ (line 5). If, for a certain label, we do not find a *multi* node which contains it, we remove such label from the candidate label set $\mathcal{L}_C$ (line 7). For finding the optimal *multi* node, we give priority to the nodes with smaller number of candidates (line 10). This criteria aims at maximizing the chances to find a suitable *multi* node for the next label at every step.

STEP 2: ARRANGEMENT OF EMPTY NODES ($x^{\text{empty}} \longrightarrow x^{\text{single}}$)

The process to turn empty nodes into multi nodes is similar as before. We aim to fill empty nodes with the less frequent labels l_s, so we start by sorting the labels by frequency of appearance (line 4). Since empty nodes do not have a candidate list, we need to find a suitable image from its neighborhood nodes' candidate lists. We start by computing the list of nodes $M(l)$ which contain the chosen label l in their candidate list (line 6). Then, we explore its 5×5 neighborhood $\mathcal{N}_x$ in the $M \times N$ grid, and create the candidate list $S(l)$ with the nodes within $\mathcal{N}_x$ which are empty (line 7). If $S(l)$ is empty, there are no hidden images with the requested label, so we remove the label from $\mathcal{L}_C$ (note that candidate images in a single node become *hidden* images after step 1). Otherwise, we select the empty node in $S(l)$ which has the fewer amount of candidate images in its neighborhood (line 12). To select the image to assign, we find the neighbor node with fewer candidates available and take any suitable image.

Algorithm 3 Step 1: Arrangement of Multi nodes

```
1: Data $G = \{X\}, \mathcal{L}$
2: $\mathcal{L}_C \leftarrow \mathcal{L}$
3: while $\exists x^{\text{multi}} \in G'$ do
4:     $\{l_s\} \leftarrow \min_{l \in \mathcal{L}_c} \gamma(l)$
5:     for all $l$ in $l_s$ do
6:         if $S(l) = \emptyset$ then
7:             $\mathcal{L}_C \leftarrow \mathcal{L}_C - \{l\}$
8:         else
9:                                   ▷ Find optimal multi node in $S(l)$
10:            $x_j \leftarrow \min_{x \in S(l)} o(x)$
11:                     ▷ Update node values: state, image, label and list of
    candidates
12:            $x_j^s \leftarrow$ single
13:            $x_j^i \leftarrow x_j^{\mathbf{h}(k)}$ : label$(x_j^{\mathbf{h}(k)}) = l$
14:            $x_j^l \leftarrow l$
15:            $x_j^{\mathbf{h}} \leftarrow x_j^{\mathbf{h}} - \{x_j^{\mathbf{h}(k)}\}$
16:        end if
17:    end for
18: end while
```

Algorithm 4 Step 2: Arrangement of Empty nodes

```
1: Data $G = \{X\}, \mathcal{L}$
2: $\mathcal{L}_C \leftarrow \mathcal{L}$
3: while $\exists x^{\text{empty}} \in G \wedge \mathcal{L}_C \neq \emptyset$ do
4:     $\{l_s\} \leftarrow \min_{l \in \mathcal{L}_c} \gamma(l)$
5:     for all $l$ in $l_s$ do
6:         $M(l) = \{x | o_h(x, l) > 0\}$
7:         $S(l) = \{x | x^{\text{empty}} \wedge x \in 8\mathcal{N}_{x'} \wedge x' \in M(l)\}$
8:         if $S(l) = \emptyset$ then
9:             $\mathcal{L}_C \leftarrow \mathcal{L}_C - \{l\}$
10:        else
11:                    ▷ First, find the optimal empty node $x_j$ in the set $S(l)$
12:            $x_j \leftarrow \min_{x \in S(l)} o_h^{\mathcal{N}}(x) : o_h^{\mathcal{N}}(x) = \sum_{x' \in 8\mathcal{N}_x} o(x')$
13:                 ▷ Second, take the representative image from the neighbor
    node with fewer available candidates. Then, update values
14:            $x_j^i \leftarrow x_k^i : x_k = \min_{x \in 8\mathcal{N}_{x_j}} o(x)$
15:            $x_j^l \leftarrow l$
16:            $x_j^s \leftarrow$ single
17:        end if
18:    end for
19: end while
```

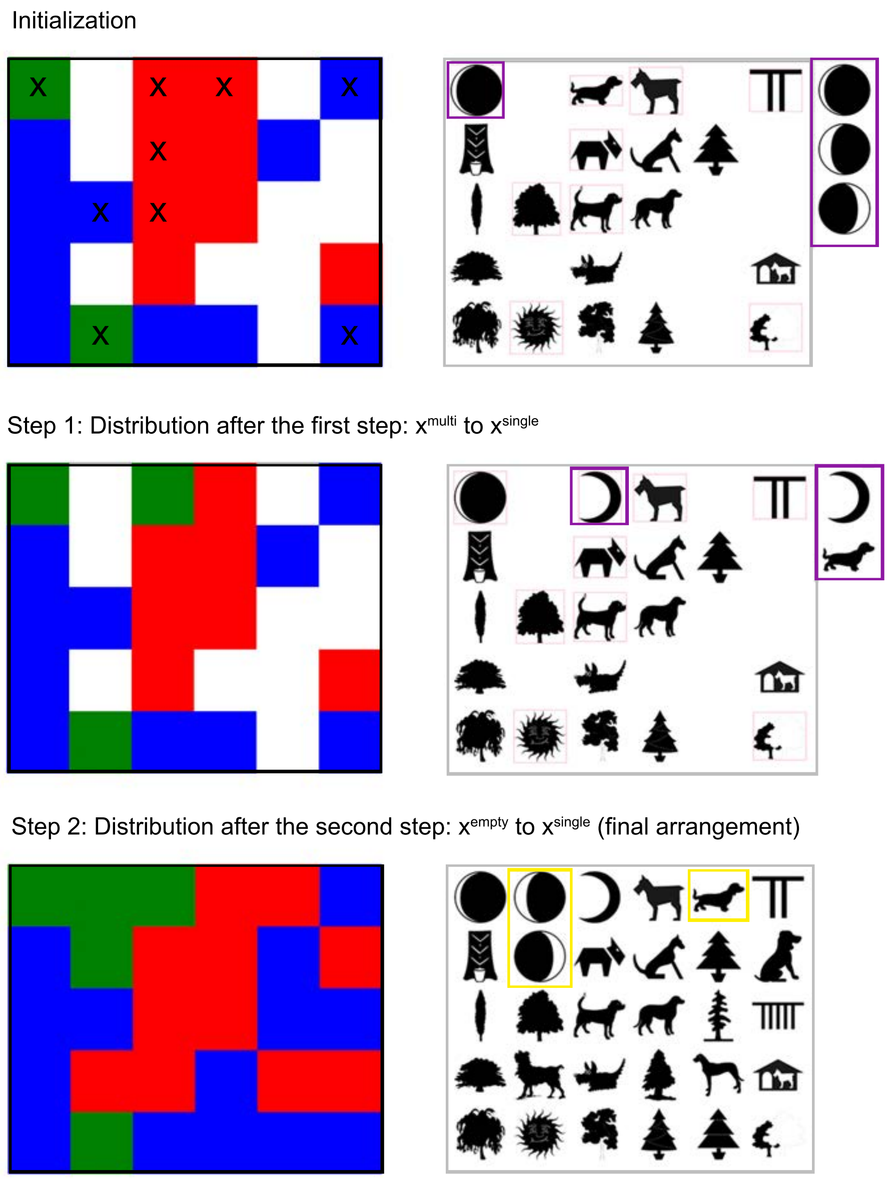

Initialization
X
X
X
X
X
X
X
X
X
Step 1: Distribution after the first step: x^{multi} to x^{single}
Step 2: Distribution after the second step: x^{empty} to x^{single} (final arrangement)

Figure 6.7: *Distribution of labels and images after each step of the algorithm. Left column: labels x^l for dog, sky and tree categories represented by red, green and blue colors respectively. Right column: panel with images x^i. In Initialization and Step 1, this panel includes an additional vertical panel on the right (squared in purple). The images within this panel belong to the corresponding purple node on the main panel, i.e. in these nodes the number of candidate images is greater than zero after both steps. After Step 2, these images have been redistributed to produce a fully occupied grid. The input is a set of 2141 images with three different labels: dog, sky, tree with 1147, 100 and 894 images per label respectively. Note that green label (sky) is the less frequent and yet it has significant representation in the final arrangement.*

The whole process is illustrated in Figure 6.7 for a subset of data and a small grid layout of 5×6. The initialization step shows several empty slots and unequal label distribution. *Multi* nodes are marked with a cross in the top panels; the assignment of labels and representative images for these nodes is done randomly in the initialization. In the second step, the label and images of *multi* nodes are set to maximize the variety of visible labels (balance condition). Finally, in the last step, the holes are filled with hidden candidates. We can see in purple and yellow two examples of this rearrangement. In the final step, all the images are visible while keeping the style arrangement.

6.6 RESULTS AND EVALUATION

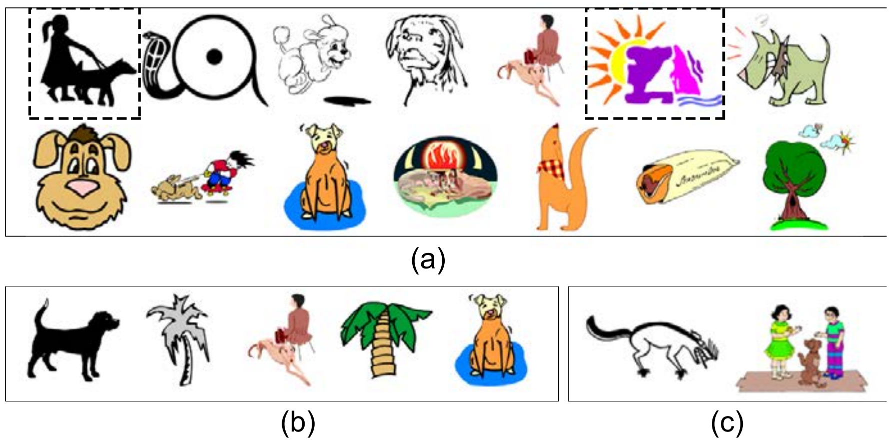


Figure 6.8: *Representative Styles of the* tree-dog-sky *data set. (a) Contains the full data set with* $\tau = 10$ *and* $\kappa = 100$*. In (b) we have reduced the amount of tree elements to 100, while in (c) we have reduced the tree category by the same amount, both with* $\tau = 5$ *and* $\kappa = 50$*. As we can see the representative styles change depending on the available data.*

In the following section, we test and evaluate every step of our approach separately. We first provide several examples of the representative styles for multiple variations of a data set. Then, we provide examples of 2D arrangements and comparisons with related work. Finally, we evaluate the usefulness of our approach with a user study.

We have selected a collection of images, which we name the *tree-dog-sky* data set, which contains 2609 images of three different categories: 568 images of *sky* category, 894 of *tree*, and 1147 of *dog*. After the branch pruning of Section 6.5.1 with parameters $\tau = 10$ and $\kappa = 100$, we obtain a total of 1129 images (376 *sky*, 321 *tree* and 432 *dog*) which yield fourteen representative styles (Figure 6.8 (a)). This step guarantees that for each of these representative styles we have at least one hundred images in total and ten images of each category. In the representative styles we can see that the data covers a great variety of styles: four nodes for black and white styles with different levels of sketchiness, two nodes for colorful styles without contour and seven additional nodes with different variations of shading and complexity. We have also done experiments by randomly removing images from one category. In Figure 6.8 (b-c) we can observe the resulting styles by reducing to 100 the number images of sky and tree categories and $\tau = 5$ and $\kappa = 50$. We can see that the representative styles change depending on the available data.

We have compared our adaptive 2D arrangement with the recent method of Fried et al. [48] and the raw output of Self-Organizing Maps (SOM) [93]. We have selected two of the style branches of Figure 6.8 (a) and mapped its content in a 2D grid (Figure 6.9) of fixed size 7×10. Since Isomatch does not handle grids of smaller size than the number of images, we have randomly sampled a subset of the data to visualize it. In our solution and SOM we mark the nodes with more than one images assigned with a red square. We can observe that our solution and SOM preserves better than Isomatch the style distances in the grid, see for example, that the moon images in the black style are located closed to each other in SOM while the arrangement of these images in isomatch looks random. The same happens with several other clusters of style marked in the figure. Our solution keeps this good behaviour of SOM while producing a fully occupied grid.

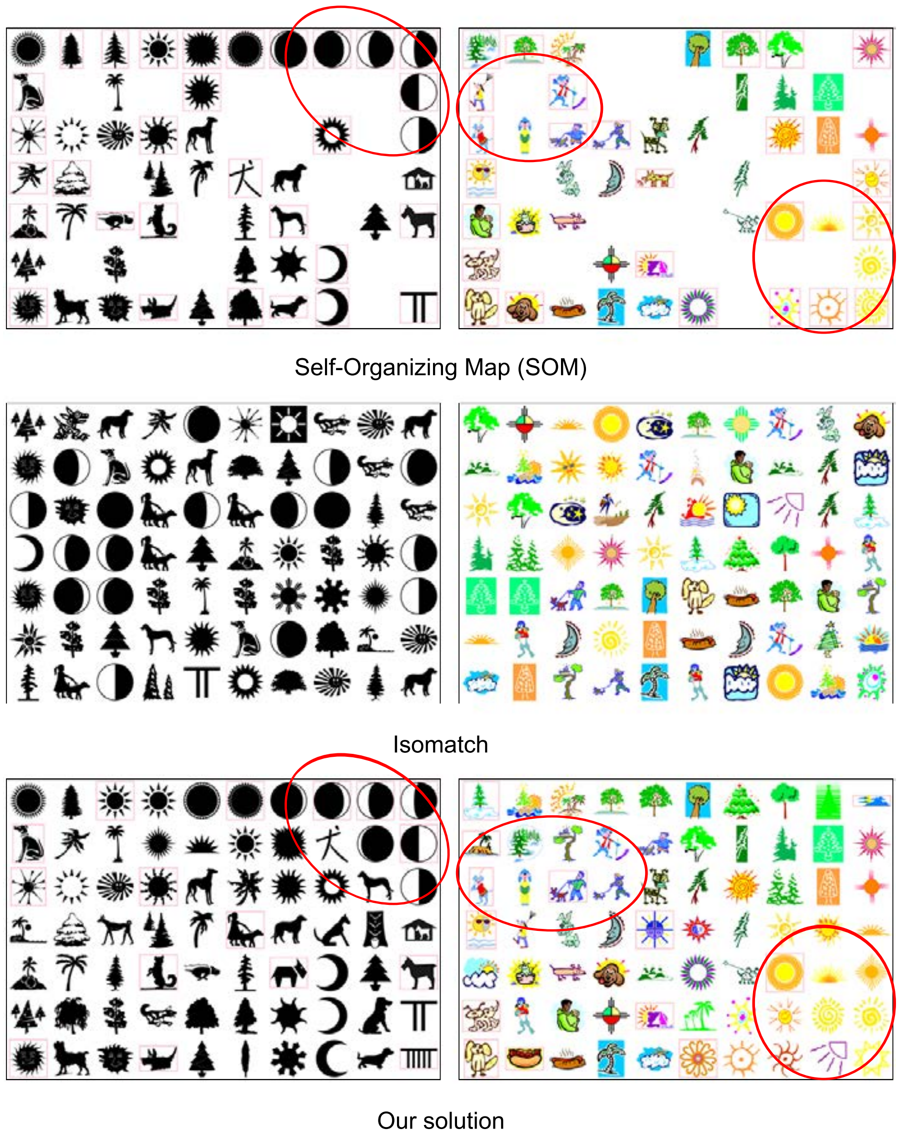


Figure 6.9: *2D arrangement for the two style branches marked with dotted lines in Figure 6.8 (a). Top: Self-Organizing Map [93]. Middle: Isomatch [48]. Bottom: Our solution. We observe clear style clusters in SOM (marked in red) which are also preserved in our solution. The arrangement of these images in Isomatch looks more chaotic.*

EVALUATION We performed a user study to evaluate the impact of our approach on icon selection tasks. In this study, some users were provided with our style-based exploratory interface, and others received a simpler version were the search results were just sorted by style similarity. Users were not aware of this difference, and they just interacted with one version of the interface.

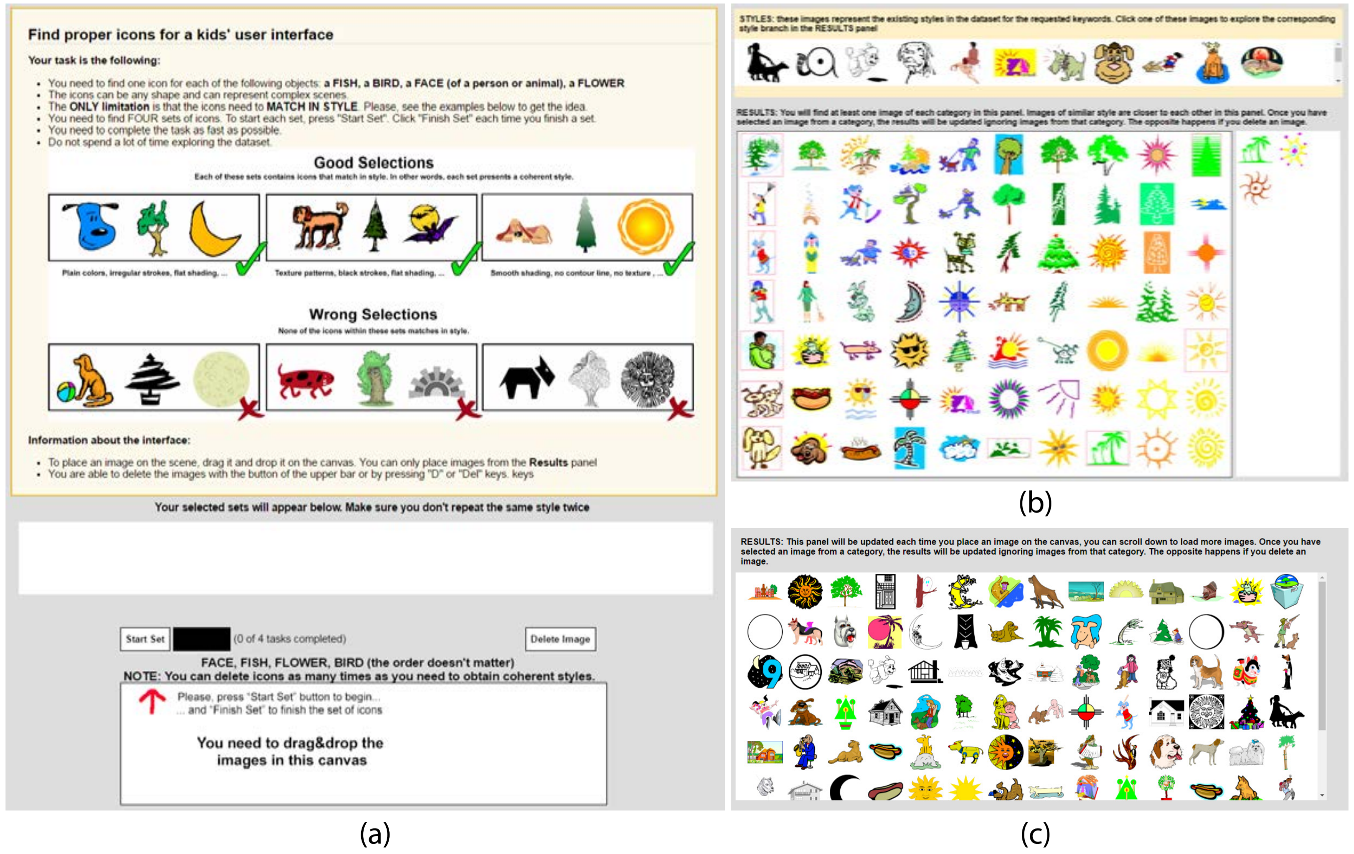
The task was to find a set of icons matching in style for a kids' user interface. We gave them the list of requested categories: *face*, *fish*, *flower* and *bird* (with 937, 1014, 1022 and 1015 images), which they had to find on the data set and place on a canvas. Each time a category was set on the canvas, we automatically eliminated the images of such category on the retrieved results to ease the task in both interfaces. We asked each user to create four sets of icons of four different styles, and at every moment, they were able to see all the completed sets to avoid repeating styles. In each of the requested sets, one of the requested categories had considerably less images than the other three. To avoid bias, the users were not aware of this fact. We specifically selected this scenario to prove that our method can handle such difficult cases, when the data set does not contain all the images in all the styles. There is a screen capture of both interfaces and the instructions in Figure 6.10. Please, see also the accompanying video.

We gathered a total of 64 icons sets, 28 created with our interface and 36 without it. For each interface, we measure the average time to complete each set and the number of delete operations. The average time for each set using our interface was 79 seconds with a confidence interval at 95% of ± 20, and 108 seconds without it , with a confidence interval of ± 27. The average number of images deleted was 0.6 with our interface and 2.8 without it, with confidence intervals at 95% of 1.1 and 1.2 respectively. We additionally captured several comments from the users which tested the basic version that complained about not having enough data of certain styles. From this experiments we can conclude that using our approach, the task of finding optimal icons sets of multiple styles is easier and around 30 % faster than using basic exploration, providing more information and guidance.

6.7 CONCLUSIONS AND FUTURE WORK

In this work we have presented a method to explore and visualize big collections of clip art images according to its visual style. We have relied on an existing style similarity metric which we have evaluated with two novel approaches: by means of a user study, we have learnt that users perceive stroke, shape and color as the most dominant attributes to identify similar styles; by using a labeled data set, we have obtained objective error metrics which measures its quality. Using this metric, we have built a hierarchy which captures the underlying structure of styles existent in a given data set. A combined approach between dimensionality reduction techniques, and strategic sampling of certain nodes of the hierarchy allows to intuitively visualize the styles present in the data set and navigate through them. We have tested and confirmed the usefulness of our approach by means of a user study.

The main problem we have found to capture style is that visual style attributes are very correlated. For this reason, is highly difficult to identify clear categories. An interesting avenue of future work would be to describe the style with relative attributes [143], that is, it is easier to say that an image is very colorful than to classify an image as colorful or not. In this sense, we want to explore fuzzy clustering algorithms to see whether they can capture complex relationships between the visual elements.



Find proper icons for a kids' user interface

Your task is the following:

- You need to find one icon for each of the following objects: **a FISH, a BIRD, a FACE (of a person or animal), a FLOWER**
- The icons can be any shape and can represent complex scenes.
- The **ONLY limitation** is that the icons need to **MATCH IN STYLE**. Please, see the examples below to get the idea.
- You need to find FOUR sets of icons. To start each set, press "Start Set". Click "Finish Set" each time you finish a set.
- You need to complete the task as fast as possible.
- Do not spend a lot of time exploring the dataset.

Good Selections

Each of these sets contains icons that match in style. In other words, each set presents a coherent style.

Plain colors, irregular strokes, flat shading, ... ✓ Texture patterns, black strokes, flat shading, ... ✓ Smooth shading, no contour line, no texture , ... ✓

Wrong Selections

None of the icons within these sets matches in style.

Information about the interface:

- To place an image on the scene, drag it and drop it on the canvas. You can only place images from the **Results** panel
- You are able to delete the images with the button of the upper bar or by pressing "D" or "Del" keys. keys

Your selected sets will appear below. Make sure you don't repeat the same style twice

(0 of 4 tasks completed)

FACE, FISH, FLOWER, BIRD (the order doesn't matter)
NOTE: You can delete icons as many times as you need to obtain coherent styles.

Please, press "Start Set" button to begin...
... and "Finish Set" to finish the set of icons

You need to drag&drop the images in this canvas

STYLES: these images represent the existing styles in the dataset for the requested keywords. Click one of these images to explore the corresponding style branch in the RESULTS panel

RESULTS: You will find at least one image of each category in this panel. Images of similar style are closer to each other in this panel. Once you have selected an image from a category, the results will be updated ignoring images from that category. The opposite happens if you delete an image.

RESULTS: This panel will be updated each time you place an image on the canvas, you can scroll down to load more images. Once you have selected an image from a category, the results will be updated ignoring images from that category. The opposite happens if you delete an image.

Figure 6.10: *(a) Instructions for all users and both interfaces. (b) Our proposed solution. On the top row we can check the available styles. At the bottom, the selected branch is arranged in 2D. In nodes with more than one image, the remaining images are shown on the right vertical panel. (c) Basic interface, where images of the data set are sorted randomly at the beginning, and then, sorted by style similarity to the images of the canvas.*

Part IV

CONCLUSION

7 CONCLUSIONS AND FUTURE WORK

In this thesis we have presented a variety of contributions towards exploring *appearance* and *style* in heterogenous visual data, including real world representations and artistic depictions. Although in both cases, the input data is just a set of pixels—arranged in 2D (photo or picture), 3D (video) or 4D (light field)—and no additional input is used, the nature of these two problems is completely different. Appearance can be defined in terms of physical laws, so we can build models in advance that approximate such laws; while style is a subjective concept derived from social and environmental factors, for which we need to rely on human-based perception and machine learning techniques. In the following, we present the conclusions of each topic separately.

APPEARANCE IN REAL SCENES

Appearance can have multiple definitions, and be modeled in several ways depending on the complexity of the materials and the light we want to capture. However, the more information we want to capture about the scene, the more complex and expensive the equipment needs to be. In this thesis we have focused on visual data that can be obtained from conventional consumer cameras, such as images, video sequences or light fields. We have relied on a very simple model named "intrinsic decomposition", which represents a scene as the product of two components: reflectance and shading. This model assumes that the materials are Lambertian and that scene is illuminated by a white light. Despite these assumptions, we have shown during Part II of this thesis that this model is valid for a huge number of scenes present in the natural world.

In Chapter 2, we have presented our solution to the problem of intrinsic decomposition in single images. In our formulation we have relaxed common C_0 and C_1 constraints of Retinex-based methods by allowing C_1 discontinuities in the shading layer. Our approach which is based on a two level clustering strategy -first in color space and then in image space- allows to incorporate global and local constraints, respectively. Our method is free of user interaction and yields results on pair with user-assisted methods highly costly to tune. We also believe that our approach could inspire future work on color and reflectance clustering. Due to the difficulty of the task, this is still an open problem and recent trends rely on data-driven models where a dictionary of reflectances is learnt via human computation [13] or using synthetic data for training in conjunction with deep neural networks [133].

In Chapter 3, we extend the problem in the temporal dimension, taking video sequences as input. This domain has new challenges nonexistent in single images. First, the amount of data to handle increases drastically e.g. a typical 10 seconds video sequence of 30fps might contain 300 frames to process; second, we need to guarantee the temporal consistency and avoid flickering artifacts between frames; third, the content of the scene is not static, so new objects might appear not present at the beginning of the sequence. With all this constraints in mind, we have developed a solution which does not explodes computationally, can handle new unseen objects, and is temporal consistent. Our work, although not real time, was seminal in the field, and inspired recent works that have reached real time rates [127] producing on pair quality results to ours. The benefits of this decomposition in video sequences span those we found for single images, such as segmentation or material editing, while can improve other tasks which are sensible to illumination changes and occlusions, such as optical flow estimation.

In Chapter 4, we further extend the problem in the angular dimension, taking light fields as input. This type of data shares the benefit of single images in terms of static scenes, but it has an increased difficulty with respect to video sequences, as the angular coherency is much more difficult to maintain than the temporal one. We could try to "stack" the light field into multiple video combinations but that would be highly inefficient. Instead, in our approach we leverage the whole volume to obtain more precise and coherent cues about albedo variations. Our solution is a simple approach to the problem that still assume Lambertian materials. Thus, given that light field volumes can capture much rich information about material properties, there is still plenty of potential future work in this field, which we will discuss later on in this chapter.

STYLE IN ARTISTIC SCENES

In the second part of this thesis we have focused on the problem of recognizing style from illustration art. In Chapter 5, we presented a similarity metric trained from crowdsourced data on the human perception of style similarity. In our framework, we propose a set of engineered features which capture several properties of style, in particular, shading, strokes, color and texture features. Then, we leverage human computation to learn the relative importance of each of those features to perceive style in this kind of artwork. Beyond our application of clip art style, we believe that the proposed framework is suitable for any other domain which requires perceptual comparisons. In fact, our work was source of inspiration for subsequent works which used the same framework to learn perceptual similarity of shapes [119], faces [25], infographics [152], 3D furniture [112], or decoration for interiorism [12]. The success of this idea was in part due to the recent demand to search and explore huge amounts of data coming from multiple domains. As the amount of online information grows, the need to develop better tools to manage this data in alternative dimensions rather than using the hand-made labels becomes more and more necessary. Besides, we might find sources of data that might not be even labeled. Style is a domain which, contrarily to visual semantic search, is still highly unexplored, and we believe that our work has planted a fruitful seed.

On the other hand, just a style similarity metric might not be enough for a successful application, and we might want to combine this property with hand-made labels or visual semantics. Thus, in Chapter 6 we have proposed a style-based exploratory interface which combines semantic labeling of the images with the style similarity metric. We have been able to summarize the most frequent styles of a dataset so that the user can figure out in a first glance if the dataset contains all the images he or she needs in a certain style and with a particular content.

FUTURE WORK

Regarding the problem of appearance capture, an exciting avenue of future work is to use deep convolutional networks to learn more complex representations of appearance. In particular, light field volumes have a very structured representation which make them specially suitable to fit these type of non-linear models [85]. Recent works in the field support this idea, for example, Heber and Pock train a convolutional network to identify disparity values and extract depth maps [72], or Jampani et al [79], propose a framework to learn sparse high dimensional filters. On the other hand, viewpoint dependent effects like specular reflections [171] have a particular behaviour in the light field volume. Thus, it could be interesting to extend the intrinsic model to take into account specular effects, for example, using the Dichromatic Reflection Model [159]. Another potential idea to improve this low level task is to use high level semantic information which can be successfully captured with deep convolutional networks [95]. For example, if we knew that an area of surface is labeled as "rug", we would know that variations in chromaticity in that area of the scene would be due to reflectance an not shading variations. This idea was inspired by a recent work in the field that uses both types of input—semantic and structural—to predict occlusion boundaries in video sequences [50].

Regarding *style*, we would like to explore other domains in the field of digital design. Following our previous work [44], which selects an optimal subset of icons via human computation, we would like to extend this approach by providing a style similarity metric for icons. Then, by taking as input a dataset of icons like the one provided by the Noun Project[1], the whole pipeline would be fully automatic. Automatizing design tasks is a very interesting topic too, and in particular, we have been working on style transfer for vector art. However, this problem is highly challenging as the style of this type of data is highly defined. Contrarily to transferring style for pictures and photographs [58], vector art is much more structured and, many times, style in this kind of artworks involve changing the abstraction level, that is, the shape, making the task more difficult. In this regard, recent advances in the use of deep generative networks to transfer style between fonts [176] makes us think that a combination between structural information with deep learning strategies could be a interesting way to go.

Intending to combine what we have learned from both domains, real scenes and vector art, we would like to investigate the cross-depiction problem, and try to find an answer to the question: What does make a cat look like a cat? Since our human visual system is able to recognize a cat equally if it is sketch or a realistic picture, is it possible to obtain a model which is invariant to depiction? This is an open question subject of current research, which might be possible to answer if we will be able to understand and find part-based models in deep neural networks.

1 `https://thenounproject.com/`

PERSONAL CONCLUSIONS

During these years I have had the opportunity to work with many people in many different projects. I was lucky to be part of a highly collaborative group and to get to know and work with very respectful researchers around the world. All this has helped me to grow as a researcher, from the very first meeting that I can remember where everybody knew everything and me nothing, to the very last one, where it was me who lead the discussion group. I have learnt to adapt to the circumstances and the problems and to understand that sometimes is better to kill a project on time, rather than banging your head against a brick wall—this is something that took me quite some time to learn. From working remotely with other people outside the lab, I had to improve my communication skills and to figure out ways to present the results effectively. Besides, being part of a multidisciplinary group opened my mind to learn other fields apart from my initial interests, which I think it is something very valuable for a researcher.

More personally, it is difficult to express with words all the feelings that come up when I think about these five years, the ups and downs, the excitation and the frustration, the hits and the failures... But in the end, if I put all those in a balance the outcome is always positive.

BIBLIOGRAPHY

[1] Intrinsic Light Fields - Supplementary Material. `http://webdiis.unizar.es/~elenag/projects/intrinsicLF/supplementary/supplementary.html`. Accessed: 2016-08-08.

[2] Ao, H., Zhang, Y., Jarabo, A., Masia, B., Liu, Y., Gutierrez, D., and Dai, Q. Light Field Editing Based on Reparameterization. In *Proc. Pacific-Rim Conference on Multimedia* (2015), Springer.

[3] Aucouturier, J.-J., and Pachet, F. Music Similarity Measures: What's the Use? In *Proc. International Society for Music Information Retrieval* (2002).

[4] Averkiou, M., Kim, V., Zheng, Y., and Mitra, N. J. ShapeSynth: Parameterizing Model Collections for Coupled Shape Exploration and Synthesis. *Computer Graphics Forum (Proc. Eurographics) 33*, 2 (2014).

[5] Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. Neural Codes for Image Retrieval. In *Proc. European Conference on Computer Vision* (2014), Springer.

[6] Bae, S., Paris, S., and Durand, F. Two-scale Tone Management for Photographic Look. *ACM Trans. Graphics (Proc. SIGGRAPH) 25*, 3 (2006).

[7] Bai, X., Wang, J., Simons, D., and Sapiro, G. Video SnapCut: Robust Video Object Cutout using Localized Classifiers. *ACM Trans. Graphics (Proc. SIGGRAPH) 28*, 3 (2009).

[8] Barron, J., and Malik, J. Shape, Illumination, and Reflectance from Shading. *IEEE Trans. Pattern Analysis and Machine Intelligence 37* (2015).

[9] Barron, J. T., and Malik, J. Intrinsic Scene Properties from a Single RGB-D Image. In *Proc. Computer Vision and Pattern Recognition* (2013), IEEE.

[10] Barrow, H. G., and Tenenbaum, J. M. Recovering Intrinsic Scene Characteristics from Images. In *Proc. Computer Vision Systems* (1972).

[11] Bell, R. M., and Koren, Y. Lessons from the Netflix Prize Challenge. *ACM SIGKDD Exploration 9*, 2 (2007).

[12] Bell, S., and Bala, K. Learning Visual Similarity for Product Design with Convolutional Neural Networks. *ACM Trans. Graphics (Proc. SIGGRAPH) 34*, 4 (2015).

[13] Bell, S., Bala, K., and Snavely, N. Intrinsic Images in the Wild. *ACM Trans. Graphics (Proc. SIGGRAPH) 33*, 4 (2014).

[14] Berger, I., Shamir, A., Mahler, M., Carter, E., and Hodgins, J. Style and Abstraction in Portrait Sketching. *ACM Trans. Graphics (Proc. SIGGRAPH) 32*, 4 (2013).

[15] Bernardini, F., and Rushmeier, H. The 3D Model Acquisition Pipeline. *Computer Graphics Forum 21*, 2 (2002).

[16] Bhat, P., Zitnick, C. L., Cohen, M., and Curless, B. GradientShop: A Gradient-domain Optimization Framework for Image and Video Filtering. *ACM Trans. Graphics (Proc. SIGGRAPH) 29*, 2 (2010).

[17] Bi, S., Han, X., and Yu, Y. An L 1 Image Transform for Edge-Preserving Smoothing and Scene-Level Intrinsic Decomposition. *ACM Trans. Graphics (Proc. SIGGRAPH) 34*, 4 (2015).

[18] Birklbauer, C., Schedl, D. C., and Bimber, O. Nonuniform Spatial Deformation of Light Fields by Locally Linear Transformations. *ACM Trans. Graphics 35*, 5 (2016).

[19] Blumenson, J. J. *Identifying American Architecture: A Pictorial Guide to Styles and Terms, 1600-1945*. WW Norton & Company, 1981.

[20] Bonneel, N., Sunkavalli, K., Paris, S., and Pfister, H. Example-Based Video Color Grading. *ACM Trans. Graphics (Proc. SIGGRAPH) 32*, 4 (2013).

[21] Bonneel, N., Sunkavalli, K., Tompkin, J., Sun, D., Paris, S., and Pfister, H. Interactive Intrinsic Video Editing. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 33*, 6 (2014).

[22] Bousseau, A., Paris, S., and Durand, F. User-assisted Intrinsic Images. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009).

[23] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. High Accuracy Optical Flow Estimation based on a Theory for Warping. In *Proc. European Conference on Computer Vision* (2004), Springer.

[24] Campbell, N. D. F., and Kautz, J. Learning a Manifold of Fonts. *ACM Trans. Graphics (Proc. SIGGRAPH) 33*, 4 (2014).

[25] Cao, C., and Ai, H.-Z. Facial Similarity Learning with Humans in the Loop. *Springer Journal of Computer Science and Technology 30*, 3 (2015).

[26] Carneiro, G., Chan, A. B., Moreno, P. J., and Vasconcelos, N. Supervised Learning of Semantic Classes for Image Annotation and Retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence 29*, 3 (2007).

[27] Chen, B., Ofek, E., Shum, H.-Y., and Levoy, M. Interactive Deformation of Light Fields. In *Proc. Symposium on Interactive 3D Graphics and Games* (2005), ACM.

[28] Chen, Q., and Koltun, V. A Simple Model for Intrinsic Image Decomposition with Depth Cues. In *Proc. International Conference on Computer Vision* (2013), IEEE.

[29] Chen, T., Cheng, M.-M., Tan, P., Shamir, A., and Hu, S.-M. Sketch2Photo: Internet Image Montage. *ACM Trans. Graphics (Proc. SIGGRAPH) 28*, 5 (2009).

[30] Chen, X., Bennett, P. N., Collins-Thompson, K., and Horvitz, E. Pairwise Ranking Aggregation in a Crowdsourced Setting. In *Proc. ACM WSDM* (2013).

[31] Cho, D., Kim, S., and Tai, Y.-W. Consistent Matting for Light Field Images. In *Proc. European Conference on Computer Vision* (2014), Springer.

[32] Dana, K. J., van Ginneken, B., Nayar, S. K., and Koenderink, J. J. Reflectance and Texture of Real-world Surfaces. *ACM Trans. Graphics 18*, 1 (1999).

[33] Datta, R., Joshi, D., Li, J., and Wang, J. Z. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Computing Surveys 40*, 2 (2008).

[34] Debevec, P. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography. In *Proc. SIGGRAPH* (1998), ACM.

[35] Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W., and Sagar, M. Acquiring the Reflectance Field of a Human Face. In *Proc. SIGGRAPH* (2000), ACM.

[36] Doersch, C., Singh, S., Gupta, A., Sivic, J., and Efros, A. What Makes Paris Look like Paris? *ACM Trans. Graphics 31*, 4 (2012).

[37] Dong, Y., Tong, X., Pellacini, F., and Guo, B. AppGen: Interactive Material Modeling from a Single Image. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 30*, 6 (2011).

[38] Duchêne, S., Riant, C., Chaurasia, G., Lopez-Moreno, J., Laffont, P.-y., Popov, S., Bousseau, A., and Drettakis, G. Multi-View Intrinsic Images of Outdoors Scenes with an Application to Relighting. *ACM Trans. Graphics 34*, 5 (2015).

[39] Durand, F. An Invitation to Discuss Computer Depiction. In *Proc. NPAR* (2002).

[40] Eisemann, E., Winnemöller, H., Hart, J. C., and Salesin, D. Stylized Vector Art from 3D Models with Region Support. *Computer Graphics Forum 27*, 4 (2008).

[41] Eitz, M., Hay, J., and Alexa, M. How Do Humans Sketch Objects? *ACM Trans. Graphics (Proc. SIGGRAPH) 31*, 4 (2012).

[42] Epshtein, B., Ofek, E., and Wexler, Y. Detecting Text in Natural Scenes with Stroke Width Transform. In *Proc. Computer Vision and Pattern Recognition* (2010), IEEE.

[43] Farbman, Z., and Lischinski, D. Tonal Stabilization of Video. *ACM Trans. Graphics (Proc. SIGGRAPH) 30*, 4 (2011).

[44] Farnung-Laursen, L., Koyama, Y., Chen, H.-T., Garces, E., Gutierrez, D., Harper, R., and Igarashi, T. Icon Set Selection via Human Computation. In *Pacific Graphics Short Papers* (2016).

[45] Felzenszwalb, P. F., and Huttenlocher, D. P. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision 59*, 2 (2004).

[46] Filip, J., Chantler, M., and Haindl, M. On Uniform Resampling and Gaze Analysis of Bidirectional Texture Functions. *ACM Trans. Applied Perception 6*, 3 (2009).

[47] Freund, Y., and Schapire, R. E. A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. In *Proc. European Conference on Computational Learning Theory* (1995), Springer.

[48] Fried, O., DiVerdi, S., Halber, M., Sizikova, E., and Finkelstein, A. IsoMatch: Creating informative grid layouts. *Computer Graphics Forum (Proc. Eurographics) 34*, 2 (2015).

[49] Frisby, J. P., and Stone., J. V. *Seeing: The Computational Approach to Biological Vision*. MIT Press, 2010.

[50] Fu, H., Wang, C., Tao, D., and Black, M. J. Occlusion Boundary Detection via Deep Exploration of Context. In *Proc. Conference on Computer Vision and Pattern Recognition* (2016), IEEE.

[51] Funt, B. V., Drew, M. S., and Brockington, M. Recovering Shading from Color Images. In *Proc. European Conference on Computer Vision* (1992), Springer.

[52] Furukawa, Y., and Ponce, J. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Trans. Pattern Analysis and Machine Intelligence 32*, 8 (2010).

[53] Garces, E., Agarwala, A., Gutierrez, D., and Hertzmann, A. A Similarity Measure for Illustration Style. *ACM Trans. Graphics (Proc. SIGGRAPH) 33*, 4 (2014).

[54] Garces, E., Agarwala, A., Hertzmann, A., and Gutierrez, D. Style-based Exploration of Illustration Datasets. *Multimedia Tools and Applications* (2016).

[55] Garces, E., Echevarria, J. I., Zhang, W., Wu, H., Zhou, K., and Gutierrez, D. Intrinsic Light Fields. *ArXiv e-prints* (Aug. 2016).

[56] Garces, E., Gutierrez, D., and Lopez-Moreno, J. Graph-Based Reflectance Segmentation. In *Proc. SIACG* (2011).

[57] Garces, E., Munoz, A., Lopez-Moreno, J., and Gutierrez, D. Intrinsic Images by Clustering. *Computer Graphics Forum (Proc. EGSR) 31*, 4 (2012).

[58] Gatys, L. A., Ecker, A. S., and Bethge, M. A Neural Algorithm of Artistic Style. In *Proc. Conference on Computer Vision and Pattern Recognition* (2015), IEEE.

[59] Gehler, P. V., Rother, C., Kiefel, M., Zhang, L., and Schölkopf, B. Recovering Intrinsic Images with a Global Sparsity Prior on Reflectance. In *Proc. Neural Information Processing Systems* (2011).

[60] Gooch, B., and Gooch, A. *Non-Photorealistic Rendering*. AK Peters Ltd, 2001.

[61] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. The Lumigraph. In *Proc. SIGGRAPH* (1996), ACM.

[62] Graham D. Finlayson, M. S. D., and Lu, C. Intrinsic Images by Entropy Minimization. In *Proc. European Conference on Computer Vision* (2004), Springer.

[63] Grosse, R., Johnson, M. K., Adelson, E. H., and Freeman, W. T. Ground-truth Dataset and Baseline Evaluations for Intrinsic Image Algorithms. In *Proc. International Conference on Computer Vision* (2009), Springer.

[64] Grundmann, M., Kwatra, V., Han, M., and Essa, I. Efficient Hierarchical Graph-Based Video Segmentation. In *Proc. Computer Vision and Pattern Recognition* (2010), IEEE.

[65] Guo, X., Yu, Z., Kang, S. B., Lin, H., and Yu, J. Enhancing Light Fields through Ray-Space Stitching. *IEEE Trans. Visualization and Computer Graphics*, 99 (2015).

[66] Han, Z., Liu, Z., Han, J., and Bu, S. 3D Shape Creation by Style Transfer. *The Visual Computer 31*, 9 (2014).

[67] Haralick, R. M., Shanmugam, K., and Dinstein, I. Textural Features for Image Classification. *IEEE Trans. Systems, Man and Cybernetics 3*, 6 (1973).

[68] Hasler, D., and Susstrunk, S. Measuring Colourfulness in Natural Images. In *Proc. SPIE: Human Vision and Electronic Imaging* (2003), vol. 5007.

[69] Hauagge, D., Wehrwein, S., Baval, K., and Snavely, N. Photometric Ambient Occlusion. In *Proc. Computer Vision and Pattern Recognition* (2013), IEEE.

[70] Hauagge, D., Wehrwein, S., Upchurch, P., Bala, K., and Snavely, N. Reasoning about Photo Collections using Models of Outdoor Illumination. In *Proc. British Machine Vision Conference* (2014).

[71] He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proc. Computer Vision and Pattern Recognition* (2016), IEEE.

[72] Heber, S., and Pock, T. Convolutional Networks for Shape from Light Field. In *Proc. Computer Vision and Pattern Recognition* (2016), IEEE.

[73] Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. H. Image Analogies. In *Proc. SIGGRAPH* (2001).

[74] Horn, B. K. Determining Lightness from an Image. *Computer Graphics and Image Processing 3*, 4 (1974).

[75] Hougen, D. R., and Ahuja, N. Estimation of the Light Source Distribution and its use in Integrated Shape Recovery from Stereo and Shading. In *Proc. International Conference on Computer Vision* (1993), Springer.

[76] Huang, S.-S., Shamir, A., Shen, C.-H., Zhang, H., Sheffer, A., Hu, S.-M., and Cohen-Or, D. Qualitative Organization of Collections of Shapes via Quartet Analysis. *ACM Trans. Graphics (Proc. SIGGRAPH) 32*, 4 (2013).

[77] Hurtut, T., Gousseau, Y., Cheriet, F., and Schmitt, F. Artistic Line-Drawings Retrieval based on the Pictorial Content. *ACM Journal Computing and Cultural Heritage 4*, 1 (2011).

[78] Isaksen, A., McMillan, L., and Gortler, S. J. Dynamically Reparameterized Light Fields. In *Proc. SIGGRAPH* (2000), ACM.

[79] Jampani, V., Kiefel, M., and Gehler, P. V. Learning Sparse High Dimensional Filters: Image Filtering, Dense CRFs and Bilateral Neural Networks. In *Proc. Computer Vision and Pattern Recognition* (2016), IEEE.

[80] Jarabo, A., Masia, B., Bousseau, A., Pellacini, F., and Gutierrez, D. How Do People Edit Light Fields? *ACM Trans. Graphics (Proc. SIGGRAPH) 33*, 4 (2014).

[81] Jarabo, A., Masia, B., and Gutierrez, D. Efficient Propagation of Light Field Edits. In *Proc. SIACG* (2011).

[82] Jensen, H. W., Marschner, S. R., Levoy, M., and Hanrahan, P. A Practical Model for Subsurface Light Transport. In *Proc. SIGGRAPH* (2001), ACM.

[83] Jiang, X., Schofield, A. J., and Wyatt, J. L. Correlation-based Intrinsic Image Extraction from a Single Image. In *Proc. European Conference on Computer Vision* (2010), Springer.

[84] Jin, R., Wang, S., and Zhou, Y. Regularized Distance Metric Learning: Theory and Algorithm. In *Proc. Neural Information Processing Systems* (2009).

[85] Johannsen, O., Sulc, A., and Goldluecke, B. What Sparse Light Field Coding Reveals about Scene Structure. In *Proc. Conference on Computer Vision and Pattern Recognition* (2016), IEEE.

[86] Kalogerakis, E., Nowrouzezahrai, D., Breslav, S., and Hertzmann, A. Learning Hatching for Pen-and-Ink Illustration of Surfaces. *ACM Trans. Graphics 31*, 1 (2012).

[87] Kanungo, T., Mount, D. M., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. Y. A Local Search Approximation Algorithm for k-Means Clustering. *Computational Geometry: Theory and Applications 28* (2004).

[88] Karayev, S., Trentacoste, M., Han, H., Agarwala, A., Darrell, T., Hertzmann, A., and Winnemoeller, H. Recognizing Image Style. In *Proc. British Machine Vision Conference* (2014), BMVA Press.

[89] Khan, E. A., Reinhard, E., Fleming, R. W., and Bülthoff, H. H. Image-based Material Editing. *ACM Trans. Graphics (Proc. SIGGRAPH) 25*, 3 (2006).

[90] Kim, C., , H., Pritch, Y., Sorkine-Hornung, A., and Gross, M. H. Scene Reconstruction from High Spatio-angular Resolution Light Fields. *ACM Trans. Graphics (Proc. SIGGRAPH) 32*, 4 (2013).

[91] Kimmel, R., Elad, M., Shaked, D., Keshet, R., and Sobel, I. A Variational Framework for Retinex. *International Journal of Computer Vision 52* (2003).

[92] Kleiman, Y., Fish, N., Lanir, J., and Cohen-Or, D. Dynamic Maps for Exploring and Browsing Shapes. In *Proc. Eurographics/ACMSIGGRAPH Symposium on Geometry Processing* (2013).

[93] Kohonen, T. The Self-Organizing Map. *Proc. of the IEEE 78*, 9 (1990).

[94] Kong, N., Gehler, P. V., and Black, M. J. Intrinsic Video. In *Proc. European Conference on Computer Vision* (2014), Springer.

[95] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet Classification with Deep Convolutional Neural Networks. In *Proc. Neural Information Processing Systems* (2012).

[96] Kulis, B. Metric Learning: A Survey. *Foundations and Trends in Machine Learning 5*, 4 (2013).

[97] Laffont, P., Bousseau, A., and Paris, S. Coherent Intrinsic Images from Photo Collections. *ACM Trans. Graphics (Proc. SIGGRAPH) 31*, 6 (2012).

[98] Laffont, P.-Y., and Bazin, J.-C. Intrinsic decomposition of image sequences from local temporal variations. In *Proc. International Conference on Computer Vision (ICCV)* (2015), Springer.

[99] Laffont, P.-Y., Bousseau, A., and Drettakis, G. Rich Intrinsic Image Separation for Multi-View Outdoor Scenes. Research Report RR-7851, INRIA, 2011.

[100] Laffont, P.-Y., Bousseau, A., Paris, S., Durand, F., and Drettakis, G. Coherent Intrinsic Images from Photo Collections. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 31* (2012).

[101] Lalonde, J.-F., Hoiem, D., Efros, A. A., Rother, C., Winn, J., and Criminisi, A. Photo Clip Art. *ACM Trans. Graphics 26*, 3 (2007).

[102] Land, E. H., and McCann, J. J. Lightness and Retinex Theory. *Journal of the Optical Society of America 61*, 1 (1971).

[103] Lang, M., Wang, O., Aydin, T., Smolic, A., and Gross, M. Practical Temporal Consistency for Image-based Graphics Applications. *ACM Trans. Graphics (Proc. SIGGRAPH) 31*, 4 (2012).

[104] Lee, K. J., Zhao, Q., Tong, X., Gong, M., Izadi, S., Lee, S. U., Tan, P., and Lin, S. Estimation of Intrinsic Image Sequences from Image + Depth Video. In *Proc. European Conference on Computer Vision* (2012), Springer.

[105] Lessig, L. *Remix: Making Art and Commerce Thrive in the Hybrid Economy*. Penguin Press, 2008.

[106] Levin, A., Lischinski, D., and Weiss, Y. Colorization using Optimization. *ACM Trans. Graphics (Proc. SIGGRAPH) 23*, 3 (2004).

[107] Levoy, M., and Hanrahan, P. Light Field Rendering. In *Proc. SIGGRAPH* (1996), ACM.

[108] Li, C., Member, S., and Chen, T. Aesthetic Visual Quality Assessment of Paintings. *IEEE J. Sel. Topics in Signal Processing 3*, 2 (2009).

[109] Li, H., Zhang, H., Wang, Y., Cao, J., Shamir, A., and Cohen-Or, D. Curve Style Analysis in a Set of Shapes. *Computer Graphics Forum 32*, 6 (2013).

[110] Li, Y., Ju, T., and Hu, S.-M. Instant Propagation of Sparse Edits on Images and Videos. *Computer Graphics Forum 29*, 7 (2010).

[111] Li, Y., Sun, J., Tang, C.-K., and Shum, H.-Y. Lazy Snapping. *ACM Trans. Graphics (Proc. SIGGRAPH) 23*, 3 (2004).

[112] Liu, T., Hertzmann, A., Li, W., and Funkhouser, T. Style Compatibility for 3d Furniture Models. *ACM Trans. Graphics (Proc. SIGGRAPH) 34*, 4 (2015).

[113] Liu, X., Wan, L., Qu, Y., Wong, T.-T., Lin, S., Leung, C.-S., and Heng, P.-A. Intrinsic Colorization. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 27*, 5 (2008).

[114] Lombardi, S., and Nishino, K. Reflectance and Illumination Recovery in the Wild. *IEEE Trans. on Pattern Analysis and Machine Intelligence 38*, 1 (2016).

[115] Lopez, A., Garces, E., and Gutierrez, D. Depth from a Single Image through User Interaction. In *Proc. Congreso Español de Informatica Grafica* (2014), Eurographics.

[116] Lopez-Moreno, J., Garces, E., Hadap, S., Reinhard, E., and Gutierrez, D. Multiple Light Source Estimation in a Single Image. *Computer Graphics Forum 32*, 8 (2013).

[117] Lopez-Moreno, J., Hadap, S., Reinhard, E., and Gutierrez, D. Compositing Images through Light Source Detection. *Computers & Graphics 34*, 6 (2010).

[118] Lumsdaine, A., and Georgiev, T. The Focused Plenoptic Camera. In *Proc. International Conference on Computational Photography* (2009), IEEE.

[119] Lun, Z., Kalogerakis, E., and Sheffer, A. Elements of Style: Learning Perceptual Shape Style Similarity. *ACM Trans. Graphics (Proc. SIGGRAPH) 34*, 4 (2015).

[120] Luo, Y., Liu, T., Tao, D., and Xu, C. Decomposition-Based Transfer Distance Metric Learning for Image Classification. *IEEE Trans. on Image Processing 23*, 9 (2014).

[121] Lytro Inc. The Lytro camera. `http://www.lytro.com`, 2013.

[122] Masia, B., Jarabo, A., and Gutierrez, D. Favored Workflows in Light Field Editing. In *Proc. CGVCVIP* (2014).

[123] Matsushita, Y., Nishino, K., Ikeuchi, K., and Sakauchi, M. Illumination Normalization with Time-Dependent Intrinsic Images for Video Surveillance. *IEEE Trans. Pattern Analysis Machine Intelligence 26*, 10 (2004).

[124] McCloud, S. *Understanding comics: The invisible art*. Northampton, Mass, 1993.

[125] McFee, B., and Lanckriet, G. Metric Learning to Rank. In *Proc. International Conference on Machine Learning* (2010).

[126] McFee, B., and Lanckriet, G. Learning Multi-modal Similarity. *J. Machine Learning Research 12* (2011).

[127] Meka, A., Zollhöfer, M., Richardt, C., and Theobalt, C. Live Intrinsic Video. *ACM Trans. Graphics (Proc. SIGGRAPH) 35*, 4 (2016).

[128] Müller, G., Meseth, J., Sattler, M., Sarlette, R., and Klein, R. Acquisition, Synthesis and Rendering of Bidirectional Texture Functions. In *Proc. Eurographics State of the Art Reports* (2004).

[129] Murray, N., Barcelona, D., Marchesotti, L., and Perronnin, F. AVA: A Large-Scale Database for Aesthetic Visual Analysis. In *Proc. Computer Vision and Pattern Recognition* (2012), IEEE.

[130] Nachtigal, N., and Semeraro, B. QMR Methods in Computational Fluid Dynamics.

[131] Nam, G., Lee, J., Wu, H., Gutierrez, D., and Kim, M. Simultaneous Acquisition of Microscale Reflectance and Normals. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 35*, 6 (2016).

[132] Nan, L., Sharf, A., Xie, K., Wong, T.-T., Deussen, O., Cohen-Or, D., and Chen, B. Conjoining Gestalt Rules for Abstraction of Architectural Drawings. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 30*, 6 (2011).

[133] Narihira, T., Maire, M., and Yu, S. X. Direct Intrinsics: Learning Albedo-Shading Decomposition by Convolutional Regression. In *Proc. International Conference on Computer Vision* (2015), Springer.

[134] Ng, R. Fourier Slice Photography. *ACM Trans. Graphics 24*, 3 (2005).

[135] Nguyen, C. H., Ritschel, T., and Seidel, H.-P. Data-Driven Color Manifolds. *ACM Trans. Graphics 34*, 2 (2015).

[136] Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W., and Limperis, T. Radiometry. USA, 1992, ch. Geometrical Considerations and Nomenclature for Reflectance.

[137] O'Donovan, P., Agarwala, A., and Hertzmann, A. Collaborative Filtering of Color Aesthetics. In *Proc. Computational Aesthetics* (2014).

[138] O'Donovan, P., Lībeks, J., Agarwala, A., and Hertzmann, A. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Trans. Graphics (Proc. SIGGRAPH) 33*, 4 (2014).

[139] Oh, B. M., Chen, M., Dorsey, J., and Durand, F. Image-based Modeling and Photo Editing. In *Proc. SIGGRAPH* (2001), ACM.

[140] Ojala, T., Pietikäinen, M., and Mäenpää, T. Multiresolution Grayscale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence 24*, 7 (2002).

[141] Omer, I., and Werman, M. Color Lines: Image Specific Color Representation. In *Proc. Computer Vision and Pattern Recognition* (2004), IEEE.

[142] Oskam, T., Hornung, A., Sumner, R., and Gross, M. Fast and Stable Color Balancing for Images and Augmented Reality. In *Proc. 3DIM-PVT* (2012), IEEE.

[143] Parikh, D., and Grauman, K. Relative Attributes. In *Proc. International Conference on Computer Vision* (2011), IEEE.

[144] Paris, S. Edge-Preserving Smoothing and Mean-Shift Segmentation of Video Streams. In *Proc. European Conference on Computer Vision* (2008), Springer.

[145] Patow, G., and Pueyo, X. A Survey of Inverse Rendering Problems. *Computer Graphics Forum 22*, 4 (2003).

[146] Pérez, P., Gangnet, M., and Blake, A. Poisson Image Editing. *ACM Trans. Graphics (Proc. SIGGRAPH) 22*, 3 (2003).

[147] Radlinski, F., Bennett, P. N., Carterette, B., and Joachims, T. Redundancy, Diversity and Interdependent Document Relevance. *SIGIR Forum 43*, 2 (2009).

[148] Ramamoorthi, R., and Hanrahan, P. An Efficient Representation for Irradiance Environment Maps. In *Proc. SIGGRAPH* (2001), ACM.

[149] Raytrix GmbH. 3D Light Field Camera Technology. `http://www.raytrix.de`, 2013.

[150] Reinert, B., Ritschel, T., and Seidel, H.-P. Interactive By-example Design of Artistic Packing Layouts. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2013).

[151] Rubinstein, M., Gutierrez, D., Sorkine, O., and Shamir, A. A Comparative Study of Image Retargeting. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 29*, 5 (2010).

[152] Saleh, B., Dontcheva, M., Hertzmann, A., and Liu, Z. Learning Style Similarity for Searching Infographics. In *Proc. Graphics Interface Conference* (2015).

[153] Sangkloy, P., Burnell, N., Ham, C., and Hays, J. The Sketchy Database: learning to retrieve badly drawn bunnies. *ACM Trans. Graphics 35*, 4 (2016).

[154] Schultz, M., and Joachims, T. Learning a Distance Metric from Relative Comparisons. In *Proc. Neural Information Processing Systems* (2003).

[155] Seitz, S. M., and Kutulakos, K. N. Plenoptic Image Editing. *International Journal of Computer Vision 48*, 2 (2002).

[156] Serrano, A., Garces, E., Gutierrez, D., and Masia, B. Convolutional Sparse Coding for capturing High Speed Video Content. *Computer Graphics Forum* (2016).

[157] Serrano, A., Gutierrez, D., and Masia, B. Compressive high-speed video acquisition. In *Proc. Congreso Español de Informatica Grafica* (2015).

[158] Serrano, A., Gutierrez, D., Myszkowski, K., Seidel, H., and Masia, B. An Intuitive Control Space for Material Appearance. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 35*, 6 (2016).

[159] Shafer, S. A. Color. Jones and Bartlett Publishers, Inc., 1992, ch. Using Color to Separate Reflection Components.

[160] Shamir, L., Macura, T., Orlov, N., Eckley, D. M., and Goldberg, I. G. Impressionism, Expressionism, Surrealism: Automated Recognition of Painters and Schools of Art. *ACM Trans. Applied Perception 7*, 2 (2010).

[161] Shapira, L., Shamir, A., and Cohen-Or, D. Image Appearance Exploration by Model-Based Navigation. *Computer Graphics Forum (Proc. Eurographics) 28*, 2 (2009).

[162] Shen, J., Yang, X., Jia, Y., and Li, X. Intrinsic Images Using Optimization. In *Proc. Computer Vision and Pattern Recognition* (2011), IEEE.

[163] Shen, L., Tan, P., and Lin, S. Intrinsic Image Decomposition with Non-local Texture Cues. In *Proc. Computer Vision and Pattern Recognition* (2008), IEEE.

[164] Shen, L., and Yeo, C. Intrinsic Images Decomposition using a Local and Global Sparse Representation of Reflectance. In *Proc. Computer Vision and Patter Recognition* (2011), IEEE.

[165] Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., and Cohen-Or, D. Unsupervised Co-Segmentation of a Set of Shapes via Descriptor-Space Spectral Clustering. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 30*, 6 (2011).

[166] Snavely, N., Seitz, S. M., and Szeliski, R. Photo Tourism: Exploring Photo Collections in 3D. *ACM Trans. Graphics 25*, 3 (2006).

[167] Strothotte, T., and Schlechtweg, S. *Non-photorealistic computer graphics: modeling, rendering, and animation*. Morgan Kaufmann, 2002.

[168] Sunkavalli, K., Matusik, W., Pfister, H., and Rusinkiewicz, S. Factored Time-lapse Video. *ACM Trans. Graphics (Proc. SIGGRAPH) 26*, 3 (2007).

[169] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going Deeper with Convolutions. In *Proc. Computer Vision and Pattern Recognition* (2015), IEEE.

[170] Tamuz, O., Liu, C., Belongie, S., Shamir, O., and Kalai, A. Adaptively Learning the Crowd Kernel. In *Proc. International Conference on Machine Learning* (2011).

[171] Tao, M., Su, J.-C., Wang, T.-c., Malik, J., and Ramamoorthi, R. Depth Estimation and Specular Removal for Glossy Surfaces Using Point and Line Consistency with Light-Field Cameras. *IEEE Trans. Pattern Analysis and Machine Intelligence*, August (2015).

[172] Tao, M. W., Hadap, S., Malik, J., and Ramamoorthi, R. Depth from Combining Defocus and Correspondence Using Light-Field Cameras. In *Proc. International Conference on Computer Vision* (2013), IEEE.

[173] Tappen, M., Freeman, W., and Adelson, E. Recovering Intrinsic Images from a Single Image. *IEEE Trans. Pattern Analysis Machine Intelligence 27*, 9 (2005).

[174] Tappen, M. F., Adelson, E. H., and Freeman, W. T. Estimating Intrinsic Component Images using Non-Linear Regression. In *Proc. Computer Vision and Pattern Recognition* (2006), IEEE.

[175] Tenenbaum, J. B., and Freeman, W. T. Separating Style and Content with Bilinear Models. *Neural Computation 12*, 6 (2000).

[176] Upchurch, P., Snavely, N., and Bala, K. From A to Z: Supervised Transfer of Style and Content Using Deep Neural Network Generators. *arXiv preprint arXiv:1603.02003* (2016).

[177] van der Maaten, L., and Hinton, G. E. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research 9* (2008).

[178] Venkataraman, K., Lelescu, D., Duparré, J., McMahon, A., Molina, G., Chatterjee, P., Mullis, R., and Nayar, S. PiCam: An Ultra-thin High Performance Monolithic Camera Array. *ACM Trans. Graphics 32*, 6 (2013).

[179] Wang, T.-c., Efros, A. A., and Ramamoorthi, R. Occlusion-aware Depth Estimation Using Light-field Cameras. In *Proc. International Conference on Computer Vision* (2015), Springer.

[180] Wanner, S., and Goldluecke, B. Variational Light Field Analysis for Disparity Estimation and Super-Resolution. *IEEE Trans. Pattern Analysis Machine Intelligence 36*, 3 (2014).

[181] Ward, J. H. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association 58*, 301 (1963).

[182] Weiss, Y. Deriving Intrinsic Images from Image Sequences. In *Proc. International Conference on Computer Vision* (2001), IEEE.

[183] Welinder, P., Branson, S., Belongie, S., and Perona, P. The Multidimensional Wisdom of Crowds. In *Proc. Neural Information Processing Systems* (2010).

[184] Willats, J., and Durand, F. Defining Pictorial Style: Lessons from Linguistics and Computer Graphics. *Axiomathes 15*, 3 (2005).

[185] Xu, K., Li, Y., Ju, T., Hu, S.-M., and Liu, T.-Q. Efficient Affinity-based Edit Propagation using Kd Tree. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009).

[186] Yamaguchi, K., Kiapour, M. H., Ortiz, L. E., and Berg, T. L. Retrieving Similar Styles to Parse Clothing. *IEEE Trans. on Pattern Analysis and Machine Intelligence 37*, 5 (2015).

[187] Yan, X., Shen, J., He, Y., and Mao, X. Re-texturing by Intrinsic Video. In *Proc. DICTA* (2010), IEEE.

[188] Yang, L. Distance Metric Learning: A Comprehensive Survey. Tech. rep., 2006.

[189] Yang, S., Wang, J., Fan, W., Zhang, X., Wonka, P., and Ye, J. An Efficient ADMM Algorithm for Multidimensional Anisotropic Total Variation Regularization Problems. In *Proc. SIGKDD International Conference on Knowledge Discovery and Data Mining* (2013), ACM.

[190] Yatziv, L., and Sapiro, G. Fast Image and Video Colorization using Chrominance Blending. *IEEE Trans. Image Processing, 15*, 5 (2006).

[191] Ye, G., Garces, E., Liu, Y., Dai, Q., and Gutierrez, D. Intrinsic Video and Applications. *ACM Trans. Graphics (Proc. SIGGRAPH) 33*, 4 (2014).

[192] Yumer, M. E., and Kara, L. B. Co-abstraction of Shape Collections. *ACM Trans. Graphics 31*, 6 (2012).

[193] Zell, E., Aliaga, C., Jarabo, A., Zibrek, K., Gutierrez, D., McDonnell, R., and Botsch, M. To Stylize or not to Stylize? Effect of Shape and Material Stylization on the Perception of Computer Generated Faces. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 34*, 6 (2015).

[194] Zhang, F. L., Wang, J., Shechtman, E., Zhou, Z. Y., Shi, J. X., and Hu, S. M. PlenoPatch: Patch-based Plenoptic Image Manipulation. *IEEE Trans. on Visualization and Computer Graphics*, 99 (2016).

[195] Zhang, Y., and Yang, Y. H. Multiple Illuminant Direction Detection with Application to Image Synthesis. *IEEE Trans. Pattern Analysis and Machine Intelligence 23*, 8 (2001).

[196] Zhang, Z., Wang, L., Guo, B., and Shum, H.-Y. Feature-based Light Field Morphing. *ACM Trans. Graphics 21*, 3 (2002).

[197] Zhao, Q., Tan, P., Dai, Q., Shen, L., Wu, E., and Lin, S. A Closed-Form Solution to Retinex with Nonlocal Texture Constraints. *IEEE Trans. Pattern Analysis and Machine Intelligence 34*, 7 (2012).

[198] Zhou, T., Krähenbühl, P., and Efros, A. A. Learning Data-driven Reflectance Priors for Intrinsic Image Decomposition. In *Proc. International Conference on Computer Vision* (2015), IEEE.

[199] Zhu, C., Bichot, C.-E., and Chen, L. Multi-scale Color Local Binary Patterns for Visual Object Classes Recognition. In *Proc. International Conference on Pattern Recognition* (2010), IEEE.

[200] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Mathematical Software 23*, 4 (1997).