PERSONAL PHOTO ENHANCEMENT

HUIWEN CHANG

A DISSERTATION

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE

BY THE DEPARTMENT OF

COMPUTER SCIENCE

ADVISER: PROFESSOR ADAM FINKELSTEIN

JUNE 2018

Abstract

Thanks to the democratization of digital cameras and camera-equipped smartphones, casual photographers have an increasing need to be able to easily enhance their growing personal photo collections. Existing commercial tools for photo enhancement are either expressive but too sophisticated for casual photographers (e.g. Adobe Photoshop), or easy-to-use but limited in expressiveness (e.g. one-click filters from Instagram). This poses the challenge of how to enable novices to easily edit and enhance their personal photo collections. This thesis describes three systems that collectively build a foundation for solving the practical problem of enhancing personal photo collections as a whole.

Our goal is to enhance a personal photo collection, and the first challenge is to select which photos in the collection are worth keeping, editing and sharing. In particular we would like to automatically exclude bad photos from a series of shots taken of the same scene. Note that this goal takes a different perspective from that of previous work, for two reasons. First, casual personal photos have different statistics from those of general images. Second, the problem of selecting which are the best photos in a series of similar shots is different from establishing an overall quality measure for any single photo in isolation.

Next, the thesis studies the color transfer problem on photo enhancement. Because visual taste is personal, an enhancement tool is necessary to provide users with both enough freedom and intuitive control. Therefore, additional to a novel color transfer algorithm, we also design a simple interface for making color exploration easy and intuitive. When taking a photo collection as a whole, the color transfer algorithm can also be applied to enhance the overall color consistency among multiple photos automatically, which also enables users to tune their entire photo album simultaneously.

Finally, we narrow down photo enhancement to specific domains and focus on portraiture in personal photo collections. We propose a novel framework for modifying portrait images to match with the style of another person in a reference photo, and in particular, we explore the application of this technique towards transferring makeup styles.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor Adam Finkelstein for his patience, encouragement, optimistic attitude, and for his unwavering support of my research work and study. Whenever I came across a bottleneck in research, he had always been able to motivate and guide me. I could not have imagined having a better advisor and mentor who makes my Ph.D. study such a pleasant journey.

My sincere thanks goes to Jue Wang, Michael Cohen and Jingwan(Cynthia) Lu, who provided me an opportunity to join their team as intern. I am also grateful to Kaiming He for enlightening me the first glance of research. Besides, I appreciate to collaborate with Fisher Yu, and Yiming Liu and Steve Diverdi. Doing projects together is both fun and a great learning experience for me.

I appreciate my thesis committee members for providing invaluable feedback on this thesis, despite the numerous revisions we made in the last minute. I thank all Tiggraph reviewers and members of the Princeton Graphics Group who have read preliminary version of this work and provided great suggestions. My PhD life would not have been so amazing without them.

Last but not the least, I would like to thank my mom and my friends for supporting me spiritually throughout writing this thesis and my life in general.

My graduate studies and this work were supported by Princeton Fellowship, Microsoft PhD Fellowship and Adobe.

To my mom.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ease and ubiquity of digital cameras has led to an ever-increasing size of personal photo collections. In 2017, Facebook reported that people were uploading photos at an average rate of more than 300 million images per day. However, an overwhelming majority of these photo collections are casual – they effectively chronicle a moment, but may not be well-framed, posed, lit, and generally lack the eye of an expert photographer.

The challenge we face, as computational photography researchers, has shifted from photo acquisition to finding ways to allow novices easily editing and enhancing their personal photo collections. Yet existing commercial software are ill-equipped to tackle this challenge. Image editing software like Adobe Photoshop empowers professional photographers to achieve this impact by manipulating pictures with tremendous control and flexibility. However, using such software requires tremendous care and effort, which is neither possible nor warranted from casual photographers. In contrast, photo-sharing platforms like Instagram allow anyone to quickly and easily apply style filters to their images with one simple click. But these operations are restricted to a set of carefully designed filters, which means that many desired results may be out-of-scope and unachievable.

There are extensive efforts of research work on photo enhancement as it is a long-standing goal in computational photography. Early work such as denoising [11, 80], deblurring [103, 5], and tone correction [97], guide the correction of common flaws in order to enhance the *quality* of images, and have achieved significant results. Recent work [49, 39, 126] shed light on the enhancement regarding the image *aesthetics*. They typically rely on optimizing a function which assesses image aesthetics either in general or in specific domains, such as composition and lighting. Despite recent progress towards computational image aesthetics classification, assessing and enhancing photo aesthetics is still challenging. The reasons are manifold. One of the major obstacles is that numerous factors can affect people's assessment or feeling about a photo, and it is hard to parameterize those factors. Learning such preference requires a large scale dataset to encompass all the different types of images and scenarios a model will need to perform well in. Therefore, previous work has often turned to online photography communities and photography challenges to gather images for their datasets, which helped achieve notable progress towards *general image* aesthetics classification problem. However, those datasets rarely contain *personal photos* with casual characteristics as defined above. Chapter 2 presents a project on gathering large scale dataset distilled from personal photo collections, and using it to establish a relative ranking of photographs by learning the human preferences with machine learning techniques [17]. Another reason why photo aesthetics enhancement is challenging is that aesthetics assessment is inherently subjective. On one hand, fully automatic methods may hinder novices from fully expressing their feelings or meeting their goal; on the other hand, sophisticated tools are inscrutable for novices to master. As such, there exists this tradeoff between the freedom of exploration and the ease of use for enhancing photos. Chapters 3 and 4 discuss this tradeoff for two different problems, respectively – manipulating image colors [15] and personalizing makeup styles for portraits [16].

Different from previous work, the focus of this thesis is not limited to enhancing individual images; it also builds a foundation for solving a more practical problem – the problem of enhancing personal photo collections as a whole. This thesis studies three particular aspects of the general problem.

First, people usually take many photos in order to capture a moment or scene well. However, sifting through and organizing a huge collection of photos – grouping similar photos and deciding which are the "keepers" and which ones to omit – is cumbersome and time consuming. To facilitate this *photo triage* process, previous methods operate by gauging the aesthetics of individual images on an absolute scale (from "bad" to "good"). However, much of what makes a photograph a high aesthetic score goes beyond the camera setting or composition into the subjective response to its particular content. We discover that previous work tend to yield similar scores for similar image content, and therefore perform poorly for the photo triage problem. In contrast, we propose to establish a relative ranking for "better" or "worse" photos among a series taken of the same scene. Thus we acquire and share a dataset specific to this application; we develop a model for human preferences in this data; and we establish a benchmark for measuring the success of such models. As these models improve in the future, they can also serve as guidance for both automatic and human-in-the-loop image enhancement algorithms

Second, the preferred photos selected from large photo collection usually will be gathered into a new album. Professional photographers expend great care and control on maintaining the same lighting, subject and camera setting for each photo collection, but these goals are largely unattainable for casual photographers. Due to the lack of those controls, photos curated by casual photographers may look inconsistent with each other. Though exemplar-based color transfer is widely studied [96, 118, 38, 121], color transfer among a collection of images has received little attention. In Chapter 3, we discuss an application of the color transfer algorithm on enhancing the overall color consistency

among a set of images automatically. It also has the side benefit that it enables users to tune the entire album simultaneously instead of manipulating each picture separately.

Third, as photos have become a key element in online social interactions, portraits take up a large portion of photo collections. As a result, digitally editing portraits has come to play a crucial role. Modern applications allow consumers to add cartoon elements, or turn photos into paintings. However, it is hard to achieve photo realistic effects when stylizing faces, because people are especially sensitive to facial features and artifacts. Furthermore, framing this as a supervised learning problem is arduous, because it needs collections of triplets of photos from which to learn: the source photo, the reference style photo, and the ground truth output (which preserves identity of the source and style of the reference). Thus, Chapter 4 introduces a novel framework for modifying a portrait photo to match with the style of another person in a reference photo.

Below we list the principal contributions of this thesis.

Chapter 2: First, we identify the problem of automatic triage for a photo series. We publish a large-scale dataset together with a public benchmark for evaluating various models of human preference. We introduce several machine learning approaches, including a variant of the Siamese network model, for predicting human preference among photo pairs from a series. We propose a new framework for automatic photo enhancement, as well as a new method for selecting frames from video, based on the new dataset and learner.

Chapter 3: Next, we describe an intuitive graphical interface for recoloring a photo by editing its color palette. We enumerate the desired properties for both user interface and color transformation algorithm. Our solutions are deeply motivated by these properties. We describe an efficient algorithm for automatically generating a suitable color palette for this task, as well as a robust recoloring algorithm that produces compelling results at interactive rates. Two crucial qualities are that this approach leaves colors in gamut and avoids reordering relative pixel luminance. We present a user study

showing that our method outperforms other recoloring strategies in expressiveness as well as ease of use. We introduce an application for enhancing the color consistency of multiple photos.

Chapter 4: We introduce a feed-forward makeup transformation network that can quickly transfer the style from an arbitrary reference makeup photo to an arbitrary source photo. We describe an asymmetric makeup transfer framework wherein we train a makeup removal network jointly with the transfer network in order to preserve the identity. We also train a style discriminator to encourage faithful reproduction of the reference makeup style. We present a new dataset of high quality before- and after-makeup images gathered from YouTube videos.

Chapter 2

Photo triage

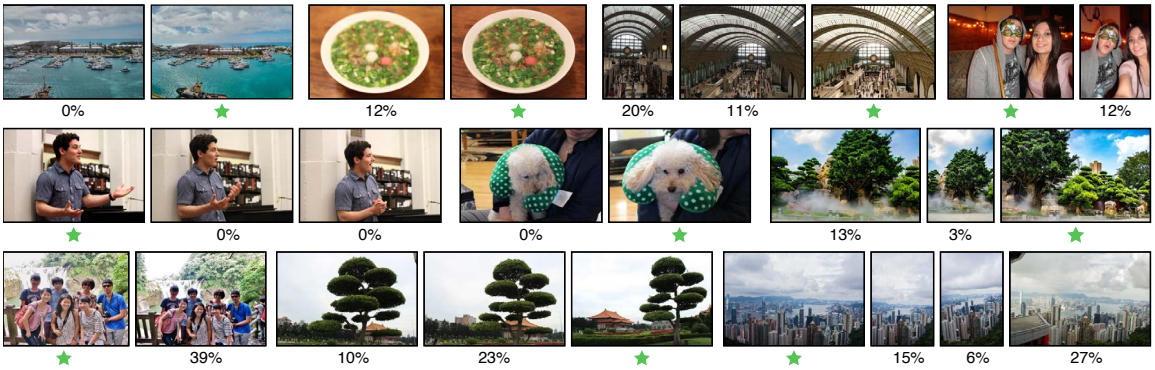


Figure 2.1: Multiple photo series, each sampled from one category in our dataset (Figure 3a: water, object, interior, selfie, ...). The starred photo in each series is the one preferred by the majority of people, while the percentage below each other photo indicates what fraction of people would prefer that photo over the starred one in the same series.

2.1 Introduction

The ease and ubiquity of digital cameras has led to an ever-increasing size of personal photo collections. To capture a vacation, party or other event, people often take a series of shots of a particular scene with varied camera parameters and content arrangement, in the hope of later being able to pick a few good photos to edit, post or share (Figure 2.1). However, sifting through and managing a huge collection of photos – grouping similar photos and deciding which are the "keepers" and which ones to omit – is cumbersome

and time consuming. There are currently several commercial tools that facilitate this photo *triage* process. Photo browsers such as iPhoto and Picasa allow consumers to hierarchically organize and navigate through photo groups based on time or geographic information, while several online photo-sharing social networks like Facebook and Google Photos extract faces and higher level content for labeling purposes. While these tools improve album organization, none of them provide a way to automatically filter out the bad images or find the best images among a series of similar shots. To know which ones are worth keeping or sharing, users still need to sift through the entire album by hand, which is tedious. Our goal is to facilitate this process of finding the best images, by providing an automatic estimator of their relative ranking.

There has been extensive effort on assessing photo quality or aesthetics that gauges each image independently on an absolute scale (from "bad" to "good"). In contrast our goal is to establish a relative ranking for "better" or "worse" photos from among a series of similar shots (Figure 2.2). Existing methods that produce absolute scores tend to yield similar scores for similar images, and therefore perform poorly for our problem. There are also methods for selecting the best among a photo "burst" (typically based only on low-

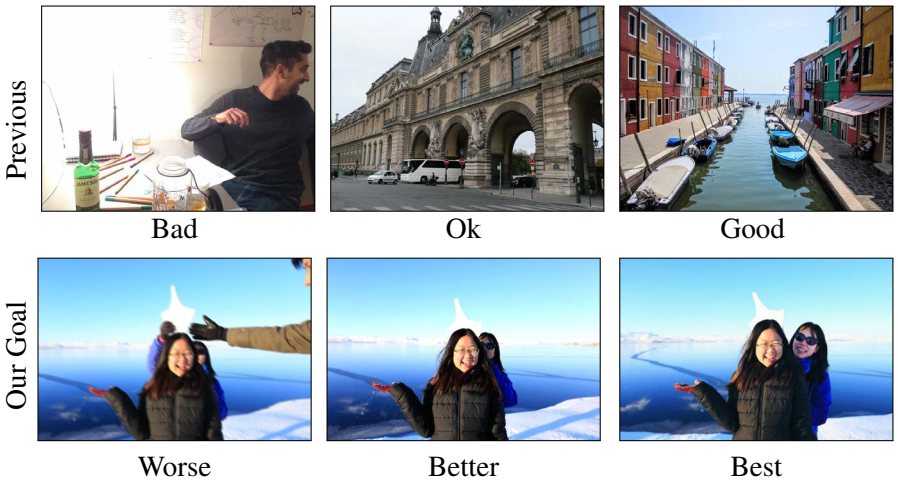


Figure 2.2: Above: Previous work evaluates photo quality on an absolute scale (bad to good). Below: Our goal is to find relative ranking among a series shots of the same scene (for which previous methods would typically provide similar absolute scores).

level features like blurriness or closed eyes) but in our case the composition usually varies substantially more and requires higher-level features for effective comparison.

To address this problem, we collected a new dataset that contains $15,545$ personal photos, organized in $5,953$ series. The dataset consists of unedited photos from personal photo collections. The contributors are photographers of varying ability, and the vast majority of the photos are "casual" – characteristic of typical consumer photo albums. We conducted a study on Amazon Mechanical Turk in which subjects were asked for preferences (and comments) among $98,780$ pairs of photos, each pair from within a series, from which we extract a ground truth ranking of human preference across each series (★ and % in Figure 2.1). With this data we also establish a benchmark – training, validation and test sets, as well as a set of criteria for evaluating new models for human preference against the data. We believe this is the first large public dataset of unedited personal photos designed to address the photo triage problem.

To model human preference, we experimented with several machine learning approaches based on hand-crafted features as well as features established in the object recognition literature. Among pairs of images where human preference is reasonably consistent ($\geq$70% agreement), our best-performing model (based on a variant of the Siamese network prototyping from the VGG model) is able to predict human preference 73% of the time. Comparing with several baseline methods from previous research, we find the best-performing one (from Khosla et al. [56]) achieves 57% accuracy.

In addition to modeling human preference for automatic triage, we also introduce two new applications for this kind of data. First, we show that new images can be automatically enhanced by analogy to other pairs of images in the dataset. Second we show that our learner can be used to select "good" frames from video to provide an overview of a shot.

2.2 Related Work

Photo Triage. With the ever-increasing size of personal photo collections, photo triage has drawn the attention of both researchers and developers in recent years. Drucker et al. [28] design a user interface which enhances viewing experience and eases the control for users during photo triage. Jacobs et al. [46] introduce the notion of "cosaliency" – local structural changes between image pairs – which facilitates the manual triage process by helping the viewer easily see the differences between images. Both of them aim at expediting the process of photo triage interactively, while we propose a fully automatic approach for triage among photo series. Zhu et al. [129] focus on selecting more attractive portraits from large photo collections based on facial features. In addition to considering facial expressions, our chapter addresses the general (more difficult) problem, including natural scenes, urban environments, interior scenes, cluttered scenes, and lone objects.

Image Summarization. Researchers have worked on summarizing image collections The primary goal of this line of research is to select a few representative images from a collection by jointly considering photo quality, event coverage and scene diversity. Sinha et al. [107] propose content- and context-based optimization functions for summarizing large personal photo collections. Simon et al. [105] propose a summarization approach that is more tailored towards 3D scenes. The triage problem we address can be viewed as a sub-problem of image summarization, as it focuses only on finding good images from photo series, but does not consider the representativeness of these images to the whole album, an essential consideration in summarization. Our approach thus can be used as a component to improve existing summarization systems.

Quality Assessment. An alternative way to deal with photo triage is selecting by assessing quality for each photo in series. Researchers have studied aesthetics or quality evaluation for photos, and such methods can be used to determine a ranking among photos. Early efforts employ handcrafted features to evaluate visual aesthetics and adhere to principles of photography. The features involve both low-level features such

as lighting [78, 13, 53, 125], texture [26, 111] and color [26, 87], as well as high-level features such as composition [78, 7, 68, 27, 36, 91, 127] and content [27, 77, 53]. Such approaches achieve good results by using joint features to predict image aesthetics, but they are typically limited to heuristics inspired by the photographic guidelines (e.g., the rule of thirds, golden ratio, and simplicity). Because generic visual features are effective to represent semantic image descriptors, methods like that of Marchesotti et al. [81] and Murray et al. [86] attempt to rate visual properties using features such as SIFT [73], GIST [89], and the Fisher Vector [81]. Ye et al. [120] borrows the idea of code book representations and automatically predicts human perceived image quality without a reference image. Recently, Lu et al. [74] applied deep neural network approaches and outperformed handcrafted and generic visual features on an aesthetic dataset. For the benchmark introduced in this chapter, the best existing method we have found is the "memorability" score of Khosla et al. [56], which is out-performed for this task by new approaches we introduce. The general strategy of establishing an absolute assessment across the range of images found in a typical photo album is more difficult than finding a relative ranking among a series of similar photos (Figure 2.2), and existing methods for finding absolute scores tend to provide similar scores across a series.

Datasets for Aesthetics Assessment. Prior datasets for aesthetic analysis of photos have been obtained from online photo sharing communities. Photography enthusiasts, including both amateur and professional photographers, share photos and rate those taken by peers in these communities. Datta et al. [26] compiled a dataset containing more than 3,000 images from Photo.net, with scores ranging between 1 (ugly) and 7 (beautiful). The CUHK dataset of Ke et al. [54] contains 60,000 images from DPChallenge.com, with binary labels (good or bad). The AVA dataset of Murray et al. [86] compiles aesthetic judgments from on-line communities of photography amateurs to obtain 250,000 images with human rated aesthetic scores. It has been used for evaluation in the most recent aesthetic assessment research. The dataset presented in this chapter complements these

prior efforts in three ways. First, our data are collected from raw photo albums (rather than harvested from online sharing communities) so the photos themselves are characteristic of personal albums – they have not been pre-filtered for quality by the owners, nor have the photos themselves been edited. Second, our photos are organized into a collection of photo series. Third, our images are complemented by crowdsourced human judgements that allow us to rank photos within a series. These three attributes uniquely tailor our dataset to the problem of triage among photo series.

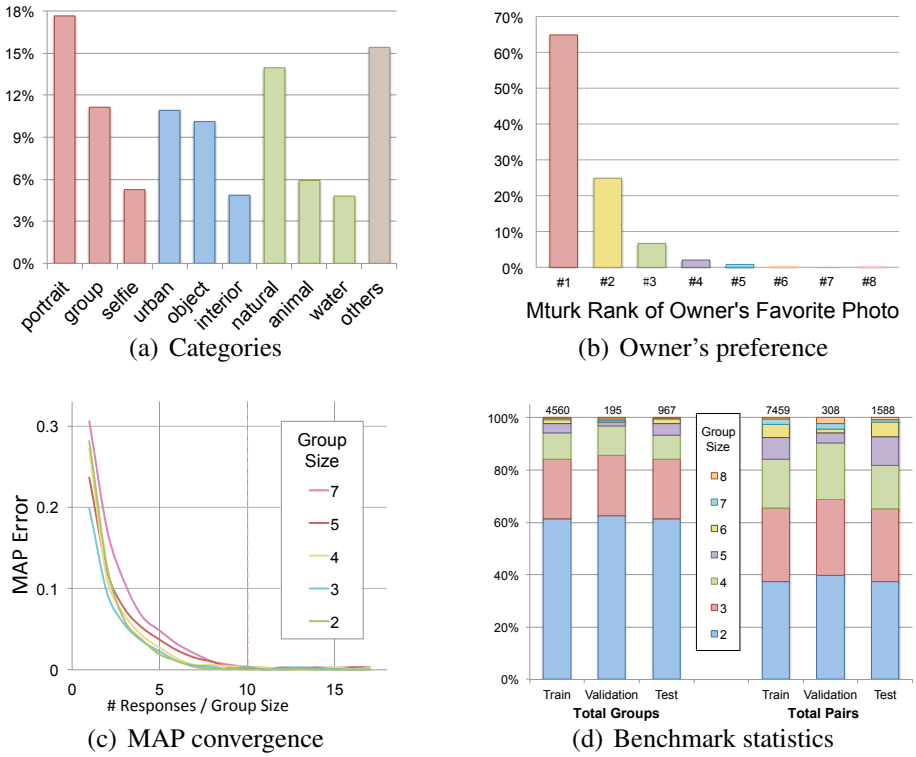
2.3 The Dataset

To understand how human beings evaluate similar photos, we first collect $5,953$ photo series from a large group of contributors, containing $15,545$ personal photos. We then conduct a crowd-sourced user study on people's responses in these series that contains $98,780$ preferences and rationales in total. In this section, we first describe the methods and apparatuses of our data collection and user study process in more detail, then provide an analysis of the user study results.

2.3.1 Collecting Personal Photo Series

Since there's no public dataset to address the problem of photo triage for a series of shots, we describe in this section how we build a dataset specially targeted at this problem. But there are strong restrictions concerning the availability of data imposed by our goals. For example, we intend to collect unedited, complete photo series from personal albums. Such data is hard to harvest online, since most online public photo sharing sites (e.g. Flickr) only contain images that the users have selected to submit, and most users only select a small number of good photos from their photo series to share. Furthermore, a large portion of images submitted to these sites have already been edited or enhanced. Other cloud storage services such as Apple iCloud do store users' raw photo series, but their data is inaccessible.

In order to collect a large scale personal photo series dataset for open academic research, we designed and ran a contest that was open to college students, faculty, and staff. In the contest, we asked participants to submit unedited or slightly edited personal photo albums, where image sizes are larger than 600×800 pixels. The contest lasted for two weeks, and we have collected over 350 album submissions from 96 contributors.



(a) Categories (b) Owner's preference

(c) MAP convergence (d) Benchmark statistics

Figure 2.3: Dataset properties. (a) Distribution of categories is consistent with personal photo collections. (b) Turkers generally agree with the photo contributors' favorite photo in a series. (c) MAP error declines with increasing number of human comparisons, converging by ten times the group size. (d) Distribution of group sizes and labeled pairs within the training, validation and testing sets of our benchmark.

To identify photo series from all the submitted images, we first discard redundant shots, e.g. those captured in a burst session, as the differences among them are too subtle to be treated as a series. We achieve this by sorting images according to the dates in their metadata, and computing the Root Mean Square (RMS) color difference between neighboring images downsampled to 128^2. Duplicates are identified when the

RMS difference is smaller than a threshold (0.06 for normalized colors in $[0, 1]$, found by experimentation).

To find photo series from remaining images, we extract SIFT descriptors and build scene correspondences based on SIFT matching [73] between neighboring images. If two neighboring images have good scene matching, we group them together into a series. However, local feature matching cannot handle special cases such as one of the two images having severe camera motion blur. To handle such cases, we also check the color similarity between two images as another metric to merge images into series, measured by the earth moving distance of the color histograms. Finally, since a large portion of user photos contain human faces, we apply face verification using the Face++ Toolkit [83, 128] in all the merged photo series. For semantic consistency, we expect each series contain the same group of people. If a series contain different groups in different images, we split it into smaller ones based on face identity. We also do not allow a series to have more than 8 images. If a series contains more than 8, we split it by using variant k-means on the 116-dimension global features as depicted in [116], where the center is a representative photo instead of a mean. After gathering all the clusters automatically using the process described above, we manually checked the resulting series to filter clusters with terrible image quality or privacy concerns (such as credit cards).

This data collection and selection process yields $5,953$ photo series: a few examples are shown in Figure 2.1. Together with Figure 2.3(a), they show that our dataset contains a wide variety of photography subjects, including portraits, group and family photos, selfies, urban scenes, objects and food, interiors, natural landscapes, animals and water scenery. They also show that images in our dataset contain large variations in composition, human pose, color and lighting characteristics, etc.

2.3.2 Crowdsourced Ranking

To obtain human preference on images in each photo series, we employ the Amazon Mechanical Turk (MTurk) [57] for a crowd-sourced user study. Given that our goal is to develop a relative photo quality metric instead of an absolute one, we ask participants to perform pairwise comparisons on images from the same series. For each comparison, a pair of photos are randomly selected from a series and are shown side-by-side, each fitted into a 640x640 frame with the original aspect ratio. The question we ask for each pair is: "Imagine you take these two photos, and can only keep one. Which one will you choose, and why?" We used a forced-choice methodology in order to better measure small differences. Participants are also required to fill out at least one of the two forms: one describes the reason(s) why a particular photo is preferred (positive attributes); the other describes the reason(s) why the other photo is not preferred (negative attributes). The purpose of asking for written explanation is two-fold: (1) it forces the participants to make justifiable decisions, thus the quality of gathered user data is higher; (2) the user comments carry insightful information that can guide us on feature extraction.

The pairwise comparison results are used to obtain a global ranking for each image in a series. Denote the pairwise comparison annotations as a count matrix $S = \{s_{i,j}\}$, where $s_{i,j}$ is the number of times that the ith photo I_i is preferred over the jth photo I_j. We use the Bradley-Terry model [8] for obtaining the global ranking, which describes the probability of choosing I_i over I_j as a sigmoid function of the score difference between two photos $\Delta_{i,j} = c_i - c_j$, i.e.

$$P(I_i > I_j) = F(\Delta_{i,j}) = \frac{e^{\Delta_{i,j}}}{1 + e^{\Delta_{i,j}}} \tag{2.1}$$

The score parameter c can be estimated by solving a maximum a-posteriori (MAP) problem. Since we assume the prior is a uniform distribution, the objective is to maximize

$$\log Pr(S|c) = \sum_{i,j} s_{i,j} F(\Delta_{i,j}) \tag{2.2}$$

which could be solved by using gradient descent.

Previous work [129] shows that the convergence of the MAP estimation typically occurs in a linear rather than a quadratic number of pairwise comparisons. We thus conduct a pilot user study to determine the linear coefficient k, i.e. $k(n-1)$ pairwise comparisons need to be collected when the size of the series is n. In the pilot user study, we evaluated the MAP estimation on 15 photo series of different sizes with varying numbers for the linear coefficient. As illustrated in Figure 2.3(c), all MAP averages in different series converge by $k = 10$. Thus in our subsequent study, we gather $\lceil 20/n \rceil$ responses per pair in a series of n photos.

2.3.3 Analysis of Crowdsource Data

We have received $98,780$ responses from MTurk in total. We briefly report some interesting findings from the results here.

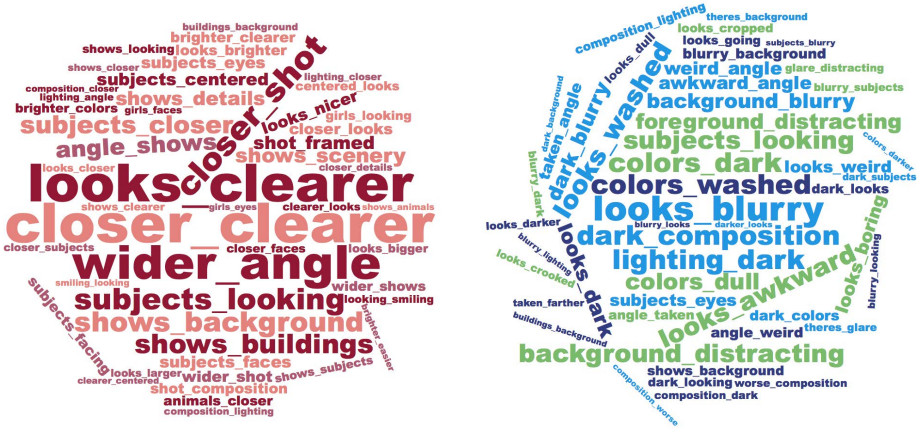
Preferences

It is clear that people have different levels of consistency on evaluating different image pairs. In Figure 2.7, we show example pairs of different levels of agreement. For pairs that have clear human preferences, the compared images often differ significantly in some image features, making automatic prediction easier to succeed. For examples with more diverse options, it is a harder job for an automatic algorithm to make a decision that is consistent with the majority. Therefore, the problem of automatically predicting human preference becomes easier as the level of agreement increases (Figure 2.7b-f). Difficulties are well balanced across our dataset (top of bars).

One may argue that evaluating personal photos can be subjective, depending on whether or not the viewer has prior knowledge about the scene/people in the photos. We conducted an additional experiment to investigate whether MTurk reviewers' preferences are consistent with those of the photograph owners. We selected 551 sampled series from 10 owners, and asked the album owners to select the ones they like most from their own photo series. The results are shown in Figure 2.3(b). In 358 out of 551 series, the owners made exactly the same decisions with the collected intelligence of MTurk reviewers. In addition, in 69.5% of the series in which album owners made different choices, the MTurk reviewers also had relatively low consistency among themselves. We thus conclude that MTurk responses offer a reasonable proxy not just for generic human preferences but also for photo owners, on our dataset.

Comments

To better understand the determining factors when people compare similar images, we extract the most frequent double-word phrases from the comments, including both positive and negative one, as visualized in Figure 2.4.



(a) Reasons for preferred photos (b) Reasons for rejected photos

Figure 2.4: Word clouds visualizing most frequent rationales for photo preference.

For negative reasons, blur, dark lighting, washed colors, and distracting foregrounds or backgrounds were the primary topics of discussion. For positive reasons, less blurry, clearer and closer photos showing more detail, and brighter colors were highlighted. People also liked wide angle shots often. These findings provide a useful guidance for defining features for an automatic recommendation system.

2.3.4 Benchmark

To facilitate future research, we set up an online benchmark

for photo triage which may be found together with our dataset and models at our project webpage[1]. We divide the whole dataset into three subsets. Out of the 5,953 series, 4560 are randomly sampled for training, 195 for validation, and the remaining 967 for testing. Figure 2.3(d) shows a detailed breakdown for each set. We release all of the images but only the human labels for the training and validation sets will be publicly available. New results on the testing set can be submitted through an online interface, and will be evaluated by two metrics: log likelihood and accuracy, which will be explained in more detail in Section 2.5.

2.4 Modeling Preferences

Based on our dataset, we propose a variety of methods for learning and predicting human preference in evaluating photo series, including both feature-based approaches (Section 2.4.1) and end-to-end deep learning (Section 2.4.2). In Section 2.5, we quantitatively compare these methods with previous approaches in the literature.

2.4.1 Feature-based Learning

Extensive work has been done on extracting or learning features for photo aesthetics assessment [87, 53, 127]. To address the new problem of learning human preference

[1]Project Webpage: **phototriage.cs.princeton.edu**.

between a pair of similar images, we first explore assorted features motivated from either heuristics or prior work, including color and lighting statistics, clarity, face factor, composition, and content.

Hand-tuned Features

Color and Lighting As revealed by our user study results, people are usually sensitive to color and lighting differences between two photos of the same scene. For measuring them we compute a standard color histogram feature of 16 bins in L*, a*, and b* channels in CIELAB color space.

Face. From the MTurkers' comments on portrait photos, the dominant facial features are "smiling face", "closer face", "face angle", and "eyes". We detect face position, areas, angle, smileness and eye openness using existing methods [24, 14], and normalize them to $[0, 1]$. Since multiple faces could be detected, we extract the median of all face positions, both the median and max of face areas, angle, smileness, and eye openness as the face feature.

Composition. In previous work, general composition rules such as the rule of third and the rule of diagonals are widely used for photo aesthetics evaluation [68, 27]. These approaches detect the salient objects as foreground regions in an image, and use their locations for evaluating composition. In our problem, aside from detecting salient regions in a single image, we need to further discover the common salient region that appears in both images, and explore its composition difference. To achieve this, we first segment both images into 600 superpixels using Ren et al.'s method [100]. We then compute pixelwise saliency maps using the method proposed by Judd et al. [50], and compute the average saliency for each superpixel. Next, we build non-rigid dense correspondence between two photos [37], and select the common foreground region as a set of salient superpixels that appear in both images. We encode the foreground saliency map as a 900 dimension vector by downsampling to 30×30, and use it as the composition feature.

Clarity. Having motion blur or being out of focus is a common problem in casual photographs. For blur detection, we compute the clarity score using the CORNIA metric [120], in both the entire image and the extracted shared foreground region. This is motivated by the observation that a sharp foreground against a blurry background is often visually pleasing, although its overall clarity score could be low.

Deep Features

Content. Convolutional networks (ConvNet) have been shown to produce high quality image representations. Deep features trained for object classification have been successfully applied to a variety of content-related computer vision tasks such as object recognition, object detection [33, 99], and recognizing image styles [52]. We take features pre-trained on the ImageNet dataset via AlexNet [58] and the 16-Layer VGGNet [106], and extract the response of the second-to-last fully connected layer as our content features. Both feature sets have 4096 dimensions and are computed from center crops of images resized to 256^2.

Combining Features

We concatenate the hand-crafted features with deep features described above from a given photo pair, and preprocess the feature vectors by a whitening transformation. We then train a Random Forest classifier [9] and Support Vector Machines (SVM) with Ridge normalization to learn which image in the pair is more preferable as a binary label. In the Ridge SVM experiment, we set alpha to 1.0 and use the LSQR solver [90]. In the Random Forest method, we fit 100 trees on the sub-samples of the dataset. We evaluate the performance of hand-tuned features, deep features, and the fusion of both in the Section 2.5.

2.4.2 End-to-End Learning

In this section, we explore using ConvNet to design an end-to-end prediction model. Recent success on image representation learning illustrates the power of learning image features directly. Since AlexNet was used in ImageNet image classification challenge, convolutional neuron networks have dominated the challenge leaderboard and been constantly improved [108, 106, 109, 41]. The main benefit of using convolutional neuron networks is that they can learn the image representation and prediction model directly (end-to-end learning). The success of convolution networks demonstrate that the learned representation can outperform the hand crafted features in the classification task.

A natural question is how we can learn image representation for our task. The main difficulty is lack of data. Although we have more than 10,000 images in our dataset, it is still far less than the number of images used to successfully train a deep convolutional network from scratch. Because the network for learning natural image representation usually requires millions of parameters, network training without large amount of data can easily get stuck in bad local minimum. This problem also exists for most of the computer vision tasks. While there are millions of images available for classification, there are fewer images for the other tasks such as object detection and segmentation, since it is much harder to obtain accurate ground truth annotations. One way to solve the training problem is to have good initialization. To achieve this, researchers have tried to adopt the

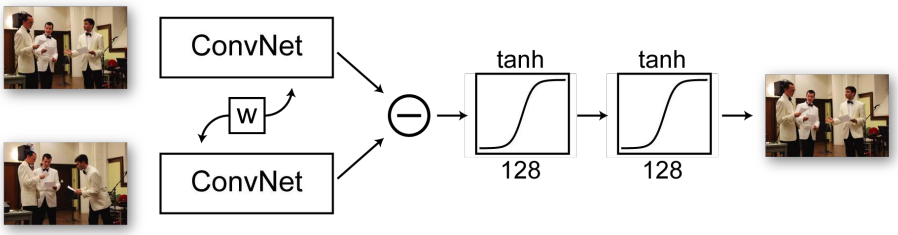


Figure 2.5: Siamese architecture: Features are first extracted from each of the pair of images by two ConvNets with shared weights. Then the difference of the features are passed through two levels of hidden layers, with 128 channels activated by `tanh`*. Finally a two-way* `softmax` *decides which image is preferred.*

network designed for image classification together with the model trained on the millions of images to new tasks such as object detection [32, 98], semantic segmentation [72, 122], memorability [56] and so on. The adoption involves two steps. The network is first extended to a new domain by adding new components [72, 40, 124]. For example, for semantic segmentation in [72], instead of predicting a single label for the whole image, the classification network can be extended to predict a label for each pixel by adding upsampling layers. Then in training, the extended network is initialized by the model parameters pretrained on the image classification task, and then it is fine-tuned with the data for the new task. This fine-tuning scheme is working very well and the resulting algorithms can achieve better performance than those using hand-crafted features. Furthermore, it is found [122] that even if the original network is slightly modified, the pretrained model parameters can still act as effective initialization. Therefore, we try to design a network based on existing architecture and fine-tune the network parameters on our dataset.

To extend the image classification network, we have to consider several differences of our problem. Firstly, the networks for image classification only take one image as input and make the prediction, while our prediction depends on a pair of images. Also, the features for an image should be the same no matter whether the image is the first or second in the pair. This leads us to consider Siamese architecture to learn extracting image features. In Siamese architecture [10], two inputs are sent into two identical sub-networks independently, which share the same network parameters in both training and prediction phases. As shown in [10, 22, 6], Siamese network can learn image embedding when the supervision is the distance between two inputs. However, in our problem, we only have a binary label for which input is preferred. Also, if we swap the two inputs, the label is supposed to be flipped. This motivates us to design a new cost function based on Siamese architecture.

We now put forth the idea more concretely. The input of our model is a pair of images $(I_1, I_2) \in \mathbb{I} \times \mathbb{I}$, where $\mathbb{I}$ is the space of images. We aim at learning a function

$p : \mathbb{I} \times \mathbb{I} \mapsto \{-1, 1\}$, where 1 means the first image is better and -1 means the opposite. The definition requires p be skew-symmetric, that is, $p(I_1, I_2) = -p(I_2, I_1)$. To achieve this, we decompose the prediction function into two stages $s : \mathbb{I} \times \mathbb{I} \mapsto \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \{-1, 1\}$ so that $p(I_1, I_2) = f(s(I_1, I_2))$. Conceptually, s is the n-dimension feature extraction function learned from the input image pairs and f classifies the features computed by s. We observe that if s is skew-symmetric and f is odd, p is skew-symmetric. We use a Siamese architecture to learn s, and use a multi-layer proceptron to learn f. The details of the two stages are explained in the following two paragraphs.

At the first stage of learning s, two input images are first passed to two identical sub-networks with shared weights (parameters). The final output of the first stage is the difference of the outputs of the two identical sub-networks with different inputs. Different from most of the other uses of the Siamese architecture, we don't compute the distance of the outputs from the identical networks, because our label for the input image pair doesn't tell us how similar or dissimilar they are. Instead, we pass the difference to the second stage to classify which image is better. In our implementation, we try both AlexNet and 16-layer VGGNet as the sub-networks and initialize them with the weights trained on ImageNet dataset. However, the last fully connected (FC-1000) and soft-max layers are removed. So the output dimension of each sub-network is 4096. To guarantee that the sub-network can produce the same results when they take the same inputs, Dropout layers are also removed. We find this is helpful for training the network.

At the second stage of learning f, a multi-layer perceptron classifies the features of the image pairs. In our perceptron, there are two hidden layers, each of which has a 128-dimension output. Each hidden layer comprises a linear fully connected layer and a non-linear activation layer. Because f is supposed to be odd, we use $\tanh$ in activation layers. The outputs of the second hidden layer are fed to a two-way Softmax. The outputs of Softmax indicate which of the two input images is better. We do not use Dropout

for regularization because we find that Dropout can prevent the training from reaching convergence in our experiments.

We concatenate these two stages of network and train them together. AlexNet and VGGNet take input images of size 227^2 and 224^2, respectively. Therefore, we resize each image so that its larger dimension is the required size, while maintaining the original aspect ratio and padding with the mean pixel color in the training set. Stochastic gradient decent (SGD) is used to optimize the network with L2 regularization. The learning rate is 0.001. The momentum is 0.9 and weight decay is 0.0005.

2.5 Results

We evaluated different methods at two levels: series-level and pair-level, both visualized in the pair of bar charts in Figure 2.6. The series-level evaluation measures the log likelihood orderings over all series as follows. For a series k, given the human preferences on each image, we identify the one most preferred by humans as $winner(k)$. For a method M, we apply it to all image pairs that include $winner(k)$ and another image in that series. Next we compute the logarithm of the joint probability of the decisions of method M on all such pairs.

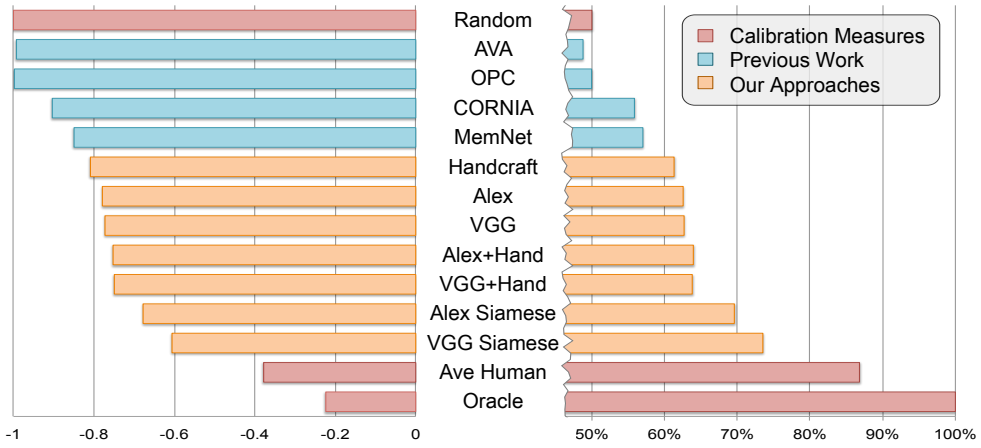
$$\log \mathcal{L}_k(M) = \sum_{i \neq winner(k)} \log P(M, winner(k), i),$$

where

$$P(M,i,j) = \begin{cases} Pr(I_i > I_j), M \text{ predicts photo i better than j} \\ Pr(I_j > I_i), \text{otherwise} \end{cases}$$

$Pr(I_i > I_j)$ is calculated according to Bradley-Terry model (see Section 2.3.2). The resulting log-likelihood value is negative, and visualized in Figure 2.6 relative to a naive baseline of random guessing. The reason we use the log-likelihood measurement instead of the frequency with which a particular method selects the best image is that series of

different sizes have different levels of difficulty – it is harder to choose the best one from larger series than smaller ones.



(a) Log likelihood (b) Accuracy ($\geq$70% agreement)

Figure 2.6: Benchmark performance of various methods, where further to the right indicates better performance in all cases: (a) Log likelihood of predictions made by each method, based on human preferences in the testing set, where values are normalized such that random guessing has value -1. (b) Each method's accuracy in predicting human preference among pairs where humans agree at least 70% of the time. Overall, VGG Siamese is our best performing method, beating our other proposed approaches as well as baseline methods from previous work.

On the pair level, considering that some pairs have a clear human preference while others do not (Figure 2.7), we only choose pairs where the majority agreements is over 70% calculated by Bradley-Terry model as test pairs, and compute the accuracy of different methods on these high-consistency pairs.

In both results shown in Figure 2.6, the "oracle" predictor selects the majority of human votes, while "average human" selects each photo with the probability proportional to the human votes, which represents the average performance of a single person.

We also identify several previous approaches for comparison. CORNIA [120] is a codebook-based approach for non-reference image quality assessment. This work deals with general low-level quality attributes such as noise and blur. Given a photo pair, CORNIA provides a distortion score for each photo and the one with the lower score is considered to be better. Although it uses only low-level features, it performs reasonably

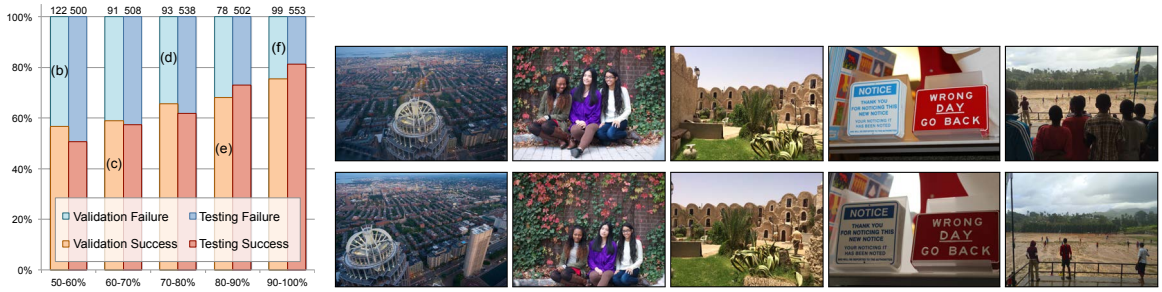
well on our dataset, given that image sharpness is a main factor for comparing similar images, according to the user study results.

RAPID [74] uses a deep neural network to evaluate the aesthetic quality of a given image. Taking cropped and resized photos as features, it produces a probability of whether a given image is high-quality. We use this probability to make the decision when comparing an image pair. While their approach works well on the AVA dataset [86], its performance in our task is close to random guess, which implies that their neural network is incapable of differentiating images that are in similar aesthetic levels and image styles. Liu et al. [68] proposed OPC, which uses several aesthetic guidelines for evaluating photo composition. We use this score as a criteria to compare the aesthetics quality of an image pair. The performance of this method on our dataset is also close to random guess. This is because composition is only one factor among many others when people evaluate similar images, and a few photography rules are not comprehensive enough for describing the subtle composition differences among casual photos.

MemNet [56] computes a global memorability score for each image. It is trained on the largest annotated image memorability dataset (LaMem) that contains 60,000 images. The network architecture is based on AlexNet. The model is pre-trained on scene and object images and then fine-tuned on LaMem. It is shown that MemNet can reach human consistency in predicting memorability. Intuitively, the preferred image in a series may be the most memorable one, so we also evaluate this model on our dataset. Our results show that MemNet outperforms other previous methods, indicating that memorability has a strong connection with how people choose images in photo series.

Figure 2.6 shows that both our feature-based methods and end-to-end learning approaches outperform previous methods by both measures. The methods only using ConvNet features perform better than only using handcrafted features, which implies that mid-level features are important in the photo triage problem. Combining both features further improves the performance. While the hybrid method that combines handcrafted features

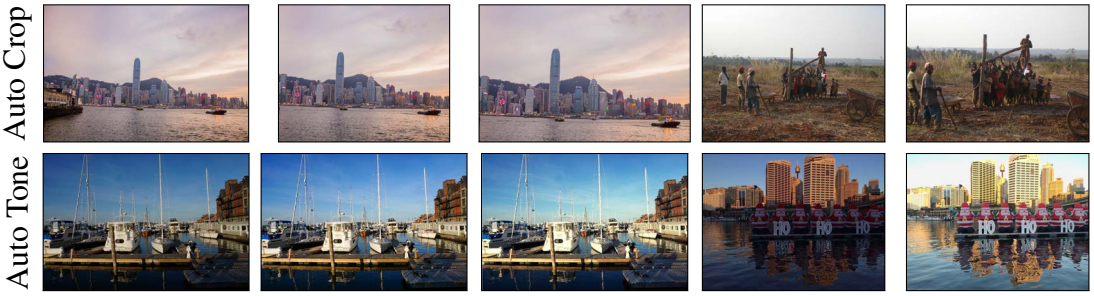
with VGG performs slightly worse than the AlexNet counterpart, the VGG Siamese method achieves the best performance overall – highest likelihood with 73% accuracy.



(a) human agreement (b) (c) (d) (e) (f)

Figure 2.7: Photo pairs in validation and testing sets that have greater human agreement tend to be easier to predict. (a) The performance of our best predictor (Siamese) relative to increasing deciles of human agreement. (b-f) Examples of prediction failures and successes from these deciles, where the upper photo is preferred by the majority of humans.

In Figure 2.7, we further analyze the performance of the end-to-end model on subsets of data with different levels of human agreement. It shows that as human opinions become more consistent, the performance of our model on both validation and testing sets also increases. We pick five examples of photo pairs, each from one human agreement level, to illustrate some success and failure cases. For Figure 2.7(b), our predictor does not pick the slightly better one, but human decisions are also widely divided. For pair (d), people prefer the upper one, perhaps due to a small back-facing person in the lower photo; but our method fails to recognize it given the current Siamese network takes low resolution images as input. For pair (f), the upper scene is heavily occluded, but people prefer it because it shows that the game is well-attended. Our model cannot capture this high level, semantic image interpretation. The successful predictions of our model on pairs (c) and (e) show that our model can make correct decisions based on different factors such as composition and color in each specific case.



(a) Original (b) Commercial (c) Ours (d) Nearest (e) Nearest Better

Figure 2.8: Automatic enhancement. The original (a) is improved either by a commercial product (b) or our proposed method (c). In the proposed method, a nearest neighbor (d) is found in the dataset where a better alternative (e) from the same series implies the transformation that can be applied to the original (by analogy) to improve it.

2.6 Applications

A direct application of our work is to develop smart album viewers. In addition to providing faster photo triaging, a smarter viewer could automatically identify photo series, and show only the top images for each series instead of presenting all images in it to the user at once. This allows a more efficient interface to manage photo albums and the photo series in them. Beside triage, here we describe a few other photo and video editing applications that can benefit from the proposed dataset and methods.

Automatic Photo Enhancement

We have discovered that in our dataset, there are plenty of image pairs where one photo is strongly preferred over the other for only one major reason, such as composition, color,

	Original	Commercial	Ours
Auto Crop	22%	29%	49%
Auto Color	9%	48%	43%

Table 2.1: Comparison of our proposed automatic cropping method with AutoCrop feature in Photoshop Elements, and our automatic color enhancement with AutoTone in Lightroom. Humans compared the original and two automatically enhanced photos. Percentages are fraction of photos where a particular variant was most preferred of the three options. Our cropping method performs best. Even though humans prefer the Lightroom enhancement most often, ours is still preferred in many cases.

or exposure. Such pairs are valuable examples to learn not only human preference in each aspect, but also the corresponding transformation for improving human preference that can be applied to a new photo. We present a new example-based photo enhancement framework similar in spirit to Image Analogy [43]. We demonstrate two applications of this framework: auto crop and color correction.

We first build a dataset of photo pairs for each application. We collect photo pairs such that for each pair after applying a transformation (color or crop), the less favored photo (B_i for "bad") becomes visually similar to the preferred one (G_i for "good"). Next, given a new photo N, we search through all pairs in the dataset and find the pair i for which B_i is most similar to N (by L_2). Then we can automatically apply a transformation to make an enhanced image E by the analogy $B_i : G_i :: N : E$.

The search for the nearest B_i depends on the operation. For color correction, we concatenate L*a*b* color histogram and content features introduced in Section 2.4.1. For auto crop, we only consider images B_i that have roughly the same aspect ratio as N, and extract GIST features with 8 blocks (images resized to 256^2) for finding the nearest neighbor. (We also tried content features as we did for color correction, but in our experiments found GIST features perform better for describing the geometry for cropping purposes.)

Next we compute a transformation from the selected photo pair and apply it to N to produce E. For cropping, we (1) use SIFT matching [73] to find a perspective transformation from B_i to G_i; and (2) if this is nearly a similarity transformation, we choose the crop that best approximates it. For color correction, we use the NRDC approach [37] to find a global color mapping from B_i to G_i. Figure 2.8 shows two examples of enhancements made by this method, and more examples may be found on our project webpage **phototriage.cs.princeton.edu**.

To objectively evaluate the proposed editing framework, we apply it on two new test datasets: one with 468 images for cropping; the other with 556 images for color correction.

We conduct additional MTurk user studies to compare our results against the originals, as well as the automatic enhancement results produced by Adobe "Auto Crop" and "Auto Color" features. The results are reported in Table 1, suggesting that our results are in general quite comparable to those produced by existing commercial tools that are dedicated for these tasks. Although the nearest image B_i found in this relatively small dataset often has very different content than the input N, they share similar characteristics in either color or composition. Thus the computed transformation can often produce reasonable results. Nevertheless, we expect the performance of this approach to improve with a larger dataset.

Video Filmstrips

A number of applications rely on a "filmstrip" that summarizes a video clip. These are often shown in video browsing and editing software as well as browsers for video on smart phones. Uniformly sampling frames from the video sometimes chooses good frames and sometimes bad ones, essentially by luck. A line of research seeks to summarize video by

Figure 2.9: Video Filmstrips. Uniformly sampling in time (first and third rows) gets a mix of good and bad frames. Our approach (second and fourth row) uses our learner to pick the "best" frame from a region around each uniformly sampled frame. Video frames courtesy of the Flickr users Oleg Sidorenko (upper) and Juhan Sonin (lower).

various approaches, for example modeling human attention [79]. We propose a simple alternative approach based on our learned model of human preference among a series of photos. We extract sequential groups of keyframes from the video, uniformly sampled in time, and choose the "best" from each series using our learned model for preference. This approach is compared with uniform sampling in Figure 2.9. These examples are about 80 sec, so each frame of the filmstrip represents about 16 secs or 480 raw frames of video (from which our method chose the "best" of around 30 keyframes). In one case the same frame is chosen, but more often than not we find the frames selected from our video are indeed better than the uniformly sampled ones.

Chapter 3

Photo Recolor

Figure 3.1: Palette-based photo recoloring. From left: original (computed palette below); user changes green palette entry to red (underlined), and the system recolors photo to match; user changes multiple colors to make two other styles.

3.1 Introduction

Research and commercial software offer a myriad of tools for manipulating the colors in photographs. Unfortunately these tools remain largely inscrutable to non-experts. Many features like the "levels tool" in software like Photoshop and iPhoto require the user to interpret histograms and to have a good mental model of how color spaces like RGB work, so non-experts have weak intuition about their behavior. There is a natural tradeoff between ease of use and range of expressiveness, so for example a simple hue slider, while easier to understand and manipulate than the levels tool, offers substantially less control over

the resulting image. This chapter introduces a tool that is easy for novices to learn while offering a broad expressive range.

Methods like that of Reinhard et al. [96] and Yoo et al. [121] allow a user to specify complex image modifications by simply providing an example; however an example of the kind of change the user would like to make is often unavailable. The method of Liu et al. [71] allows users to modify the global statistics of an image by simply typing a text query like "vintage" or "new york." However, for many desired color modifications it is hard to predict what text query would yield the desired effect. Another challenge in color manipulation is to selectively apply modifications – either locally within the image (e.g., this hat) or locally in color space (e.g., this range of blue colors) instead of globally. Selection is particularly challenging for non-experts, and a binary selection mask often leads to visual artifacts at the selection boundaries.

Our approach specifies both the colors to be manipulated and the modifications to these colors via a *color palette* – a small set of colors that digest the full range of colors in the image. Given an image, we generate a suitable palette. The user can then modify the image by modifying the colors in the palette (Figure 3.1). The image is changed globally such that the chosen colors are interpolated exactly with a smooth falloff in color space expressed through radial basis functions. These operations are performed in LAB color space to provide perceptual uniformity in the falloff. The naive application of this paradigm would in general lead to several kinds of artifacts. First, some pixels could go out of gamut. Simply clamping to the gamut can cause a color gradient to be lost. Therefore we formulate the radial falloff in color space so as to squeeze colors towards the gamut boundary. Second, many natural palette modifications would give rise to unpleasant visual artifacts wherein the relative brightness of different pixels is inverted. Thus, our color transfer function is tightly coupled with a subtle GUI affordance that together ensure monotonicity in the resulting changes in luminance.

This kind of color editing interface offers the best creative freedom when the user has interactive feedback while they explore various options. Therefore we show that our algorithm can easily be accelerated by a table-based approach that allows it to run at interactive frame rates, even when implemented in javascript running in a web browser. It is even fast enough to recolor video in the browser as it is being streamed over the network.

We perform a study showing that with our tool untrained users can produce similar results to those of expert Photoshop users. Finally, we show that our palette-based color transfer framework also supports other interfaces including a stroke-based interface, localized editing via a selection mask, fully-automatic palette improvement, and editing a collection of images simultaneously.

3.2 Related work

Color Transfer (Example based Recoloring)

Researchers have proposed many recoloring methods requiring an example image as input Reinhard et al. [96] exploit the Lab color space and apply a statistical transformation to map colors from another image. When the reference is dissimilar, users need to manually point out the region correspondence by swatches. Tai et al. [118] modify a Gaussian mixture model by adding spatial smoothness and do parametric matching between source and reference images. Chang et al. [18] classify pixels into basic color categories (experimentally derived), then match input pixels to reference pixels within the same category. HaCohen et al. [38] utilize dense correspondences between images to enhance a collection. Their results are compelling but the dependence on compatibility between images is high. Yoo et al. [121] find local region correspondences between two images by exploiting their dominant colors in order to apply a statistical transfer. It is important to note that all these methods require a reference image as input, which needs to be provided by the user or produced by another algorithm.

Automatic Color Enhancement

Bychkovsky et al. [12] build a large retouching dataset collected from professional photographers to learn an automatic model for tone adjustment in the luminance channel. Cohen-Or et al. [23] propose to automatically enhance image colors according to harmonization rules. However, the user can only control the hue template type and rotation, which is not flexible enough for our needs. Hou and Zhang [44] provide several concepts for users to change the mood of an image. The concepts are extracted by clustering hue histograms for different topics. They only provide 8 concepts, which limits their transformation and editing styles. More flexibly, Wang et al. [115] and Csurka et al. [25] use a semantic word to describe a desired editing style or emotions. The semantic word is automatically quantified by a color palette, however, people cannot set the target palette directly. Shapira et al. [104] enable users to explore editing alternatives interactively using a Gaussian mixture models (GMM). In Section 3.4 we compare against a GMM based algorithm.

Edit Propagation (Stroke Based Recoloring)

Stroke-based methods [59, 94, 4, 63, 61, 62] propose to recolor images by drawing scribbles in a desired color on different regions, automatically propagating these edits to similar pixels. Levin et al. [59] allow UV changes (in YUV). Qu et al. [94] are specific for manga recoloring. An and Pellacini [4] propose to approximate the all-pairs affinity matrix for propagation, and Xu et al. [119] further accelerate it by using adaptive clustering based on k-d trees. Chen et al. [19] propose sparsity-based edit propagation by computing only on a set of sparse, representative samples instead of the whole image or video. This accelerates and saves memory, especially for high-resolution inputs; we compare against this approach in Section 3.4. Section 3.5.5 describes our unified framework that uses both stroke-based and palette-based interactions.

Palette based Recoloring

A recent work [66] proposes a method for coloring vector art by palettes based on a probabilistic model. They learn and predict the distribution of properties such as saturation, lightness and contrast for individual regions and adjacent regions, and use the predicted distributions and color compatibility model by [88] to score pattern colorings. Wang et al. [113] adapt the edit propagation method in An and Pellacini [4] to obtain a soft image segmentation and recolor an image. While our method works for a pair of initial and final values for each palette entry, they only have the final palette colors. Thus the bulk of their method addresses how to associate pixels in the image with the final palette colors (which can be ill-posed, and also leads to a complex and slow method).

3.3 Approach

This section introduces a new, simple approach for palette-based photo recoloring. Section 3.3.1 describes our user interface, as motivated by a set of explicit goals that support non-expert as well as expert users. Second, Section 3.3.2 introduces a clustering approach based on k-means suitable for creating an initial palette from a photo. Section 3.3.3 describes the goals that motivate our color transfer algorithm, which is introduced in the subsequent two sections. Section 3.3.4 describes our approach for preserving monotonicity in luminance, while Section 3.3.5 introduces our color transfer algorithm. Finally, Section 3.3.6 shows a table-based acceleration that allows the algorithm to run at interactive rates.

3.3.1 User interface

These criteria are important for a color manipulation user interface:

Figure 3.2: Our GUI shows a representative color palette. The user adjusts a selected palette color via an HSL controller, and the image updates interactively.

Simple. The GUI should be simple enough to learn and use, even for non-expert users. For example it should not require a deep understanding of color theory or various color spaces.

Expressive. The GUI should offer sufficient degrees of freedom that it is possible to achieve what the user wants.

Intuitive. Assuming there exist *some* settings that achieve what the user wants (previous goal), the user should be able to find them quickly and easily.

Responsive. The GUI should produce results at interactive frame rates so as to facilitate creative freedom in exploration and experimentation.

While there are many existing tools that allow users to recolor images, to our knowledge none of them simultaneously achieve these goals (especially when coupled with the algorithmic goals described in Section 3.3.3). Sliders like those in Photoshop that adjust *hue, saturation,* and *lightness* (or similarly in iPhoto *exposure, contrast, saturation, temperature* and *tint*) while they are simple, responsive, and (to a lesser extent) intuitive, they are not sufficiently expressive to achieve many of the effects shown in this chapter. On the other hand, histogram adjustment methods like the *levels tool* in Photoshop provide

huge expressive range at responsive rates, but are neither simple nor intuitive. Methods that match the statistics of a reference image (e.g. [121, 71]) address all four goals above, but succeeds for expressiveness only when the user already has an example of what they want.

We describe a palette-based GUI, shown in Figure 3.2. When a photo is loaded into the application, a palette is automatically generated (Section 3.3.2). The user needs only to click on a palette color (C) and change it (to C') via a HSL color picker. As the user interactively adjusts C' in the color picker, the overall color statistics of the photo are smoothly adjusted such that pixels colored C in the original photo become C'. Our interface meets all four criteria above. Section 3.4 shows that novice users are able to learn the interface in a few minutes and then quickly produce edited images that are qualitatively (and numerically) similar to those produced by experts in Photoshop. This chapter is not the first to describe this kind of palette-based recoloring. However, some existing palette based approaches are not sufficiently responsive while others are less expressive than ours, as discussed in Section 3.4.1.

3.3.2 Automatic Palette Selection

This section describes our automatic approach for creating a palette based on an image, using a variant of the k-means algorithm. Our goal is to select a set of k colors $\{C_i\}$ that distill the main color groups in the image, to be used as "controls" during editing. The choice of k matters, and often depends on the image as well as the user's desired modifications. If k is too small, then some colors to be changed might not be well-represented among $\{C_i\}$. On the other hand if k is too large, the user may have to change a large subset of $\{C_i\}$ to get a desired change. There are automatic methods for choosing k (e.g., [92]), but in our application this choice depends heavily on the user's intentions. Thus we leave the choice of k up to the user. We find that $k \in [3, 7]$ works well for typical operations, and use $k = 5$ by default.

The literature describes a number of methods for creating a palette from an image. Based on a large dataset of user-rated "color themes," O'Donovan et al. [88] build a measure of the compatibility of a set of colors as well as a method for extracting a theme from an image. Because the dataset they use is targeted at graphic design applications, we find the method tends not to produce high-quality palettes for natural photographs. Lin et al. [65] describe a method for creating color themes that works well for natural photos, built on a study of how people do so. The study assumes a palette of exactly five colors and thus their model intrinsically uses $k = 5$, as does that of O'Donovan et al. [88]. However, as discussed above, we find that for many editing goals a different k works better. While the $k = 5$ assumption is not a fundamental limitation of these previous approaches, they rely on large datasets that do assume $k = 5$ so to change k would require new datasets. Moreover, their methods are slow to compute the palette. Therefore we propose a simpler approach that produces comparably good palettes (at least for our application), but works for any choice of k. Shapira et al. [104] describe a method for clustering image pixels based on a Gaussian mixture model (GMM). The straightforward application of GMM is too slow for interactive use with moderately large images. While acceleration options are available for GMM, we describe a variant of k-means that is already faster than GMM (by a factor of about three for megapixel images, but still too slow in typical cases) and then further improve its performance. In Section 3.4 we compare our method to the GMM-based approach.

In a naive application of k-means to the colors in an image, each iteration of the algorithm touches each pixel in the image, which is costly for large images. Researchers have identified various opportunities to accelerate k-means, for example by organizing the data in K-D trees [51]. Exploiting the property that our data are colors restricted to $R, G, B \in [0, 1]$, we assign them to bins in a $b \times b \times b$ histogram (we use $b = 16$ in RGB). For each bin we compute the mean color in Lab space, and these b^3 colors c_i (or less because some bins may be empty) are the data we use for k-means – typically at

Figure 3.3: Automatic palette methods: (D) O'Donovan et al. [88] build on a color theme dataset targeted at graphic design, which is not ideal for natural photos. (L) Lin et al. [65] acquire and build on a dataset for natural imagery. Both of these methods take more than one minute, and also assume a palette of size $k = 5$ (so are omitted from the third column). (G) Gaussian mixture models (GMMs) used by Shapira et al. [104] are also slow to compute (10 sec). (K) K-means is faster (2 sec) but is non-deterministic and often yields too many dark colors. (O) Our method takes 60ms.

least a couple orders of magnitude smaller than the number of pixels in the image, and now independent of image size. Because each data point c_i now represents the n_i pixels associated with that bin, we use a *weighted* mean (weighted by n_i) when finding each of the k means in each iteration. The weighted k-means approach has been used in other applications, for example the automatic stippling method of Secord [102].

For generating color palettes for our GUI, k-means suffers from two related problems: the basic formulation uses randomly selected data points to initialize the means, and the convergence of the algorithm can be sensitive to this initialization (as discussed by Pelleg and Moore [92]). To present the most helpful GUI to the user, we prefer that the algorithm be deterministic, and that palette colors be far from one another. Thus, instead of randomizing, we initialize the means as follows. We initialize the first "mean" as the color c_i representing the bin with the largest weight n_i. Next we attenuate all other weights n_j

Figure 3.4: More recoloring examples. Originals on left, followed by various edits with our system including both local changes (e.g. make the girl's shirt turquoise) and global changes (e.g. change the image tone to orange).

by a factor $(1 - \exp(-d_{ij}^2/\sigma_a^2))$ where d_{ij} is the distance in Lab space from c_i to c_j and σ_a expresses a falloff (we use $\sigma_a = 80$ which is 80% of the distance from black to white, and have found the algorithm to be relatively insensitive to this parameter). Next we choose bin with the highest remaining weight n_i, repeating until k initial "means" have been chosen. This approach is deterministic and initializes the k-means with large clusters that are far from each other.

Finally, we have observed that choosing a palette based on pixel color clustering often leads to a very dark (near black) palette entry, because typically a significant number of image pixels are dark. However, editing the dark palette entry is rarely fruitful for the user, because dark colors are hard to distinguish (regardless of hue and saturation). Essentially it offers the user a set of controls with little or no effect. Therefore we discourage very dark palette entries as follows. Rather than computing k-means, we actually compute $(k + 1)$-

means, where one of them is initialized and perpetually locked to black. The darkest colors in the image will be assigned to this mean, and will therefore not pull the other nearby means towards black. After computing the $(k+1)$-means, we discard the black entry to leave k remaining palette colors. For a comparison of our approach with the previously described methods, see Figure 3.3.

We find that this algorithm produces useful palettes, and all of the results shown in this chapter and accompanying materials (including the study described in Section 3.4) use this approach, except for Figures 3.8 and 3.13 which require other methods. However, our GUI also allows the user to select some or all of the palette entries explicitly by a color picker or by clicking on the image. Any remaining unspecified palette entries are then automatically chosen via the algorithm described above.

3.3.3 Color Transfer Goals

Now we have a set of palette colors $\{C_i\}$, and our GUI allows the user to modify the palette colors to $\{C'_i\}$ as a way of adjusting the colors in the image. That is, the association $(\{C_i\}, \{C'_i\})$ defines a transfer function f that maps colors in the original image to colors in the edited image. We assume that f acts on colors independently of pixel location or context in the image. Some color mapping approaches relax this assumption, for example the high-quality high dynamic range compression approach of Fattal et al. [29] which operates in the gradient domain. Nevertheless, for our application the assumption is reasonable and permits the acceleration method described in Section 3.3.6.

Here we identify some key properties we would like in f that address the GUI criteria described at the beginning of Section 3.3.1, particularly the "expressive" and "intuitive" qualities:

Interpolation. Any pixel in the original image with the same color as one of the original palette colors C_i should be transformed exactly as the user changed the palette color: $f(C_i) = C'_i$.

In Gamut. The output should remain in gamut $\mathcal{G} : f(p \in \mathcal{G}) \in \mathcal{G}$.

Pixel Continuity. The transfer function should be continuous with respect to pixel color p: $\lim_{q \to p} f(q) = f(p)$.

Palette Continuity. The function should be continuous with respect to changes in the palette: $\lim_{\bar{C} \to C'} f_{\bar{C}}(p) = f_{C'}(p)$.

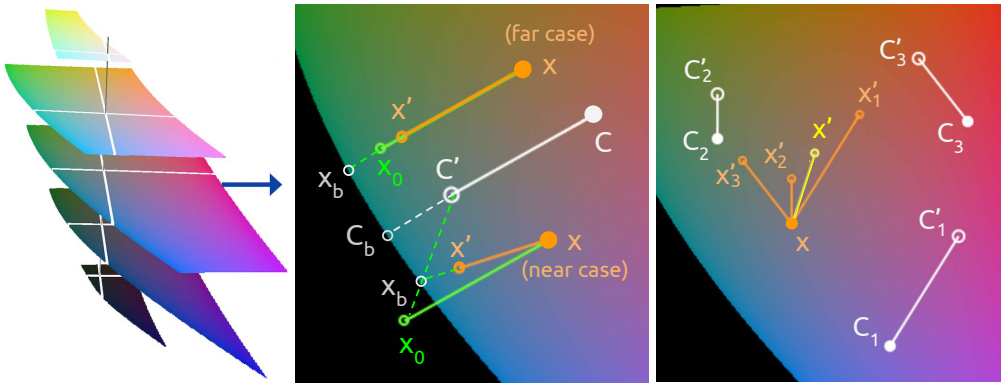
One-to-One. The function should be one-to-one. Together with the continuity requirement this prevents the function from folding over itself, which could lead to counter-intuitive behaviors. Formally: $f(p) = f(q) \implies p = q$.

Monotonicity in L. The transformation in the luminance L should be monotonic. In our early prototype applications we found that transformation functions that allowed relative brightness of different pixels to flip often led to undesirable imagery. Thus we require: $L(p) < L(q) \implies L(f(p)) <= L(f(q))$.

Dynamic Range A gradient in the input maps to a gradient in the output (not a single value). For example an algorithm that shifts all colors in some direction uniformly and then clamps to the gamut is bad by this criterion.

The literature describes a variety of transformation models that might be used for this application, for example Gaussian mixture models (GMM) [84], histogram methods [71], or non-parametric approaches [39]. However, to our knowledge no existing approaches can be easily adapted to meet the requirements listed above. For example, GMM-based editing can easily send colors out of gamut. Of course it is easy to clamp to the gamut boundary but this would violate the one-to-one and dynamic range qualities.

To make matters worse, the requirements themselves are inconsistent. For example, it is not possible to simultaneously satisfy the interpolation and monotonicity requirements. If the user changes the relative brightness of two palette colors, interpolating those colors in the resulting image would violate the monotonicity requirement. However, we would



(a) Lab slices (b) edit one color (c) RBF blending

Figure 3.5: Transfer in AB space: (a) constant-L slices of Lab space; (b) when a single palette color C is changed to C' the resulting editing effect at different locations x; (c) how the effects of changing multiple palette colors C_i are blended at location x using RBFs.

like to satisfy these requirements insofar as is possible, and the following section directly addresses the monotonicity concern.

3.3.4 Monotonic Luminance Transfer

Our application preserves monotonicity in luminance via two mechanisms, one in the GUI and one in the transfer function.

First, the GUI constrains the relative *ordering* of luminance L'_i of the edited palette colors C'_i. That is, suppose that $L_{i<j} < L_j$ in the input palette. Then the GUI constrains $L'_{i<j} < L'_j$ in the edited palette, as follows. Whenever the user modifies C'_i, the GUI also sets $L'_{j>i} = \max(\bar{L}'_j, L'_{j-1})$, where $\bar{L}'_j$ is the most recently user-edited value for palette entry j (or the initial value, if never edited). To change the luminance of a palette entry we simply modify the L channel in Lab space. This operation is evaluated in increasing order for all palette entries $j > i$; and the symmetric operation (involving $\min$ and L'_{j+1}) is applied in decreasing order for entries where $j < i$. This policy has the nice property that if the user brightens L'_i in such a way that $L'_{j>i}$ is also brightened, but then the user reverts L'_i back to the original value, then L'_j also reverts (avoiding hysteresis).

input naive transfer our transfer

Figure 3.6: A naive transfer function that copies the color offset from the palette to every color in the image, even if clamped to remain in gamut, reduces dynamic range in the image (middle). Our transfer function (right) is able to make better use of the dynamic range.

The second aspect of our treatment of luminance is that we design a transfer function that has two orthogonal components – f_L (which modifies pixel luminance based on the palette luminance) and f_{ab} (which modifies the corresponding ab values, and is discussed in the next section). The luminance transfer function simply takes a weighted combination of the two nearest palette entries (or one nearest entry and either black or white if the pixel is darker or brighter than all the palette entries).

With regard to luminance, these two strategies together ensure that both the interpolation and monotonicity requirements are satisfied as well as some others – in-gamut and the two continuity requirements. However, this approach can violate the one-to-one and dynamic range requirements, because a range of shades of gray can be collapsed into a single luminance value when one or more palette colors are pushed to the same luminance. However, we have found through experimentation that these concerns are less critical than interpolation and monotonicity. While it might be possible to satisfy all four requirements by disallowing changes in luminance in the GUI, we find this is an important feature for expressive control.

3.3.5 Color Transfer (ab)

In the last section we devised a simple luminance transfer function f_L that adjusts the luminances of pixels based on those of the palette. In this section we introduce a more complex transfer function f_{ab} that plays an analogous role in the ab channels. The design of this function is not guided by the monotonicity concern, but it does target the other requirements.

First we devise the function f_1 for the simple case where the original palette contains a single color C, and the user modifies it to be color C' (Figure 3.5b). For any color x we would like to know $x' = f_1(x)$. In general we want to translate colors in the same direction, and a naive strategy might simply add exactly the same offset vector $(C' - C)$ to every x. However, it would be easy to go out of gamut, and simply clamping to the nearest in-gamut value would violate the one-to-one and dynamic range goals, as illustrated in Figure 3.6. Instead we devise a scheme that translates colors that are far away from the boundary of the gamut, but squeezes values nearer to the boundary towards the boundary and towards the C' as follows. First we find C_b, the point where the ray from C towards C' intersects the gamut boundary. Next, we determine if $x_o = x + C' - C$ is in gamut. If so (the "far" case) we find x_b the location where the parallel ray from x intersects the gamut boundary. If not (the "near" case) we take x_b to be the point where the ray from C' towards x_o intersects the boundary. These intersections are found by binary search. Finally, we take $f_1(x) = x'$, the point on the ray from x to x_b such that:

$$\frac{||x' - x||}{||C' - C||} = min(1, \frac{||x_b - x||}{||C_b - C||})$$

In the far case this policy squeezes $f_1(x)$ toward the boundary in proportion to $C' - C$ with a maximum ratio of 1, meaning very far from the boundary we just translate colors parallel to the palette change. Very close to the boundary, the offset vectors swivel towards C', which we found by experimentation helps to achieve a desired palette change in those

areas. Also note that this transfer function enjoys several of the desired properties descried above. Assuming C and C' are in gamut, so is $f_1(x)$. It interpolates: $f_1(C) = C'$. If the gamut were convex, then it would one-to-one and continuous with respect to x and C'. Of course the gamut is only *close* to convex, so these properties are almost satisfied. While it is possible to devise cases where they are violated, we find in practice that it does not happen when performing reasonable color palette edits.

Now that we have $f_1(x)$ we will generalize it to handle the case of larger palettes containing $k > 1$ entries. Our strategy is to define k transfer functions $f_i(x)$, each equivalent to the $f_1(x)$ map described above as if it were the only palette entry, and then blend them, weighted by proximity:

$$f(x) = \sum_i^k w_i(x) f_i(x) \quad \text{and} \quad \sum_i^k w_i(x) = 1$$

For the weights we use radial basis functions (RBFs):

$$w_i(x) = \sum_j^k \lambda_{ij} \phi(||x - C_j||)$$

We tried various kernel functions and found that the Gaussian kernel works well in our application:

$$\phi(r) = exp(-r^2/2\sigma_r^2)$$

where the scalar parameter σ_r is chosen to be the mean distance between all pairs of colors in the original palette. The k^2 unknown coefficients λ_{ij} are found by solving a system of k^2 equations: k of which require $w_i(C_i) = 1$ and $k^2 - k$ of which require $w_{j \neq i}(C_i) = 0$.

This approach leads to smooth interpolation of the individual transfer functions f_i at C_i. Unfortunately solving the system of equations set up by the RBFs can lead to negative weights, with two potential hazards. First, it will add some component of the *opposite* behavior of some palette changes. Second, and more dangerously, it can throw the result

out of gamut. We therefore use a simple fix – we clamp any negative weights to zero and renormalize the non-zero weights. We find in practice this solution works well. The final weighted combination is illustrated in Figure 3.5c for a palette size of $k = 3$.

3.3.6 Acceleration

The RBF interpolation scheme described in Section 3.3.5 is relatively fast, but its naive application to the image would require making this computation for every unique color in the image (often in the *millions*). This section describes an acceleration scheme that allows it to be usable in an interactive application. First, note that the weights $w_i(x)$ are found based only on the color x and the initial palette colors C_i. Therefore, in principle the RBF computations need only be performed when the initial palette is established (not during color palette editing). We further accelerate the computation by caching these weights w_i only at a $g \times g \times g$ grid of locations uniformly sampled in the RGB cube. Next, during color

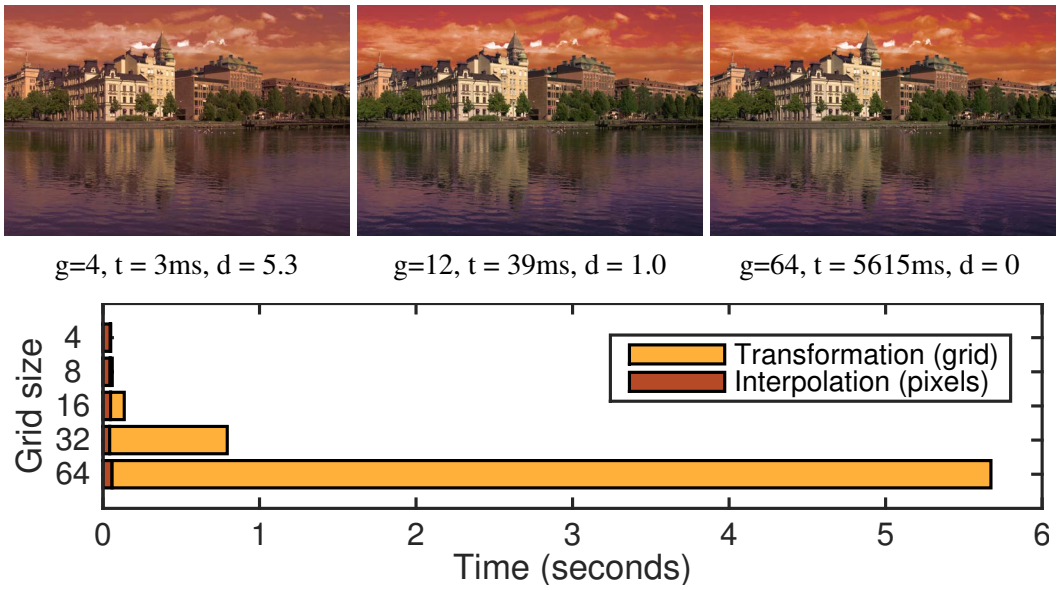


Figure 3.7: Acceleration. Top: results of varying grid size g. Time t is to update the grid values (averaged). d is CIEDE distance from the $g = 64$ version. Small grid sizes give visual differences, whereas $g = 12$ is indistinguishable from $g = 64$ (but roughly 140 times faster). Bottom: Update time vs. grid size. The time to update the pixels of the image remains near constant (brown, about 50ms for a 1 MP image).

editing when the C_i' are known, we can compute the output color $f(x)$ at these g^3 locations using the precomputed weights. Finally, to recolor each pixel in the image use trilinear interpolation on eight nearest grid values. This strategy not only gives our implementation interactive performance (in javascript running in a web browser), but also ameliorates any small discontinuities in $f(x)$ due to the non-convex gamut.

Figure 3.7 demonstrates the performance advantage. We see that using larger grid sizes g has a diminishing return in image quality such that $g = 12$ is almost indistinguishable from $g = 64$, while running about 140 times faster. Without acceleration, our runtime is Tp where T is the time to transform one color and p is the number of pixels. With acceleration, runtime is $Tg^3 + Ip$ where I is the time to perform trilinear interpolation of grid samples, and $I \ll T$. Throughout this chapter and in our demo we use $g = 12$, because it performs well across a range of computers and images, so $g^3 \ll p$. Moreover, while in principle for very large images p could grow to overwhelm the runtime, in practice p is limited by how many pixels we could reasonably "preview" on the screen. Finally we note that splitting the calculation to a pre-processing step followed by sampling and interpolation steps lends itself well to shader programming. While we did not test this idea, implementing the interpolation as an OpenGL shader would be trivial, and should achieve substantially faster performance, even for huge images or dense grids.

3.4 Evaluation

In this section we evaluate our method in two ways. Section 3.4.1 directly compares our approach to some existing methods in the literature. Section 3.4.2 presents a user study in which we ask novice users to use our method, using two alternate methods as a baseline.

Figure 3.8: Comparison to the methods of (top to bottom) Chen et al. [19], Shapira et al. [104], and Wang et al. [113]. Left to right: original photo, their result, our result. We used the same initial and final palettes as in their paper, except for Wang et al., who do not use an initial palette.

3.4.1 Other Recoloring Methods

Figure 3.8 shows a comparison to three other palette based recoloring methods, and in each case we believe our method responds more faithfully to the specified palette. The method of Chen et al. [19] (top) offers a fast and space efficient recoloring algorithm that also works for video. Our method is faster, e.g., at least $10\times$ for Big Buck Bunny. The method of Shapira et al. [104] (middle) focuses an interface for exploration, in the spirit of the *design galleries* of Marks et al. [82]. Wang et al. [113] (bottom) address the harder problem of how to map the colors when no initial palette is present, and thus their method is more complex and slower than ours.

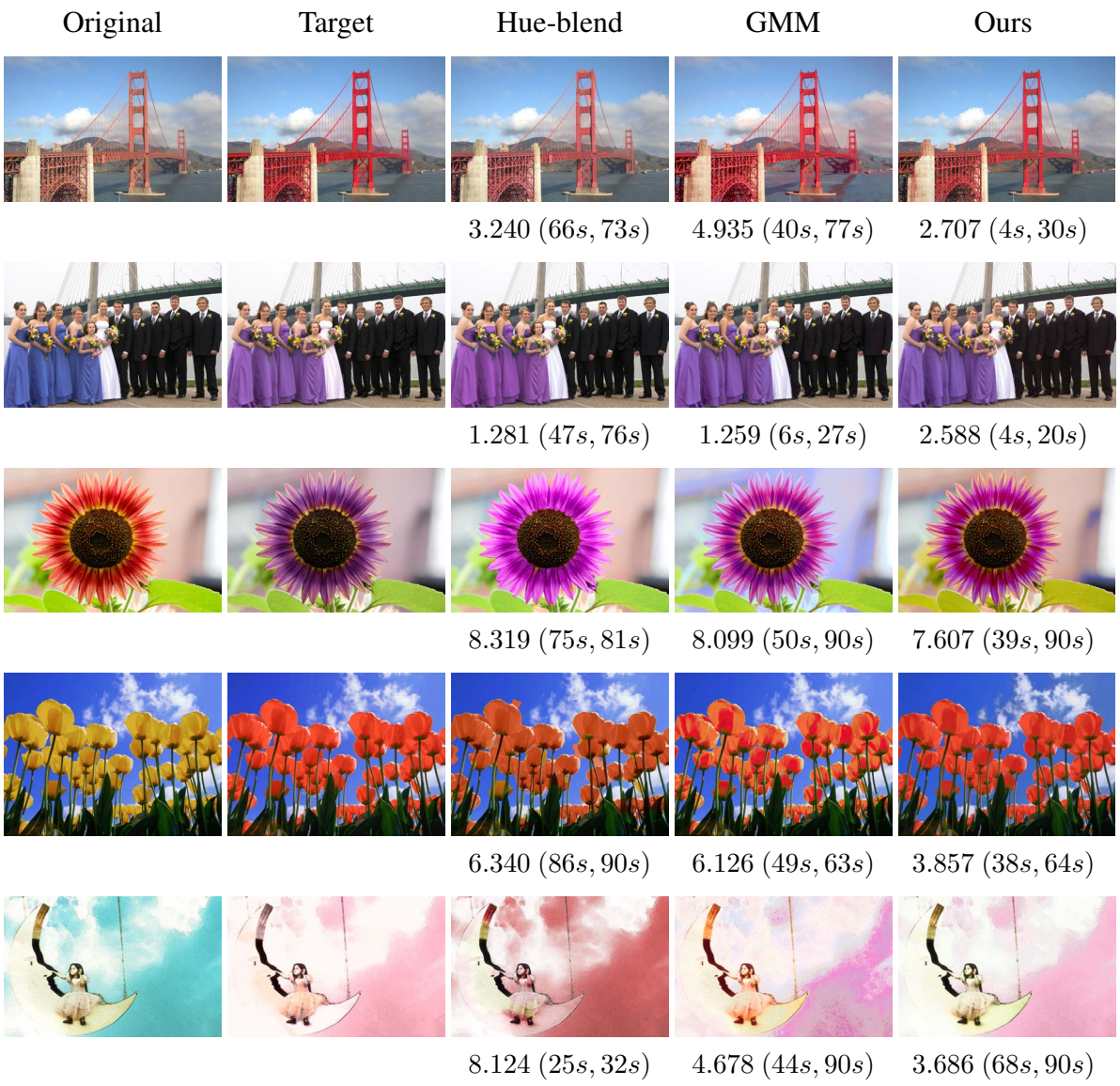


Figure 3.9: Examples from our user study. Left to right: original, target, results of hue-blend approach, GMM approach and our method. Under each result is the distance to target (RMS of CIEDE2000), and for each method this figure shows the best result out of all subjects according to this distance. Times in parentheses: number of seconds in which there was some user interaction, total number of seconds until task completion. Photos courtesy of the MIT-Adobe FiveK Dataset [2011] (top three), Chen et al. [20] (fourth), Wang et al. [113] (bottom).

3.4.2 User Study

Evaluating a task that has a component of personal taste is always challenging. Here we describe a user study showing that our method is easily learned by non-experts, sufficiently fast, expressive enough to produce a variety of imagery. In order to judge expressiveness,

the task we give our users is a matching task: they are given a target image (created from the original by manipulating its colors) and the user is asked to manipulate the original to match the target, using our method and two others. Thus we can judge whether our method can achieve a spectrum of desired results, and can use the other methods as baseline comparisons.

With the goal of capturing a broad spectrum of color editing operations, we collected 32 original-target pairs as follows. Half (16) were selected from eight papers in the literature that perform color manipulation [4, 19, 44, 71, 93, 104, 118, 113]. The other half were created by two expert Photoshop users (having the entire Photoshop tool-set at their disposal) according to a written task description like "Change the color of the bridge to be a richer red color." or "Brighten everything to make it look more like daytime."

For the study we used Amazon's Mechanical Turk framework. In each task (called a "HIT"), the subject was asked to edit four images (selected randomly without repetition per worker) from among the 32 original-target pairs. They were instructed to attempt to edit the original to match the target as best as they could within 90 seconds. Subjects who satisfied earlier were able to click a button to advance to the next image. Before beginning the task they were given a brief set of instructions that took a few minutes to complete.

Each worker was assigned randomly to one of three different conditions: our method, GMM, and hue-blend (described below). Most workers did just one task for us, but any workers that did more than one were always assigned the same condition. Our method was the approach described in Section 3.3, but with a fixed five-color palette automatically selected that could not be changed. GMM used the same GUI and instructions as our method, but the underlying color manipulation algorithm used a Gaussian mixture model both to select the palette and to manipulate it. Hue-blend was similar to a hue blend layer in Photoshop. In this interface the user "paints" hue into the image, which replaces the hue in the image but leaves saturation and luminance unchanged. Subjects were given a choice of three brush sizes which were either circular or a "smart brush" that reshapes according

	Ours	GMM	Hue Blend	Photoshop
Min	3	7	17	~ 60
Max	90*	90*	90*	~ 720
Median	71	69	90	~ 210

*our user study task was capped at 90 seconds per image.

Table 3.1: Task completion times. The table shows time (seconds) to recolor an image, for each of the methods in our user study. Last column is approximate time for expert Photoshop user to recolor the same images. Note that the user study task was capped at 90 seconds, and we observe that most of the hue blend mode users did not complete the task within that time frame. Using our full method and GMM takes roughly the same amount of time, while hue blend is slower and using Photoshop was the slowest.

to gradients in the image. (Specifically, we segment the image into superpixels [85] and paint any superpixel intersected by the circular brush.)

Via this study we collected 1820 images, of which 592, 608 and 620 were produced respectively by our method, GMM and hue-blend. The number of results for each target in each condition ranged from 15 to 25. Next we compare the three methods it terms of expressibility (were the workers able to reach the target?) and ease of use (were they able to do so within the 90 seconds?).

Figure 3.9 shows a few examples: the original image, the target image (either from related papers or created using Photoshop) and the best results for each method, measured by CIEDE2000 distance from the target [76]. In the figure (and in the study) our method produced the single best performing result by this measure more often than the other methods. Figure 3.10 aggregates all distance-from-target results. The median distance for our method is smaller than that of GMM ($p < 0.001$) and that of hue-blend ($p < 0.0001$). Statistical significance was found via a randomized permutation test with Bonferroni correction. Because the hue-blend mode did not allow subjects to adjust luminance (which is required for some of the targets) we also calculated the same distances but first using histogram equalization on the luminance channel of the result. Of course this changed all individual distances but the relative performances of the different methods (and the p-values) were comparable.

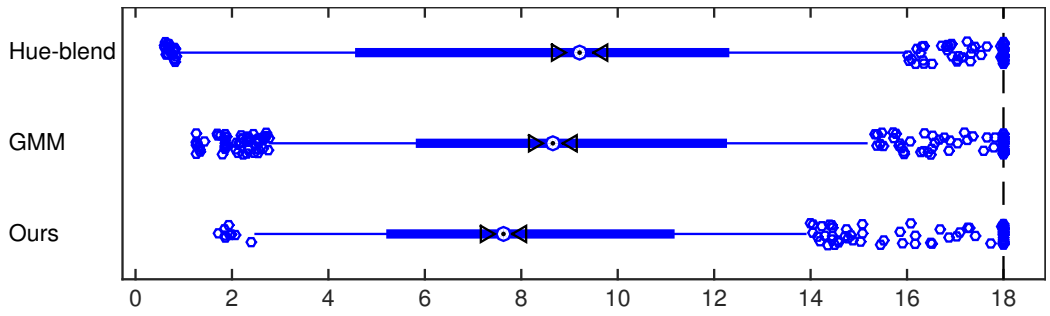


Figure 3.10: Distance comparison across methods. For all user study results, we calculate the CIEDE2000 distance between result image and target. The plot shows results for the 3 methods: hue-blend, GMM and ours (lower is better). Thick line indicates range between 25th and 75th percentiles. Dotted circle on the thick line indicates the median, which is lower (with statistical significance) for our method than for GMM and hue-blend.

Table 3.1 shows completion times over all subjects, for each of the three methods. It also shows the time it took an expert Photoshop user to produce the corresponding 16 target images. From the table it is clear that our GUI is the fastest (using either our algorithm or GMM) while using a brush to paint hues is slower. The Photoshop experts took the longest to edit images, although we note that the other three interfaces explicitly capped the amount of time available whereas the Photoshop experts had no time pressure.

Moreover, inspection of the pool of results leads us to believe that these distance measures were overly kind to the other methods. Qualitatively we observe that: (1) GMM results often contain highlights and halos that did not seem to hugely adverse effect their distance scores. (2) Hue-blend results often exhibit visual artifacts, as users had difficulty painting accurately. When these artifacts are small (yet noticeable) they do not adversely effect the distance commensurate with their visual impact. (3) Some of our results that have high distances actually look very similar to the target. The L-monotonicity constraint described in Section 3.3.4 allows the user to make subtle changes in luminance overall that induce high distance values even though they are not visually objectionable. While researchers have investigated other distance measures that better capture human perception (e.g., [117]) this remains an open research problem.

Finally we note that inasmuch as the results of our method did not exactly match those produced by the Photoshop experts (with the entire suite of color manipulation tools in that software and unlimited time to work on the images), it is not obvious that one or the other is "better."

3.5 Results

Figure 3.1 and 3.4 show examples of using our method with palettes of size 3, 4 and 5. They demonstrate both local and global changes (and a combination of the two).

Figure 3.11: Using a mask. Left-to-right: original, mask, result.

Our method supports masks, constraining the edits to specific image regions, as can be seen in Figure 3.11. Using a mask is necessary only if similarly colored objects should be edited in a different manner. Except for Figure 3.11, all results in this chapter do *not* use masks.

The remainder of this section describes various other applications made possible by our palette based approach.

3.5.1 Photo Collection Harmonization

Due to the proliferation of personal collections, it is desired to allow users to enhance the color consistency of multiple photos easily. HaCohen et al. [39] describe a method of consistently editing a whole collection of photos that share content. However, the

(a) original collection (b) automatic harmonization (c) editing result

Figure 3.12: Image collection editing. Our system can be applied to multiple images at once. The the joint palette of all input images – color theme serve as the target palette, thus achieving a consistent result across the collection for further color exploration. The entire editing session took less than 15 seconds.

approach requires the pair of input photos to share the same subject, thus are not suitable for harmonizing a general set of input images. An alternative way is to extend pairwise color transfer [96, 118, 121] to multiple images by building some type of composite image (e.g. average of the inputs) as the reference in order to enhance the consistency. Yet, it holds a strong assumption that the color statistic of every image in the input collection should match the *same* reference. It could lead to visual artifacts if input images maintain disparate color distributions. In contrast, we provide a more robust solution for harmonizing multiple images by adapting target palettes for different input images.

Given a collection of images, we first extract a color palette for each image separately. At the same time, we also extract a single *color theme* for the entire collection. For each palette color, we find its nearest color in the color theme and take it as the target palette color. We then employ the color transformation described above to achieve the harmonization of the color palette. Furthermore, the user can manipulate the color theme to meet his editing goal, thus editing operations will change all images in a consistent manner. Figure 3.12 shows an example of an edited photo collection. Notice that all photos were edited with less than 20 seconds of user input, and this number is constant regardless of the number of images we are editing. Moreover, because of the acceleration scheme

described in Section 3.3.6, the resulting output can be rendered at interactive frame rates even for collections containing many millions of pixels.

3.5.2 Video Recoloring

Our method is fast enough to be applied to a video, in real time. The interaction uses the same interface as for photos, shown in Figure 3.2. An example may be seen in the accompanying video. For this application, we need to choose a source palette for each frame. Our implementation selects a palette from one of the frames and uses it throughout the sequence. This approach encourages temporal coherence, but might not be suitable for long sequences. The selection method could be extended to a sliding window scheme, thus making is usable for longer, more heterogeneous sequences.

3.5.3 Duotone

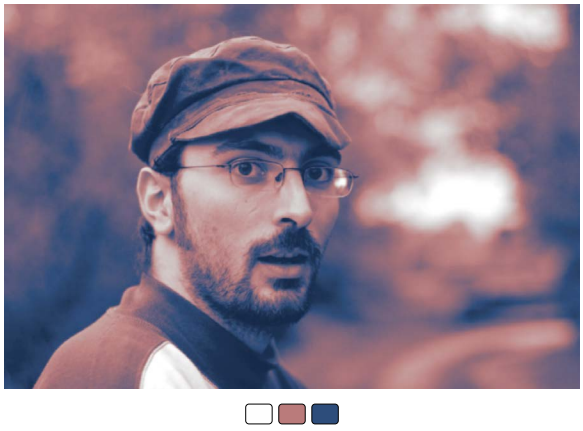


Figure 3.13: Duotone. Starting from a grayscale image, we use a three color palette (white and two other colors) to create this effect.

Duotone reproduction is a traditional printing technique that typically involves two colors of ink applied via halftone patterns over white paper. This style gives an overall sense of coloration and has a nostalgic quality (but costs less than full-color printing involving three or four inks). Digital imaging software like Photoshop provides a duotone function

to produce this effect, while allowing the user to choose the ink color(s). Our method can easily produce a similar effect as follows. We start with a grayscale image (or desaturate a color image). We select a 3-color palette and force one of the colors to be white (paper), letting the automatic algorithm of Section 3.3.2 choose the other two palette colors. Finally the user can adjust the two darker (ink) palette colors to achieve various imagery in the style of a duotone. Figure 3.13 shows an example.

3.5.4 Automatic Color Manipulation

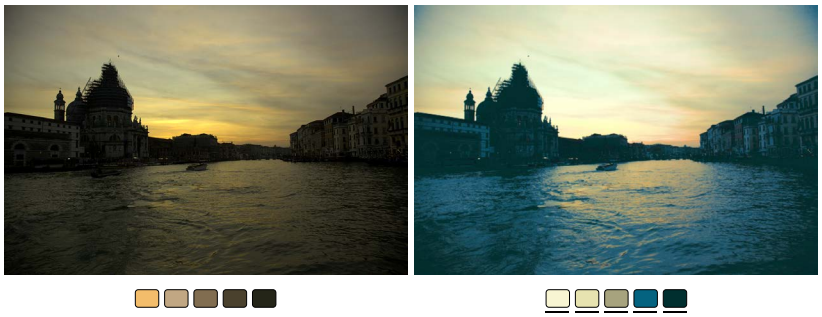


Figure 3.14: Fully automatic pipeline. As an example, we use the top rated palette from Adobe Color CC and apply it to an image. The input and output palettes are matched in increasing luminance order. Left: original, right: result of applying the "sandy stone beach ocean diver" palette.

Our methods was constructed with controllability in mind, allowing the user to select the resulting colors manually. While we believe that to be the most common use case, some users might prefer a fully automatic method. There are many on-line repositories of "good" color palettes, such as the Adobe Color CC database used by O'Donovan et al. [88]. We can automatically pick a top rated palette and apply it to an image.

For this application we need to enhance our method with an automatic way to match source and target palette colors. Using the annotation in Section 3.3.2, there we know that $C_1, C_2 ... C_k$ correspond to $C'_1, C'_2, ..., C'_k$ (e.g. C_1 corresponds to C'_1). This matching is given to us via our GUI (the user selects a palette color, then changes it) but is not available for the fully automatic method. Thus we augment our method with a palette matching step

that decides on the correct permutation of colors. As discussed in Section 3.3.4, we found that sorting colors according the luminance value is a good matching strategy. Figure 3.14 shows automatic image recoloring result using the top Adobe Color CC palettes.

3.5.5 Stroke-Based Interface

Finally, our palette based editing method can easily be augmented with a stroke-based interface that relies on the same algorithms described in Section 3. Using a mouse (or tablet or multitouch device) the user draws a stroke over the image as a way of indicating a "selection" in color space. This operation essentially specifies a palette and a selected palette color as follows. We take the mean color of the pixels under the stroke to be the selected palette color to be edited. The remainder of the palette is filled using the approach described in Section 3.2. Next as the user changes the color of the selected palette entry, the corresponding colors in the image are modified using the color transfer approach in Section 3.

As observed in Section 3.1, the size of the palette governs the locality in color space of various edits. We set the palette size:

$$k = \max(3, 7 - \lfloor \frac{7\sigma_s}{\sigma_I} \rfloor)$$

where σ_s and σ_I are the standard deviations of the pixel colors under the stroke and of the whole image. This gives a palette size between 3 and 7, depending on whether the colors under the stroke have high or low variance relative to those of the image. This approach offers the user a simple control of the locality, as illustrated in Figure 3.15-top. A small brush stroke in the blue of the sky (left) yields a palette size of 7 and thus the color edits are localized to blue colors; in contrast a large stroke that crosses the blue sky and gray clouds gives a palette of size 3 and thus affects a broader swath of colors in the result (right). Comparing this interface to that of Chen et al. [20] which requires brushes indicating both

Figure 3.15: Stroke-based editing. Top: Different brush marks select local (left) or global (right) color ranges for editing. Input images with red brush marks are left of diagonals while output is to the right with automatic palettes below. Bottom: Comparing to the approach of Chen et al. [20] (left), our method achieves similar effects with fewer marks (right).

colors to be modified (red stroke in the lower-left) and colors to be left untouched (black stroke) we find that we are able to produce similar effects with only the modifying strokes (lower-right).

Note that our current implementation uses strokes to select color ranges, regardless of location in the image. A more sophisticated approach might select for *both* color and location (in 5D rather than 3D). However, adding these dimensions to the lookup table described in Section 3.6 would have a performance impact.

Chapter 4

Portrait Stylization

4.1 Introduction

Digital photo manipulation now plays a central role in portraiture. Professional tools allow photographers to adjust lighting, remove blemishes, or move wisps of hair. Whimsical applications let novices add cartoon elements like a party hat or clown nose, or to turn photos into drawings and paintings. Some tools like Taaz [2] and PortraitPro [1] can digitally add makeup to a person in a photo, but the styles are limited to a collection of preset configurations and/or a set of parameters that adjust specific features like lip color.

This chapter introduces a way to digitally add makeup to a photo of a person, where the style of the makeup is provided in an example photo of a different person (Figure 4.1). One challenge is that it is difficult to acquire a dataset of photo triplets from which to learn: the source photo, the reference makeup photo, and the ground truth output (which preserves identity of the source and style of the reference). Previous work on style transfer avoids the need for such a training set by defining the style and content loss functions based on deep features trained by neural networks [35, 64, 70]. While those approaches can produce good results for stylization of imagery in general, they do not work well for adding various makeup styles to faces. A second challenge, specific to our makeup problem, is that people

Figure 4.1: Three source photos (top row) are each modified to match makeup styles in three reference photos (left column) to produce nine different outputs (3×3 lower right).

are highly sensitive to visual artifacts in rendered faces. A potential solution is to restrict the stylization range so as to define a specific color transformation space (such as affine transformations), or so as to preserve edges [70, 75, 64]. Unfortunately, this approach limits the range of makeup, because many styles include features that would violate the edge preservation property such as elongated eyelashes or dark eye liner.

Inspired by recent successful photorealistic style transfer based on generative adversarial networks (GANs), we take an unsupervised learning approach that builds on the CycleGAN architecture of Zhu et al. [130]. CycleGAN can transfer images between two domains by training on two sets of images, one from each domain. For our application, CycleGAN could in principle learn to apply a general make-you-look-good makeup to a no-makeup face, but it would not replicate a specific example makeup style.

Thus, we introduce a set of problems where the forward and backward functions are asymmetric. Another example application would be transferring patterns from an example shirt to a white shirt, where the paired backward function could remove patterns from a shirt to make it white. Such forward (style transfer) functions require a source image and reference style as input, whereas the backward function (style removal) only takes the stylized image as input.

Our approach relies on two asymmetric networks: one that transfers makeup style and another that removes makeup (each of which is jointly trained with an adversary). Application of both networks consecutively should preserve identity of the source photo (Figure 4.2). Finally, to encourage faithful reproduction of the reference makeup style, we train a style discriminator using positive examples that are created by warping the reference style face into the shape of the face in the source photo. This strategy addresses the aforementioned problem of a ground truth triplet dataset.

The principal contributions of this chapter are: (1) A feed-forward makeup transformation network that can quickly transfer the style from an arbitrary reference makeup photo to an arbitrary source photo. (2) An asymmetric makeup transfer framework wherein we train a makeup removal network jointly with the transfer network to preserve the identity, augmented by a style discriminator. (3) A new dataset of high quality before- and after-makeup images gathered from YouTube videos.

4.2 Related Work

Makeup Transfer and Removal. Makeup transfer is a specific form of style transfer that demands precise semantic understanding of the face to generate photorealistic details. Several previous work addressed the challenges of makeup transfer. Tong et al. [112] tackled the problem of automatic transfer of makeup from a makeup reference to a new portrait. Similar to image analogies [43], their framework requires as reference a pair of

well-aligned before-makeup and after-makeup photos of the same person. Guo et al. [35] proposed a method that requires only the after-makeup photo as reference. They decompose the reference and target portrait into three layers, face structure, skin detail and color, and transfer the makeup information for each layer separately. A major disadvantage of the approach is the need of a pre-processing step to warp the example makeup to the target face based on detected facial landmarks. Similarly, Li et al. [60] proposed to decompose the source portrait into albedo, diffuse and specular layers and transform each layer to match the optical properties of the corresponding layers of the reference. Different from previous work that transfer makeup from one reference, Khan et al. [55] introduced an approach to transfer local makeup styles of individual facial components from multiple makeup references. corresponding facial components in the target. Inspired by the recent success of neural-based style transfer techniques, Liu et al. [70] applied optimization-based neural style transfer models locally on facial components.

Other than makeup transfer, researchers have also attempted to digitally remove makeup from portraits [114, 21]. All previous work treat makeup transfer and removal as separate problems. We propose a single framework that can perform both tasks at the same time and we show that by alternating the improvement of transfer and removal processes, better results can be obtained for both tasks.

General Style Transfer. Researchers have investigated the general style transfer problems extensively. Gatys et al. [31] studied artistic style transfer. They proposed to combine the content of one image with the style of another by matching the Gram matrix statistics of deep features using optimization. Johnson et al. [48] later proposed a feed-forward network to approximate the solution to the optimization problem when the goal is to transfer a single style to arbitrary target images. The above methods are designed for painterly or artistic style transfer. When both the target and the style reference are photographs, the output exhibits artifacts reminiscent of a painterly distortion. In order to transfer photographic style, recent work [75] added semantic segmentation as an optional

guidance and imposed a photorealism constraint on the transformation function to avoid edge distortions in the result.

Style transfer can also be considered as image analogy with a weak supervision, as described in [64, 42]. Liao et al.assume the input image pairs have similar semantic structure and use PatchMatch to calculate the dense correspondences in deep feature space. Their approach works well for both artistic and photorealistic style transfer, but is computationally expensive. The follow-on work by He et al. [42] improves the formulation for color transfer tasks. Both approaches showed good makeup transfer results by treating it as a color transfer problem. However, we argue that makeup transfer is more than color transfer. Sophisticated eye makeups require the generation of new edges to imitate the example eye lashes. Lip makeup alters not only the color, but also the textural appearance, e.g. shininess, of the lip. We resort to feed-forward neural networks to learn these complicated transfer behaviors from data.

Image Domain Adaption. Style transfer can also be posed as a domain adaptation problem in the image space. Image domain adaptation methods learn a mapping function to transform images in a source domain to have similar appearance to images in a target domain. Taigman et al. [110] first formulated the problem and adopted generative adversarial network (GAN) [34] and variational autoencoder (VAE) [95] as the mapping function to enforce the transformed output to be similar to the source in feature space. Zhu et al. [130] introduced CycleGAN which uses generative network together with a cycle consistency loss to encourage the distribution of the mapped images to be indistinguishable from that of the real images in the target domain. Similarly, Liu et al. [69] employed GAN and VAE and proposed a shared latent space assumption which is shown to imply the cycle-consistency constraints in CycleGAN.

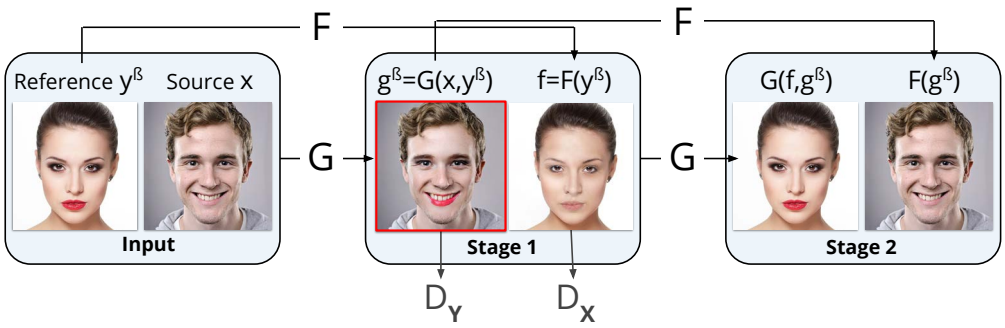


Figure 4.2: Network Pipeline. Given a source photo (without makeup) and a makeup reference, our system simultaneously learns a makeup transfer function G and a makeup removal function F. The result of the first stage can be viewed as a pair output by image analogy, and can serve as input for the second stage. We compare the output from the second stage with the source to measure identity preservation and style consistency.

4.3 Formulation

Due to the lack of pixel-aligned before-makeup and after-makeup image pairs, we tackle the makeup transfer problem using an unsupervised approach. Let X and Y be the no-makeup and with-makeup image domains where no pairings exist between the two domains. To characterize the no-makeup image domain X, we use a collection of no-makeup portrait photos of diverse facial features, expressions, skin tones, and genders, $\{x_i\}_{i=1,\ldots,N}, x_i \in X$. To model the with-makeup image domain Y, we use a diverse set of makeup examples $\{y_j\}_{j=1,\ldots,M}, y_j \in Y$, where the makeup styles range from nude and natural to artistic and intense. $Y^\beta \subset Y$ refers to a sub-domain of Y that contains images of a specific makeup style β. When $y_j \in Y^\beta$, we denote it as y_j^β.

As illustrated in Figure 4.2, our key idea is to simultaneously train two separate neural networks G and F, one to transfer specific makeup style and another to remove makeup. We hope by using diverse training examples, the network G can generalize its learning to the entire with-makeup image domain and can transfer arbitrary makeup styles to arbitrary faces at run time. Given a photo of a person with makeup, $y^\beta \in Y^\beta$, and a photo of a different person without makeup, $x \in X$, the makeup transfer network $G : X \times Y^\beta \rightarrow Y^\beta$ extracts the makeup layer from y^β and applies it to x maintaining its identity. Our result

Figure 4.3: Warping Guidance. We extract face landmarks and warp the reference face to source landmarks. The warping results exhibit not only the style but also pose, lighting, and some identity properties from the reference. The resulting face looks like a mix between the source and reference in terms of identity. We use the reference and its warping result as the positive examples of our discriminator L_S. In comparison, our results match the style of reference and the identity of source better.

$G(x, y^\beta)$, highlighted in Figure 4.2, should belong to the domain Y^β. Given the same photo y^β, the demakeup network $F : Y \rightarrow X$ learns to remove the makeup maintaining the identity of y^β. Note that G and F are unbalanced functions. G takes a pair of images as input to transfer the style from one to the other. F removes makeup given the with-makeup image itself. If G and F are successful, the output of G can be used as a makeup example to be transferred to the output of F, doubling the number of training examples. Also, if G and F operate without changing the facial features, transferring the makeup style from person 1 to 2 and then back to 1 should generate exactly the two input images. Assume x is sampled from X i.i.d according to some distribution $\mathcal{P}_X$ and y^β is sampled from Y i.i.d according to some distribution $\mathcal{P}_Y$. Based on makeup transfer properties, we adopt the following losses.

Adversarial loss for G. We first employ an adversarial loss to constrain the results of G to look similar to makeup faces from domain Y. The loss is defined as:

$$L_G(G, D_Y) = \mathbb{E}_{y \sim \mathcal{P}_Y}[log D_Y(y)] + \mathbb{E}_{x \sim \mathcal{P}_X, y \sim \mathcal{P}_Y}[log(1 - D_Y(G(x, y)))] \tag{4.1}$$

where the discriminator D_Y tries to discriminate between the real samples from domain Y and the generated samples $G(x, y^\beta)$, and the generator G aims to generate images that cannot be distinguished by the adversary D_Y.

Adversarial loss for F. We also apply an adversarial loss to encourage F to generate images indistinguishable from the no-makeup faces sampled from domain X.

$$L_F(F, D_X) = \mathbb{E}_{x \sim \mathcal{P}_X}[log D_X(x)] + \mathbb{E}_{y^\beta \sim \mathcal{P}_Y}[log(1 - D_X(F(y^\beta)))] \tag{4.2}$$

Identity loss. The adversarial loss constrains the output of G to look like a face with makeup applied. A trivial mapping function $G(x, y^\beta) = y^\beta$, i.e. G generates a result identical to y^β, will satisfy the above constraint, but is not a desirable solution due to the loss of identity of x. In fact, preserving the identity in the makeup transfer process is a challenging problem. Previous work [64] needed to apply additional post-processing step to recover the lost identity sacrificing the fidelity of the transfer. We propose to use the demakeup function F to explicitly encourage the preserving of identity in the makeup transfer process. The idea is that if we apply makeup to x and then immediately remove it, we should get back the input image x exactly. We use L1 loss to penalize the difference between $F(G(x, y^\beta))$ and x:

$$L_I(G, F) = \mathbb{E}_{x \sim \mathcal{P}_X, y^\beta \sim \mathcal{P}_Y}[||F(G(x, y^\beta)) - x||_1] \tag{4.3}$$

Style loss. The previous losses constrain the output of G to lie on the manifold of Y (faces with makeup) while maintaining the identity of X. However, they are not sufficient to ensure the successful transfer of details of a particular makeup style y^β. For this purpose, we propose two style losses, L1 reconstruction loss L_S and style discriminator loss L_P.

One key observation is that if we transfer the makeup style from face y^β to face x, and then use the result $G(x, y^\beta)$ to transfer the same makeup style back to the makeup-removed face $F(y^\beta)$, the result $G(F(y^\beta), G(x, y^\beta))$ should look exactly like the input face with makeup y^β:

$$L_S(G, F) = \mathbb{E}_{x \sim \mathcal{P}_X, y^\beta \sim \mathcal{P}_Y}[||G(F(y^\beta), G(x, y^\beta)) - y^\beta||_1] \tag{4.4}$$

Using L1 loss in the pixel domain helps the transfer of general structure and color (e.g. the shape of eyebrows and the gradient of eye-shadow), but leads to blurry results incapable of transferring sharp edges (e.g. eyelashes and eyeliners). Therefore, we add an auxiliary discriminator D_S, which decides whether a given pair of faces wear the same makeup. During training, we need to feed D_S with real makeup pairs (y^β, the same makeup style β applied to another face) and fake makeup pairs (y^β, $G(x, y^\beta)$). The challenge is that for the real makeup pairs, we do not have the *ground-truth* of applying makeup style β to another face, since all makeup styles appear only once in our training set. Therefore, we generate a *synthetic ground-truth* $W(x, y^\beta)$, by warping the reference makeup face y^β to match the detected facial landmarks in the source face x. Figure 4.3 show two example warping results. Subtle facial details important for preserving identity and expression are lost. For example, in the top row, the nose structure is changed and the laugh lines are completely removed. In the bottom row, the facial hair disappeared and the skin tone is different between the face and the neck regions. Though the warped images cannot serve as the final results, they can offer the discriminator a clue about what should be classified as positive examples of different faces wearing the same makeup. The loss for D_S is defined

as:

$$L_P(G, D_S) = \mathbb{E}_{x\sim\mathcal{P}_X, y^\beta\sim\mathcal{P}_Y}[logD_S(y^\beta, W(x, y^\beta))] + \mathbb{E}_{x\sim\mathcal{P}_X, y^\beta\sim\mathcal{P}_Y}[log(1 - D_S(y^\beta, G(x, y^\beta)))] \quad (4.5)$$

Total Loss. We optimize a min-max objective function $\min_{G,F} \max_{D_X, D_Y, D_S} L$, where the loss L is defined as:

$$L = \lambda_G L_G + \lambda_F L_F + L_I + L_S + \lambda_P L_P \quad (4.6)$$

λ_G, λ_F, and λ_P are the weights to balance the multiple objectives. The next section will provide more training details and discuss the appropriate weights.

4.4 Implementation

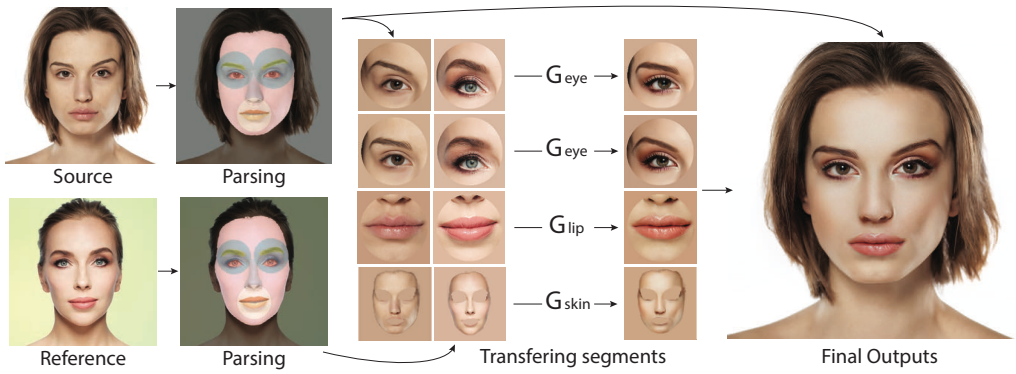


Figure 4.4: Generator per Segment. For each image, we apply face parsing algorithm to segment out each facial component. And we train three generators and discriminators separately for eyes, lip and skin considering the unique characteristics of each regions.

Training Pipeline. We aim to generate high-resolution makeup transfer results, but existing generative networks can only produce images smaller than 512×512 due to limited GPU memory. Motivated by the observation that the makeup of eye regions differs a lot from that of the skin or lip regions, we train three generators separately focusing the

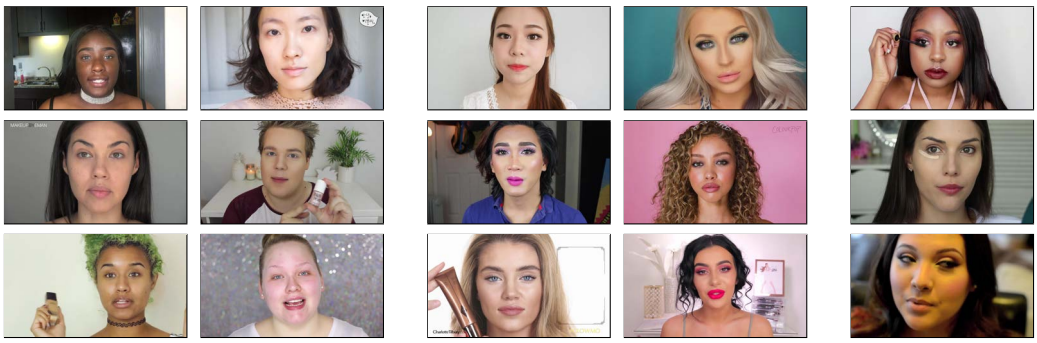


Figure 4.5: Dataset. We extract frames from Youtube makeup tutorials, filter out bad frames and cluster the remaining ones as no-makeup or with-makeup. We then conduct a user study that asks people whether it is a good no-makeup/makeup photo. The last column shows some examples people consider unqualified due to occlusion, incomplete makeup on face or bad facial expression.

network capacity and resolution on the unique characteristics of each individual regions. Given each pair of no-makeup and with-makeup images, we first apply face parsing algorithm to segment out each facial component, e.g. eyes, eyebrows, lip, nose, etc. (Figure 4.4). To best transfer eye makeup, we calculate a circle enclosing one eye, the corresponding eyebrow and the skin pixels in between and we train the generator on the entire eye region. Note that we flip the right eye regions horizontally so that we only need to train a single network for the left eye regions. We use circles for the eye and lip cutout due to the simplicity of applying random rotations as data augmentation. For skin and lip, we also perform horizontal flipping to double the amount of training data. As post-processing, we blend the generated pixels of the eye, lip and skin regions into the source using Poisson blending.

Data Collection. Existing face datasets are collected for face recognition and identification purposes and are of low image resolution (below 640x480). Faces without makeup and with makeup are often mixed together. For our unsupervised learning problem, we need two separate high-resolution datasets, one containing faces without any makeup and another one containing faces with a large variety of makeup styles. To this end, we collect our own datasets from Youtube makeup tutorial videos. For each video, we extract frames from the first quarter and the last quarter since they likely include no-makeup face and after-

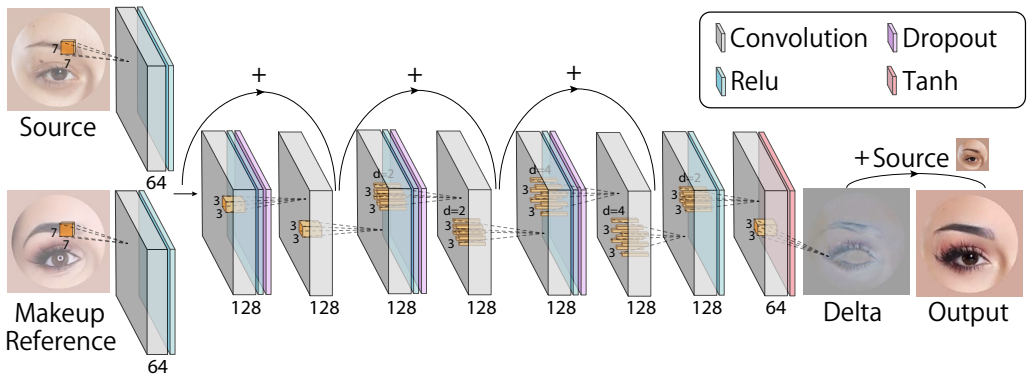


Figure 4.6: Generator Architecture. For generator G, we use DRN [123] with 3 dilated residual blocks to retain small spatial information (such as eye makeup). d indicates the dilation factor. We use two degridding layers (a 2-dilated 3×3 convolution and a 3×3 convolution) at the end of the network to avoid grid artifacts. The architecture of F is similar. The only difference is that it takes one image as input and therefore does not need concatenation.

makeup face. We discard the middle section of the video since the intermediate frames are most likely portraying the on-going makeup process. Among the extracted frames, we discard the blurry and duplicate ones and the ones containing face regions smaller than 400x400. We classify the remaining frames as either no-makeup or with-makeup using a heuristic algorithm which detects whether the eye regions are coated with non-skin colors or rich textures. After that, we ask the Amazon Mechanical Turk (MTurk) users to validate whether each frame is indeed a sharp no-makeup or with-makeup face with eyes open and without occlusions by fingers, brushes, etc. In this way, we harvest a no-makeup dataset of 1148 images and a with-makeup dataset of 1044 images. Figure 4.5 shows a few examples from our datasets and the kind of frames discarded by MTurk users. Our datasets contain a wide variety of facial identities and makeup styles.

Network Architecture. A reasonable architecture choice for the generators G and F is the traditional encoder-decoder network [101], which progressively downsamples the input image encoding it into a compact hidden code and then progressively upsamples the hidden code to reconstruct an image of the same resolution to the input. As discussed in pix2pix [45], a network architecture requiring the information to flow through a low-resolution bottleneck layer is not capable of generating sharp high-frequency details. To

circumvent the information loss, they added skip connections, following the general shape of a U-Net [101]. We also considered adopting spatial transformation network (STN) [47]. STN can generate new edges by transforming the reference makeup, but it suffers from the same problem as warping (Figure 4.3) that the identity of the source image is often lost due to the direct copy and paste of pixels from the reference. Instead of U-net and STN, we use Dilated ResNet (DRN) [123] architecture for the generators. Dilated convolution increases the receptive fields of the deeper network layers while maintaining the spatial resolution to avoid information loss. DRN utilizes the high-resolution feature maps to preserve image details. Its degridding layers can further improve the accuracy of spatial predictions. Our generator network contains 3 dilated residual blocks as shown in Figure 4.6. We also add 2 degridding layers at the end of the network to avoid artifacts in the generated results. Our generator G takes two images as input, as plotted in Figure 4.6, while our generator F only takes one with-makeup image as input and hence needs no concatenation.

Instead of directly generating the output pixels, the generator G calculates the delta image which can be added to the source image to obtain the final output, as illustrated in Figure 4.6. By doing that, we hope to maintain the original skin tone and lighting environment in the source and only transfer the makeup as an add-on layer. The skin on the face can be smoothed and sculpted by contours and highlights, but the general skin color should be similar to the neck for natural results. In this regard, we encourage the delta image to be sparse, and add a regularization term $L_R = ||G(x, y^\beta) - x||_1$ with weight 0.1 to our objective function. Our discriminators follow the design of the 256×256 discriminator in pix2pix [45]. Since faces contain distinctive global structures, we have the discriminator look at the whole image instead of image patches.

Training Details. We pretrain F using CycleGAN [130]. Makeup transfer is a one-to-many transformation and makeup removal is many-to-one. CycleGAN can remove most of the makeup from a face, but cannot transfer a specific makeup style to a new face. With F initialized by CycleGAN, we alternate the training of G and the fine-tuning of F. Since

Figure 4.7: Network Architecture and Loss Analysis. We compare our network using DRN architecture with losses described in Section 4.3, with models trained without specific loss terms and with the model trained using U-net architecture.

G is a much harder problem and F gets a good head start, we train G ten times more frequently than F. For the first 200 epochs, we set $\lambda_G = \lambda_F = \lambda_P = 0.1$ and after that, we trust the discriminators more and raise these values: $\lambda_G = \lambda_F = \lambda_P = 0.5$. The lip and face networks are trained for 400 epochs while eyes are trained for 850 epochs.

4.5 Results

In Figure 4.7, we first analyze whether each loss term is necessary by training a separate generator each time with one loss removed from our energy function. We keep all the hyper parameters the same as described before. When we remove L_G, the GAN loss for generator, G could apply the eye shadow anywhere around the eye on the source image since there is no discriminator distinguishing whether it is realistic or not. When we remove L_I, the generator encodes the characteristics of eyes in both source and reference resulting in identity loss. Without L_S, the results look less saturated and the makeup style is not fully transferred. Without L_P, the color in the result become more vivid, but some eye lashes get neglected. We also tried the U-net architecture with 9 blocks as described in the work[45]. But for our problem, DRN performs better than U-net architecture.

Figure 4.8 shows results on the lip and eye regions. Our network can faithfully reproduce the makeup material properties in the reference. In the top two rows, the generated lips not only exhibit plausible colors, but also inherit the shiny appearance from

Figure 4.8: Results of the lip and eye regions.

the reference. Our network can also synthesize high frequency eye lashes reasonably well. We would like to point out that we do not perform any pre-alignment on the facial features. With random rotation as data augmentation, our network is able to learn rotation, scaling and other transformations inherently and put makeup components in the right places. Notice that when the distance between eyes and eyebrows or the orientation of eyebrows are very different between the source and the reference, we can still synthesize plausible results (bottom two rows).

Figure 4.9: Results on the entire face.

Figure 4.9 also displays several examples of the combined results. Our network can transfer a variety of makeup styles across people of different skin tones preserving the original skin tone and other important identity details in the source.

Figure 4.10: Limitations. Extreme makeup styles (dark wings, face sparkles) unseen during training are difficult to reproduce.

However, one limitation of our approach is that the network does not work as well on extreme makeup styles unseen during training. As shown in Figure 4.10, the eye makeup is very different from the majority of the training examples. Our synthesized eyes look plausible but lack the precise pointy wing from the reference. The reference makeup contains shiny sparkles on the face which is unseen during training and is therefore not transferred in the results.

Quantitative Comparison

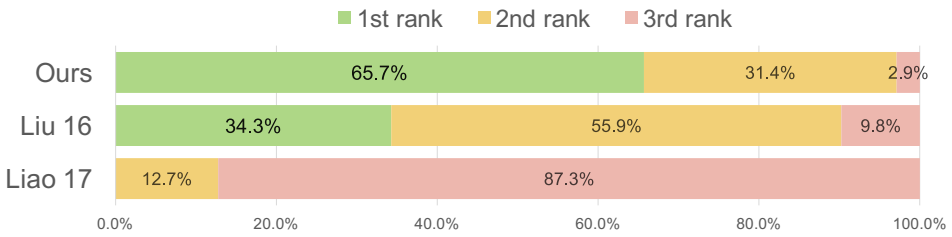


Figure 4.11: In paired comparison, how well do various methods perform, as judged by subjects on MTurk?

We conducted a user study on Amazon Mechanical Turk making a pairwise comparison among results of the method of Liao et al. [64], of Liu et al. [70], and of our method. We randomly select 102 source photos, and assign 3 of them to each of 34 style images, so we have 102 groups of source-style inputs. We then pick a pair of results from the same

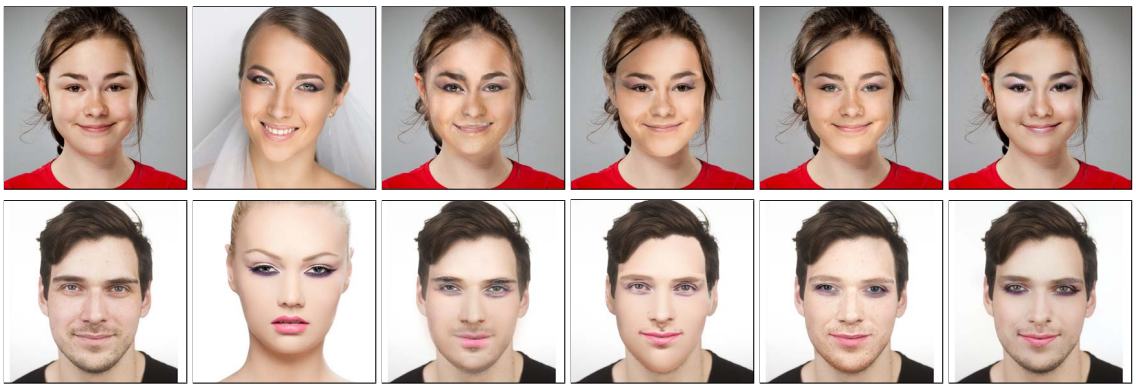


Figure 4.12: Makeup Transfer Results. We compare with makeup and portrait style transfer work [70, 30, 64].

group to compare, and we ask 10 or more subjects to select which result better matches the makeup style in the reference. On average 58.2% of people prefer our results over those of Liu et al., and 78.9% of people prefer ours over those of Liao et al.. Note that sometimes the results of Liu et al. may not look realistic, but this study focuses only on style similarity. Given the result images in the same source-makeup group, we employed the Bradley Terry model [8] to derive a quantitative performance score. Figure 4.11 shows ranking statistics – a count of how often a method had each rank. Our method outperforms all others by this measure.

Makeup Transfer Comparison.

In Figure 4.12, we compare our results with three different previous work, our implementation of *makeup like a super star* [70], *example-based portrait stylization* [30] and *deep image analogy* [64]. We cannot compare with the work by Tong et al. [112] or Khan et al. [55], because they do not solve the same problem as ours and require more inputs, e.g. before-makeup and after-makeup pair or multiple makeup references. For *makeup like a super star* [70], accurate face segmentation is crucial. We manually refined our face parsing result and included an additional eye shadow label. Their main problem is that different head poses and lighting conditions may lead to asymmetric eye makeup in the results. The

portrait stylization work [30] focuses on transferring artistic and painterly style, and sometimes distorts facial features that are only visible when transferring photographic style. We apply their method in the face region only and alpha composite the result onto the rest of the image. Similarly, we apply *deep image analogy* [64] in the face region. It finds dense correspondences in feature space between the source and reference. When the expressions differ (mouth open versus mouth closed), or the makeup style is dramatic (bottom row), the correspondences cease to exist and hence the analogy results are not as good. In favor of more realistic results and less distortions, they adopt a post-processing refinement step for photo-to-photo analogy, which transfers colors from the reference makeup and retains the structures in the source. The refinement step helps to preserve the identity but harms the system's capability to introduce high frequency details. Previous work also tend to alter the skin tone of the source with the skin tone in the reference resulting in identity loss that deviates from the goal of most professional makeup artists. For example, the results by Liao et al.contain unnatural skin tone transition from the neck to the face. In contrast, our network takes as input makeup reference of arbitrary head pose and facial expression, and is capable of properly transferring makeup styles, from natural to dramatic, preserving the source identity. Our network also maintains the input skin tone as much as possible by distilling only the makeup layer from the reference for transfer. We include more qualitative and quantitative results in the supplementary material.

Makeup Removal Comparison.

Restoring the natural look behind makeup is an ill-posed problem. There could be many plausible faces behind the same makeup, for example, the blemishes could be covered by the foundation and a natural pink lip could be painted red. Our generator tries to offer a plausible prediction of one's natural face given the makeup face as input. It may be beneficial for face verification systems. We compare our makeup removal results with "face behind makeup" [114] in Figure 4.13. Both methods produce natural-looking before-

Figure 4.13: Demakup Results. We compare with makeup removal work by Wang et al. [114]. Our demakeup network F can remove the detected makeup to virtually recover the original face.

makeup face. But our network removes makeup more aggressively and generates sharper results. The results by Wang et al. [114] retain partial eye-shadows, eyeliners and eyebrow colors. On the contrary, we remove them all and recover natural under-eye bags and suggest tentative appearances for the original eyebrows and lips. Our network also better preserves the original skin tone while removing highlights, contours and blushes. One limitation is that we cannot reconstruct realistic blemishes since most of the no-makeup faces in our training set contain clear and smooth skin.

Chapter 5

Conclusion

Photographs have become ubiquitous and a natural part of everyday life. Enhancing photos to be high-quality and eye-catching is not inaccessible in the age of camera-equipped smartphones. In this thesis, we discuss three systems that collectively build a foundation for solving the practical problem of enhancing personal photo collections. First, we identify the problem of triaging photos taken of the same scene, and seek a relative quality measure within each photo series by gathering dataset and experimenting several machine learning methods. Note that our current stage is still far from addressing the problem of learning human preference between photos compared. However, we are optimistic that in the foreseeable future, more advanced solutions with better semantic parsing can be developed to achieve human performance on this task. Second, we introduce a simple, intuitive and interactive tool that allows non-experts to recolor an image by editing a color palette. Besides, we propose to apply it on harmonizing multiple images, which can enhance the overall consistency of a photo collection. In fact, the realm of possibilities for color enhancement is far from explored, we believe incorporating more high-level notion, such as object detection or color names would enhance the intuitiveness of the system. An ultimate interface will behave "as the user expects," when users, for example, ask it to turn the black hat to red. Third, we present an unsupervised learning approach for transferring arbitrary

makeup styles to arbitrary source faces and for removing makeup, both at interactive rates. We believe this novel unsupervised learning framework can also be used in other domains ranging from similar applications like automatic aging and de-aging, to further afield, like photorealistic object style transfer.

Next, we identify several directions for future work in photo enhancement:

Multi-Photo Enhancement Professional photographers spend a long time enhancing a single image from multiple sources. A typical user however has neither the time nor the skills to perform this task. Interactive Photomontage [3] expedites this process by allowing the user to interactively create a composite by painting with high-level goals such as relighting, extended depth of field, and stroboscopic visualization of movement. This method requires the photos to be taken from a single vantage point, like on a tripod. I hope to relax this condition and work from more disparate photos – a group of pictures taken in a common setting with a hand-held camera. Moreover, I hope to leverage the preference model described in the automatic photo triage project and harness the technique of semantic segmentation (e.g. based on the COCO dataset [67]) to find which parts of different photos would make a pleasing composition.

AutoEnhancement Even though modern cellphones are equipped with insanely powerful cameras, there is still a gap between consumer and professional photos. Bychkovsky et al. [12] learned an automatic model for tone adjustment in the luminance channel based on a large retouching dataset collected from professional photographers. Existing consumer-level tools provide recoloring filters designed by experts, but few of them focus on more manipulations such as cropping, relighting and subject-oriented modifications. I hope to collect a set of retouching operations by experts so that we can learn to score each step of modification, and employ the model as an "oracle" for automatic photo enhancement.

Personalized Filters Each person has different visual taste, but most photo apps (as of early 2018) provide the same set of filters for everybody. However, offering more

options for filters is technically easy. But people's attention span and patience are limited – they usually only try out the first several filters. To address this problem, I would like to delve into modeling people's personalized preferences for editing the photos. For example, applications like Instagram may have thousands of filters for making photos and videos more attractive and interesting, but how will people select an appropriate filter? Understanding people's preferences from what they have already edited may provide the key to handling this problem.

Bibliography

[1] Portrait pro - easy photo editing software. http://www.portraitprofessional.com/.

[2] Taaz virtual makeover and hairstyles. http://www.taaz.com/.

[3] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Trans. Graph.*, 23(3):294–302, August 2004.

[4] Xiaobo An and Fabio Pellacini. Appprop: All-pairs appearance-space edit propagation. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 40:1–40:9. ACM, 2008.

[5] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.

[6] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98, 2015.

[7] Subhabrata Bhattacharya, Rahul Sukthankar, and Mubarak Shah. A framework for photo-quality assessment and enhancement based on visual aesthetics. In *Proceedings of the international conference on Multimedia*, pages 271–280. ACM, 2010.

[8] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[11] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[12] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[13] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 97–104. IEEE, 2011.

[14] Xudong Cao, Yichen Wei, Fang Wen, and Jian Sun. Face alignment by explicit shape regression. *International Journal of Computer Vision*, 107(2):177–190, 2014.

[15] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 34(4), July 2015.

[16] Huiwen Chang, Cynthia Lu, Fisher Yu, and Adam Finkelstein. Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] Huiwen Chang, Fisher Yu, Jue Wang, Douglas Ashley, and Adam Finkelstein. Automatic photo triage. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 35(4), July 2016.

[18] Youngha Chang, Suguru Saito, Keiji Uchikawa, and Masayuki Nakajima. Example-based color stylization of images. *ACM Transactions on Applied Perception*, 2(3):322345, July 2005.

[19] Xiaowu Chen, Dongqing Zou, Jianwei Li, Xiaochun Cao, Qinping Zhao, and Hao Zhang. Sparse dictionary learning for edit propagation of high-resolution images. Computer Vision and Pattern Recognition (CVPR), June 2014.

[20] Xiaowu Chen, Dongqing Zou, Qinping Zhao, and Ping Tan. Manifold preserving edit propagation. *ACM Trans. Graph.*, 31(6):132:1–132:7, Nov 2012.

[21] Ying-Cong Chen, Xiaoyong Shen, and Jiaya Jia. Makeup-go: Blind reversion of portrait edit. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[22] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[23] Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. Color harmonization. Association for Computing Machinery, Inc., July 2006.

[24] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. In *Computer Vision ECCV 98*, pages 484–498. Springer, 1998.

[25] Gabriela Csurka, Sandra Skaff, Luca Marchesotti, and Craig Saunders. Learning moods and emotions from color combinations. In *Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing*, pages 298–305. ACM, 2010.

[26] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Studying aesthetics in photographic images using a computational approach. In *Computer Vision–ECCV 2006*, pages 288–301. Springer, 2006.

[27] Sagnik Dhar, Vicente Ordonez, and Tamara L Berg. High level describable attributes for predicting aesthetics and interestingness. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1657–1664. IEEE, 2011.

[28] Steven Drucker, Curtis Wong, Asta Roseway, Steve Glenner, and Steve De Mar. Photo-triage: Rapidly annotating your digital photographs. Technical report, Microsoft Research Technical Report, MSR-TR-2003-99, 2003.

[29] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 249–256. ACM, 2002.

[30] Jakub Fišer, Ondřej Jamriška, David Simons, Eli Shechtman, Jingwan Lu, Paul Asente, Michal Lukáč, and Daniel Sỳkora. Example-based synthesis of stylized facial animations. *ACM Transactions on Graphics (TOG)*, 36(4):155, 2017.

[31] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[32] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[35] Dong Guo and Terence Sim. Digital face makeup by example. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 73–79. IEEE, 2009.

[36] YW Guo, M Liu, TT Gu, and WP Wang. Improving photo composition elegantly: Considering image similarity during composition optimization. In *Computer Graphics Forum*, pages 2193–2202. Wiley Online Library, 2012.

[37] Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, 30(4):70:1–70:9, 2011.

[38] Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. Nrdc: Non-rigid dense correspondence with applications for image enhancement. ACM SIGGRAPH 2011 papers, Article No. 70, 2011.

[39] Yoav HaCohen, Eli Shechtman, Dan B. Goldman, and Dani Lischinski. Optimizing color consistency in photo collections. *ACM Trans. Graph.*, 32(4):38:1–38:10, July 2013.

[40] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. *arXiv preprint arXiv:1411.5752*, 2014.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[42] Mingming He, Jing Liao, Lu Yuan, and Pedro V. Sander. Neural color transfer between images. *CoRR*, abs/1710.00756, 2017.

[43] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[44] Xiaodi Hou and Liqing Zhang. Color conceptualization. In *Proceedings of the 15th International Conference on Multimedia*, MULTIMEDIA '07, pages 265–268. ACM, 2007.

[45] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[46] David E Jacobs, Dan B Goldman, and Eli Shechtman. Cosaliency: Where people look when comparing images. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 219–228. ACM, 2010.

[47] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

[48] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[49] Neel Joshi, Wojciech Matusik, Edward H Adelson, and David J Kriegman. Personal photo enhancement using example images. *ACM Trans. Graph.*, 29(2):12–1, 2010.

[50] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2009.

[51] Tapas Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, Jul 2002.

[52] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. Recognizing image style. *arXiv preprint arXiv:1311.3715*, 2013.

[53] Liad Kaufman, Dani Lischinski, and Michael Werman. Content-aware automatic photo enhancement. In *Computer Graphics Forum*, pages 2528–2540. Wiley Online Library, 2012.

[54] Yan Ke, Xiaoou Tang, and Feng Jing. The design of high-level features for photo quality assessment. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 419–426. IEEE, 2006.

[55] Asad Khan, Yudong Guo, and Ligang Liu. Digital makeup from internet images. *CoRR*, abs/1610.04861, 2016.

[56] Aditya Khosla, Akhil S. Raju, Antonio Torralba, and Aude Oliva. Understanding and predicting image memorability at a large scale. In *International Conference on Computer Vision (ICCV)*, 2015.

[57] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[59] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 689–694. ACM, 2004.

[60] Chen Li, Kun Zhou, and Stephen Lin. Simulating makeup through physics-based manipulation of intrinsic image layers. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4621–4629, 2015.

[61] Congcong Li and Tsuhan Chen. Aesthetic visual quality assessment of paintings. *Selected Topics in Signal Processing, IEEE Journal of*, 3(2):236–252, 2009.

[62] Yong Li, Tao Ju, and Shi-Min Hu. Instant propagation of sparse edits on images and videos. In *Computer Graphics Forum*, volume 29, pages 2049–2054. Wiley Online Library, 2010.

[63] Yuanzhen Li, Edward Adelson, and Aseem Agarwala. Scribbleboost: Adding classification to edge-aware interpolation of local image and video adjustments. In *Computer Graphics Forum*, volume 27, pages 1255–1264. Wiley Online Library, 2008.

[64] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017.

[65] Sharon Lin and Pat Hanrahan. Modeling how people extract color themes from images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, 2013.

[66] Sharon Lin, Daniel Ritchie, Matthew Fisher, and Pat Hanrahan. Probabilistic color-by-numbers: Suggesting pattern colorizations using factor graphs. volume 32, pages 37:1–37:12, July 2013.

[67] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[68] Ligang Liu, Renjie Chen, Lior Wolf, and Daniel Cohen-Or. Optimizing photo composition. *Computer Graphic Forum (Proceedings of Eurographics)*, 29(2):469–478, 2010.

[69] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *arXiv preprint arXiv:1703.00848*, 2017.

[70] Si Liu, Xinyu Ou, Ruihe Qian, Wei Wang, and Xiaochun Cao. Makeup like a superstar: Deep localized makeup transfer network. *arXiv preprint arXiv:1604.07102*, 2016.

[71] Yiming Liu, Michael Cohen, Matt Uyttendaele, and Szymon Rusinkiewicz. Autostyle: Automatic style transfer from image collections to users images. *Computer Graphics Forum*, 33(4):21–31, 2014.

[72] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014.

[73] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[74] Xin Lu, Zhe Lin, Hailin Jin, Jianchao Yang, and James Z Wang. Rapid: Rating pictorial aesthetics using deep learning. In *Proceedings of the ACM International Conference on Multimedia*, pages 457–466. ACM, 2014.

[75] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *arXiv preprint arXiv:1703.07511*, 2017.

[76] M Ronnier Luo, Guihua Cui, and B Rigg. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application*, 26(5):340–350, 2001.

[77] Wei Luo, Xiaogang Wang, and Xiaoou Tang. Content-based photo quality assessment. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2206–2213. IEEE, 2011.

[78] Yiwen Luo and Xiaoou Tang. Photo and video quality evaluation: Focusing on the subject. In *Computer Vision–ECCV 2008*, pages 386–399. Springer, 2008.

[79] Yu-Fei Ma, Lie Lu, Hong-Jiang Zhang, and Mingjing Li. A user attention model for video summarization. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 533–542. ACM, 2002.

[80] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2272–2279. IEEE, 2009.

[81] Luca Marchesotti, Florent Perronnin, Diane Larlus, and Gabriela Csurka. Assessing the aesthetic quality of photographs using generic image descriptors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1784–1791. IEEE, 2011.

[82] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 389–400, 1997.

[83] Megvii Inc. Face++ research toolkit. `www.faceplusplus.com`, 2013.

[84] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[85] G. Mori. Guiding model search using segmentation. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1417–1423 Vol. 2, Oct 2005.

[86] Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2408–2415. IEEE, 2012.

[87] Masashi Nishiyama, Takahiro Okabe, Imari Sato, and Yoichi Sato. Aesthetic quality classification of photographs based on color harmony. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 33–40. IEEE, 2011.

[88] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Color Compatibility From Large Datasets. *ACM Transactions on Graphics*, 30(4), 2011.

[89] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[90] Christopher C. Paige and Michael A. Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, 1982.

[91] Jaesik Park, Joon-Young Lee, Yu-Wing Tai, and In So Kweon. Modeling photo composition and its application to photo re-arrangement. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 2741–2744. IEEE, 2012.

[92] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[93] Francois Pitie, Anil C Kokaram, and Rozenn Dahyot. N-dimensional probability density function transfer and its application to color transfer. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1434–1439. IEEE, 2005.

[94] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Transactions on Graphics (SIGGRAPH 2006 issue)*, 25(3):1214–1220, July 2006.

[95] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[96] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):3441, September 2001.

[97] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM transactions on graphics (TOG)*, 21(3):267–276, 2002.

[98] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[99] Shaoqing Ren, Kaiming He, Ross B. Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *CoRR*, abs/1504.06066, 2015.

[100] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 10–17. IEEE, 2003.

[101] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[102] Adrian Secord. Weighted voronoi stippling. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 37–43. ACM, 2002.

[103] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. In *Acm transactions on graphics (tog)*, volume 27, page 73. ACM, 2008.

[104] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Image appearance exploration by model-based navigation. In *Computer Graphics Forum*, volume 28, pages 629–638, 2009.

[105] Ian Simon, Noah Snavely, and Steven M. Seitz. Scene summarization for online image collections. In *ICCV*. IEEE, 2007.

[106] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[107] Pinaki Sinha, Sharad Mehrotra, and Ramesh Jain. Summarization of personal photologs using multidimensional content and context. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 4. ACM, 2011.

[108] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[109] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.

[110] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[111] Huixuan Tang, Neel Joshi, and Ashish Kapoor. Learning a blind measure of perceptual image quality. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 305–312. IEEE, 2011.

[112] Wai-Shun Tong, Chi-Keung Tang, Michael S Brown, and Ying-Qing Xu. Example-based cosmetic transfer. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pages 211–218. IEEE, 2007.

[113] Baoyuan Wang, Yizhou Yu, Tien-Tsin Wong, Chun Chen, and Ying-Qing Xu. Data-driven image color theme enhancement. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, pages 146:1–146:10. ACM, 2010.

[114] Shuyang Wang and Yun Fu. Face behind makeup. *AAAI Conference on Artificial Intelligence*, 2016.

[115] Xiaohui Wang, Jia Jia, and Lianhong Cai. Affective image adjustment with a single word. *Vis. Comput.*, 29(11):1121–1133, November 2013.

[116] Xin-Jing Wang, Lei Zhang, and Ce Liu. Duplicate discovery on 2 billion internet images. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 429–436. IEEE, 2013.

[117] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[118] Yu wing Tai, Jiaya Jia, and Chi keung Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *Proc. Computer Vision and Pattern Recognition*, pages 747–754, 2005.

[119] Kun Xu, Yong Li, Tao Ju, Shi-Min Hu, and Tian-Qiang Liu. Efficient affinity-based edit propagation using k-d tree. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 118:1–118:6. ACM, 2009.

[120] Peng Ye, Jayant Kumar, Le Kang, and David Doermann. Unsupervised feature learning framework for no-reference image quality assessment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1098–1105. IEEE, 2012.

[121] Jae-Doug Yoo, Min-Ki Park, Ji-Ho Cho, and Kwan H. Lee. Local color transfer between images using dominant colors. *J. Electron. Imaging*, 22(3), July 2013.

[122] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[123] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[124] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[125] Lu Yuan and Jian Sun. Automatic exposure correction of consumer photographs. In *Computer Vision–ECCV 2012*, pages 771–785. Springer, 2012.

[126] Chenxi Zhang, Jizhou Gao, Oliver Wang, Pierre Georgel, Ruigang Yang, James Davis, Jan-Michael Frahm, and Marc Pollefeys. Personal photograph enhancement using internet photo collections. *IEEE transactions on visualization and computer graphics*, 20(2):262–275, 2014.

[127] Luming Zhang, Mingli Song, Qi Zhao, Xiao Liu, Jiajun Bu, and Chun Chen. Probabilistic graphlet transfer for photo cropping. *Image Processing, IEEE Transactions on*, 22(2):802–815, 2013.

[128] Erjin Zhou, Haoqiang Fan, Zhimin Cao, Yuning Jiang, and Qi Yin. Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 386–391, 2013.

[129] Jun-Yan Zhu, Aseem Agarwala, Alexei A Efros, Eli Shechtman, and Jue Wang. Mirror mirror: Crowdsourcing better portraits. *ACM Transactions on Graphics (TOG)*, 33(6):234, 2014.

[130] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.