

# A Framework for Content-Adaptive Photo Manipulation Macros: Application to Face, Landscape, and Global Manipulations

FLORAINE BERTHOUSOZ

University of California, Berkeley

and

WILMOT LI and MIRA DONTCHEVA

Adobe Systems

and

MANEESH AGRAWALA

University of California, Berkeley

We present a framework for generating content-adaptive macros that can transfer complex photo manipulations to new target images. We demonstrate applications of our framework to face, landscape and global manipulations. To create a content-adaptive macro, we make use of multiple training demonstrations. Specifically, we use automated image labeling and machine learning techniques to learn the dependencies between image features and the parameters of each selection, brush stroke and image processing operation in the macro. Although our approach is limited to learning manipulations where there is a direct dependency between image features and operation parameters, we show that our framework is able to learn a large class of the most commonly-used manipulations using as few as 20 training demonstrations. Our framework also provides interactive controls to help macro authors and users generate training demonstrations and correct errors due to incorrect labeling or poor parameter estimation. We ask viewers to compare images generated using our content-adaptive macros with and without corrections to manually generated ground-truth images and find that they consistently rate both our automatic and corrected results as close in appearance to the ground-truth. We also evaluate the utility of our proposed macro generation workflow via a small informal lab study with professional photographers. The study suggests that our workflow is effective and practical in the context of real-world photo editing.

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Applications—*Photo Editing*

General Terms: Algorithms

Additional Key Words and Phrases: Human Computer Interaction, Photo Editing, Macros, Programming-By-Demonstration

## ACM Reference Format:

Berthousoz, F., Li, W., and Dontcheva, M., Agrawala, M. 2011. A Framework for Content-Adaptive Photo Manipulation Macros: Application to Face, Landscape, and Global Manipulations. *ACM Trans. Graph.* XX, Y, Article XXX (June 2011), 13 pages.

DOI = 10.1145/XXXXXXX.YYYYYYY

<http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY>

## 1. INTRODUCTION

Photographers often use image manipulation software such as Adobe Photoshop and the GNU Image Manipulation Program (GIMP) to improve the quality of their raw images – they adjust contrast, correct colors, sharpen foreground objects, add artistic effects, etc. Many photographers develop their own sequences of ad-

justments that they repeatedly apply to hundreds of images. For example, a photographer might routinely apply the same set of operations to enhance the orange hues of sunsets, create a lomography-style vignetting effect, or improve skin-tones. Such manipulations have become very popular and thousands of photographers publish photo manipulation tutorials online to share them with other users.

To facilitate repetitive procedures, Photoshop and GIMP provide basic macro authoring tools that allow users to record and then replay a sequence of operations. Yet, the macros authored today are extremely brittle; they are limited to executing exactly the same operations in the recording and cannot adapt to new target images. Thus they are inappropriate for many common photo manipulations. For example, a macro designed to correct skin-tone for one image will likely fail for new images simply because the skin is in a different location. In addition, the parameter values of the skin-tone color adjustment operation depend on both the color of the skin and the overall color cast of the target image. The parameters used to correct the skin-tone of a dark-skinned person in a photograph with a blue cast ruin the skin-tone of a light-skinned person in a photograph with an orange cast (Figure 1 left).

Image manipulations are typically composed of three types of low-level operations; 1) selecting a region, 2) drawing brush strokes and 3) applying an image processing operation within the selected region or along the brush stroke. To properly apply manipulations like skin-tone correction, users must adapt the *location* of the selection region, the *paths* of the brush strokes, and the *parameters* of the image processing operations to the content of the underlying image. Most photo manipulation tools cannot automatically adapt such content-dependent operations to new target images.

In previous work [Grabler et al. 2009], we took a first step towards producing content-adaptive macros by using automated image labeling to transfer recognized selection regions (e.g., lips, eyebrows, sky) from one image to another. However, this approach cannot transfer partially selected regions, it does not handle brush strokes, and it cannot adapt image processing parameters to the target image. In the skin-tone example, this approach would identify the skin regions in the target image, but it would then simply copy the color adjustment parameters from the example demonstration and therefore fail to properly correct the target (Figure 1d).

In this paper, we present a more comprehensive approach for generating content-adaptive macros that can automatically transfer selection regions, brush strokes and operation parameters to new target images. Our approach is a framework in the sense that we can apply these transfer mechanisms independently of the class of

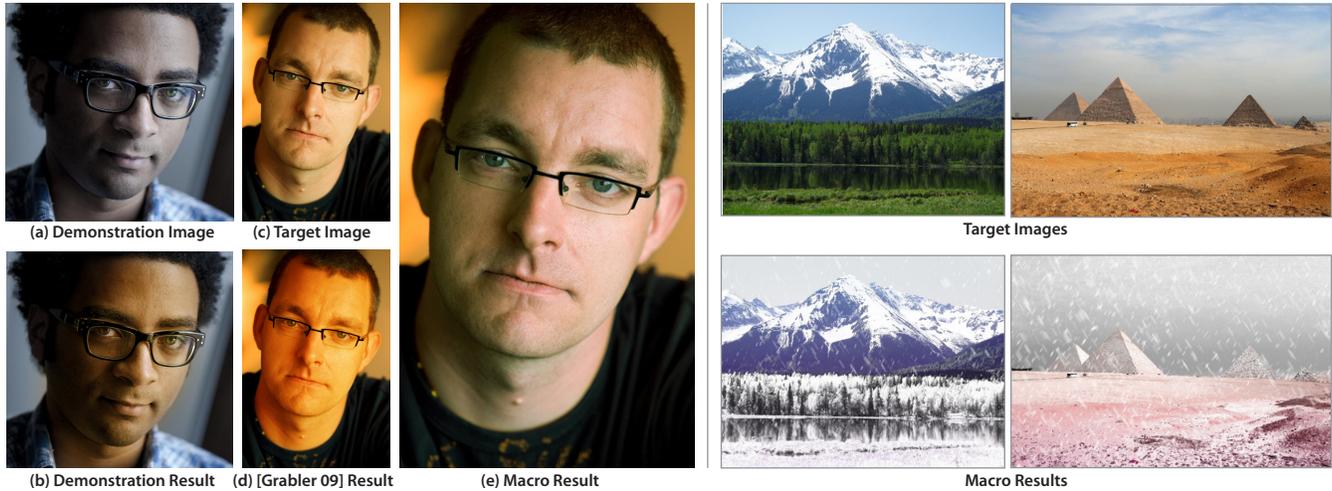


Fig. 1. (*Left*) A user demonstrates a skin-tone correction manipulation on a photograph of a dark-skinned person with a blue cast (a-b). Applying Grabler et al. [2009] and directly copying the same color adjustment parameters to correct the skin-tone of a light-skinned person turns his skin orange (c-d). Given 20 demonstrations of the manipulation, our content-adaptive macro learns the dependency between skin color, image color cast and the color adjustment parameters to successfully transfer the manipulation (e). (*Right*) After demonstrating a snow manipulation on 20 landscape images, our content-adaptive macro transfers the effect to several target images. Image credits: (a) *Ritwik Dey*, (c) *Dave Heuts*, right: *Ian Murphy*, *Tommy Wong*.

photo manipulations that we consider. In this work, we demonstrate our framework on three classes of the most commonly-used photo manipulations: face, landscape and global manipulations. In Section 8 we consider extensions to other classes of manipulations.

Our key idea is to learn the dependencies between image features (e.g. color, gradients, labels from object recognition [Zhou et al. 2003; Hoiem et al. 2005], etc.) and the locations of selection regions, the paths of brush strokes and the parameters of image processing operation. Unlike most previous techniques for transferring visual properties between images, which consider only input-output image pairs as training data, our approach requires multiple operation-level training demonstrations of the manipulation. We exploit knowledge of the low-level operations to more robustly adapt the manipulation to new target images.

One important advantage of our framework is that it supports a practical workflow for authoring and applying content-adaptive macros. Users create our macros the same way that they create traditional macros in Photoshop – by recording a demonstration of a sequence of image editing operations. While this recorded macro can automatically apply the manipulation to new target images after just one demonstration, the results usually require some corrections or modifications. Often the errors are due to either incorrect labeling from our automated image labelers or insufficient data for our learning algorithms. To help users detect and correct such errors, we provide visual feedback and interactive correction interfaces. Our system then uses each corrected transfer as an additional demonstration that improves the quality and robustness of the macro. Thus, authoring a content-adaptive macro is a continuous, incremental process that is simply a by-product of applying the recorded manipulation to new images. We show that after about 20 training demonstrations, corrections are rarely required and our framework has enough training data to successfully transfer many common face, landscape and global manipulations (see examples in Figures 1, 9 and 8). By allowing users to create macros in this incremental fashion and to easily modify or correct macro transfers, our framework enables a significantly more complete and usable workflow compared to purely automated techniques.

To evaluate the quality of our macro results, we ask viewers to compare images generated using our content-adaptive macros with

and without interactive corrections to manually generated ground-truth images. We find that they consistently rate both our automatic and corrected results as close in visual appearance to the ground-truth. We also conduct a small informal study with professional photographers, to evaluate our proposed macro authoring workflow. The results from this study suggest that the visual feedback and interactive correction features of our user interface are effective and practical in the context of real world image editing workflows.

In summary, our work makes three main contributions:

- **Framework:** We present a framework for learning the selections, brush strokes and image processing operations commonly used in photo manipulations. Using this framework we adapt face, landscape and global photo manipulations.
- **Learning from Multiple Demonstrations:** Our framework learns from multiple demonstrations, making it more robust than techniques that learn from input-output image pairs without knowledge of the low-level operations.
- **Feedback and Correction Interfaces:** We provide interfaces that help users detect and correct errors in automated labeling and parameter learning. These interfaces support an incremental approach to authoring content-adaptive macros.

## 1.1 Photo Manipulations: Target Domains

Our work focuses on face, landscape, and global manipulations because when people take pictures, they often capture people and outdoor scenes. We informally analyzed 151 tutorials from three popular online photo tutorial sites (chromasia.com, tripwiremagazine.com, good-tutorials.com) and found that 30% (45/151) of the tutorials were global manipulations (i.e., manipulations that affect the entire image and are not tied to a specific object in the scene), 26% (40/151) were face manipulations, and 18% (27/151) were landscape manipulations. The remaining 26% (39/151) of the tutorials were applied to objects not handled by the current labelers in our framework (e.g. hair, body parts, cars). The large number of face and landscape tutorials is not surprising given the importance of these two object categories. Because of their importance, computer vision researchers have developed specialized labelers for both faces [Zhou et al. 2003] and outdoor

scenes [Hoiem et al. 2005]. Some labelers detect landmark correspondence points such as the corners of the eyes, while others simply mark regions as belonging to an object category such as the ground or sky. We use of both types of labelers in our framework.

## 1.2 Limitations

Our framework cannot automatically adapt all types of image manipulations to new target images. If the correspondence between the demonstration image and target image is not correlated with any label or pixel-level image feature (e.g. color, gradients, etc.) our learning approach cannot correctly transfer spatial operations such as selections and brush strokes. For example, it would be difficult to learn a dust removal manipulation because dust can occur in arbitrary locations all over an image without any strong correlation with a label or pixel-level feature.

Image labels, and landmark correspondences are especially useful for transferring selections and brush strokes. Incorrect labels and landmarks can lead to poor transfers. Therefore we provide feedback and correction interfaces that allow users to manually rectify labels and landmarks when automated labeling fails.

Our framework can learn manipulations that have a well-defined sequence of operations that is repeated for various input images. It cannot learn subjective manipulations such as “face beautification” where the required operations may vary significantly depending on the input image. However, as we will show, our framework can transfer many well-defined components for such a manipulation, including adding makeup, removing bags under the eyes, etc.

## 1.3 Related Work

**Programming by demonstration.** Demonstration-based techniques for creating macros are compelling because they allow users to create a program by simply performing the target task, rather than learning a method to formally specify the underlying control logic [Cypher and Halbert 1993; Lieberman 2001]. Researchers have developed programming by demonstration tools to record and replay a sequence of operations in the context of desktop [Modugno and Myers 1994; Lau et al. 2004], web [Little et al. 2007; Bolin et al. 2005], and 2D graphics [Kurlander and Feiner 1992; Lieberman 1993] applications. A few researchers have also explored strategies for adapting such macros to new inputs. Kurlander and Feiner’s [1992] pioneering work on Chimera uses heuristics to generalize macros for manipulating 2D graphics. None of these techniques are designed to adapt photo manipulation macros.

**Image-based transfer of visual properties.** A recent trend in image manipulation research has been to develop algorithms for transferring certain visual properties of an example image to a target image. For example, image analogy methods [Hertzmann et al. 2001; Efros and Freeman 2001; Drori et al. 2003] take a neighborhood-based approach to transferring low-level texture properties and can imitate non-photorealistic rendering styles. Bae et al. [2006] use a histogram-based approach to transfer contrast and thereby replicate the overall look of the example image. Reinhard et al. [2001] adjust the statistics of the color distributions of the target image to match those of the example. While these techniques inspire our work, they deal with pixel-level models of the images and do not have access to higher-level information such as the content of the image or the set of selection regions, brush strokes or operations an author may have used to create the example image. As a result they cannot adapt operation-level macros to work with new content.

**Content-specific transfer algorithms.** Researchers have also developed techniques to transfer visual properties for specific types

of images. For example, human faces are important elements of many photographs, and researchers have explored image-based techniques to transfer expressions and lighting [Liu et al. 2001], makeup [Guo and Sim 2009], beards [Nguyen et al. 2008] and entire faces [Bitouk et al. 2008] from one image to another. These techniques typically rely on identifying corresponding facial regions (e.g. mouth, eyes, skin, etc.) between the two images either manually or using face-recognition algorithms, and then applying manipulation specific pixel-level transfer algorithms to these regions. While our approach also takes advantage of automatic image labeling, including face and outdoor scene recognition in our implementation, our framework is designed to work with generic labels and landmark points.

**Operation transfer algorithms.** In previous work [Grabler et al. 2009], we propose a technique for transferring selection operations from one image to another using automatic image labeling [Zhou et al. 2003; Hoiem et al. 2005]. This approach has a few key limitations: it cannot transfer brush strokes, partial selections, or selections that depend on non-label image features (e.g., color); it also cannot adapt the parameters of image processing operations. Kang et al. [2010] present a technique to transfer color and contrast adjustment operations, but they cannot transfer selections or brush strokes. Given a new target image, they find the nearest neighbor amongst a set of 5000 training images and then apply the corresponding adjustment parameters to the new target. A drawback of their approach is that learning a new manipulation requires a new set of 5000 training examples. Finally, Hasinoff et al. [2010] use an image-based approach to transfer clone brush operations. They rely on finding pixel-level matches between the training and target images. Thus, their approach is designed primarily for use with image collections from the same photo shoot. Compared with these methods, our framework is more comprehensive as it learns to adapt a wide variety of photo manipulations comprised of selections, strokes and image adjustment operations. Our approach usually requires about 20 training demonstrations and can transfer local edits such as partial selections and brush strokes across images from different photo shoots. Thus, our content-adaptive macros are more suitable for sharing and reuse by other photographers.

## 2. OVERVIEW

Our framework for generating content-adaptive macros contains three main components (Figure 2). The *demo* component captures example manipulations as authors demonstrate them in Photoshop (Section 3). The *learn* component takes one or more example demonstrations as input and produces a content-adaptive macro as output (Section 4), while the *apply* component executes the macro on new target images. Our framework also provides visual feedback and interactive controls for correcting errors that may arise as the macro is demonstrated or applied. These interfaces enable an incremental workflow where each time a macro is applied and corrected it becomes a new training demonstration (Section 5).

Our framework relies on automated image labeling to segment input images (both example and target images) into labeled regions. Our *image labeler* currently applies a face recognizer [Zhou et al. 2003], a skin recognizer [Jones and Rehg 2002], and an outdoor scene recognizer [Hoiem et al. 2005] to automatically detect such regions. We use the default parameter values for these recognizers, and together they label a  $600 \times 900$  image in about 10 seconds. Our framework is extensible and can incorporate new recognizers as they become available (see Section 8).

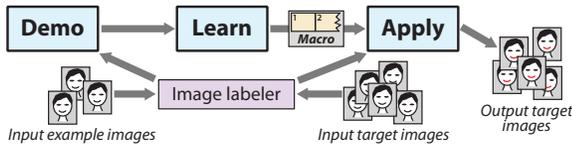


Fig. 2. Framework for generating content-adaptive macros.

### 3. RECORDING DEMONSTRATIONS

Image manipulation software like Photoshop and GIMP provide access to a diverse set of image editing tools. Analyzing this software, we observe that the tools allow users to perform three types of low-level operations; selections, brush strokes and image processing operations. Each low-level operation includes user-specified parameters that affect how they operate. We define the low-level operations and give example parameters in parentheses.

- **Selections:** These operations allow users to select a region of the image. The default selection is the entire image. Subsequent operations affect only the pixels within the selection region. Examples: free-select (location of the selection), by-color-select (range of colors to select).
- **Brush strokes:** These operations affect all pixels that lie within a brush region surrounding a stroke path. The stroke path and diameter parameters control the location and size of the stroke. Brushes may also include parameters specific to the adjustment. Examples: color brush (brush color), blur brush (blur strength).
- **Image processing operations:** These operations modify the pixels within the selection region by applying a filter, a color adjustment, or a spatial transform to them. Examples: sharpening filter (sharpening radius), contrast adjustment (strength), and rotating the selection region (angle).

Our system leverages the action recording capabilities of Photoshop’s built-in *ScriptListener* plug-in to capture the sequence of low-level operations and parameter settings as a user demonstrates a manipulation. We then apply the clean-up and grouping techniques of Grabler et al. [2009] to simplify the raw recordings by eliminating unnecessary operations and merging repetitive operations. Our framework requires that users perform all of the example demonstrations for a particular manipulation using the same sequence of operations, so that after clean-up and grouping the sequences are in one-to-one correspondence with one another. In Section 5 we present an interface designed to help users demonstrate the operations in the same order.

### 4. LEARNING CONTENT-ADAPTIVE MACROS

To generate content-adaptive macros, our framework learns the dependencies between image features and the parameters of the selections, brush strokes and image processing operations comprising a manipulation. We first present the set of image features our framework considers (Section 4.1) and then describe the algorithms for transferring selection regions (Sections 4.2), adapting the locations of brush strokes (Section 4.3) and learning non-spatial adjustment parameters for the operations (Section 4.4).

#### 4.1 Features

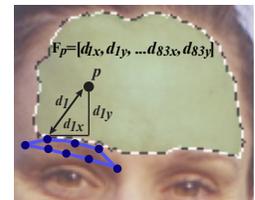
For machine learning algorithms to adapt photo manipulation macros, it is essential to identify features that correlate with the parameters of the selections, brush strokes and image processing operations. We analyzed the relationship between parameter settings and image features in common photo manipulations [Huggins

2005; Kelby 2007] and found that users often set parameters based on the following features.

**Pixel-level features** are local descriptors of the image that are based on pixel values. Such features include color, contrast, saturation, luminosity, texture, etc. Selections, brush strokes and image processing operations frequently depend on such features. For example, a user might increase the contrast of a region based on its current contrast or apply the healing brush to a dark (low luminosity) region under the eyes to remove bags. Many pixel-level features encode redundant information since they are all based on pixel values. We have found that *color features* ( $L^*a^*b^*$  space pixel value), *contrast features* ( $L^*$  gradient magnitude) and *luminosity histogram features* (range, peak and median of histogram of  $L^*$  values) are sufficient for learning pixel-level dependencies. Although the luminosity histogram features are partially redundant with the  $L^*$  color feature, we include them because many image processing operations (e.g. color curve adjustment, luminance levels) are designed to directly modify aspects of the luminosity histogram. Therefore these features are better predictors of many image processing parameters than the  $L^*$  color feature alone.

**Label-based features** describe the semantic content of an image. The location of selections and brush strokes often depend on such image content. For example, a user might select the forehead of a person to reduce its shininess or brush along the horizon of the sky to intensify a sunset. Similarly, parameters like the brush diameter often depend on the size of the object that the user brushes over. To capture these dependencies, we use *label features*, *landmark offset features* and *size features*. Our image labeler (Section 2) applies recognizers to compute a set of labels (e.g. eyes, skin, sky, ground, null) for every pixel in the image. The *label feature* is a bit vector representing the labels associated with each pixel. Some recognizers also provide landmark correspondence points that can serve as references for location parameters. For example, Zhou et al.’s [2003] face recognizer detects the 2D positions of 83 landmark points (e.g. the outside corner of the left eye) that correspond across images of different faces. However, our skin [Jones and Rehg 2002] and outdoor scene [Hoiem et al. 2005] recognizers do not provide landmark points. Since the shapes of skin and outdoor regions like sky, ground, etc. can vary drastically from one image to another, it is unclear how to put such shapes in correspondence. We therefore use a weak shape descriptor and treat the 4 corner vertices of the axis-aligned bounding box of each labeled region as landmarks. We leave it as future work to identify more appropriate shape descriptors when landmarks are not available.

We compute *landmark offset features* as the offset  $x$ - and  $y$ -coordinates between the pixel and each landmark point returned by our labeler (see inset). To better capture spatial relationships between selection or brush stroke locations and the image, we also include the 4 corner vertices of the image and of the axis aligned bounding box of the selection region as landmarks in computing the landmark offset features. We normalize the landmark offsets by the width and height of the labeled region, selection region or the entire image depending on the region the landmark point belongs to. Although the offset distance to a single landmark point may be weakly correlated with location, the offset distances to a collection of landmark points can be very discriminative. We compute the *size features* as the width and height of the bounding box of each labeled region.



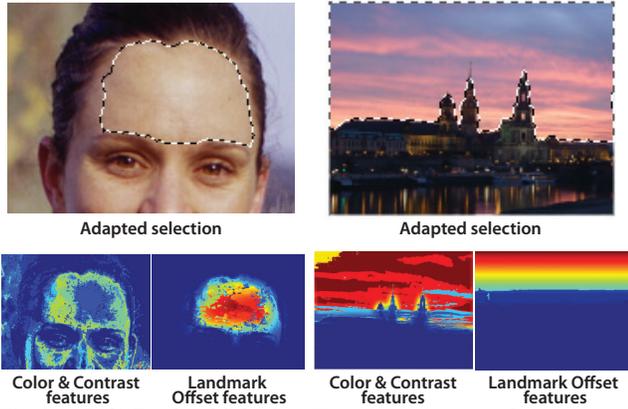


Fig. 3. **(Left) Forehead Selection.** The color and contrast features cannot discriminate between the forehead and skin area on the face. However, landmark offset features accurately predict the location of the selection. **(Right) Sky Selection.** In this case the color and contrast features are better predictors of the selection than the landmark offset features. Image credit: Franz Reichard

**Operation-based features** are characteristics that are specific to a given operation. We include two such features that are characteristic of brush stroke operations; the *stroke length feature* describes the length of the stroke in pixels and is stored as a single value for each stroke, while the *stroke orientation feature* encodes the local orientation of the strokes as a discrete value from 1 to 8 indicating for each pixel on a stroke, the position of the next pixel on the stroke. Together these features enable our framework to transfer strokes that have approximately the same length and shape as the strokes in the demonstration.

## 4.2 Adapting Selections

Users typically select a set of pixels in the image that have a common property; e.g., they belong to the same object, they are similar in color, etc. To adapt selections to new target images, we model the selection regions of the training images using a feature vector  $\mathbf{F}_j = [F_{j,1}, \dots, F_{j,N}]$  where  $j$  denotes a selected pixel in a demonstration image and  $1 \dots N$  are the features computed for each such pixel. We then apply this model to classify each pixel  $i$  of the target image into two categories, selected ( $Sel = 1$ ) or not selected ( $Sel = 0$ ), based on its feature vector  $\mathbf{F}_i = [F_{i,1}, \dots, F_{i,N}]$ .

We use Naive Bayes for this classification task, because it is simple and flexibly allows adding new features. Although Naive Bayes assumes features to be conditionally independent given a class, it is known to work surprisingly well in object categorization and information retrieval, even when this assumption is violated [Lewis 1998]. We determine whether pixel  $i$  is part of the selection region by computing  $P(Sel = 1 | \mathbf{F}_i) =$

$$\frac{P(\mathbf{F}_i | Sel = 1)P(Sel = 1)}{P(\mathbf{F}_i | Sel = 1)P(Sel = 1) + P(\mathbf{F}_i | Sel = 0)P(Sel = 0)}. \quad (1)$$

Because of the independence assumption, the likelihood of a feature vector for a selected pixel is given by

$$P(\mathbf{F}_i | Sel = 1) = \prod_{k=1}^N P(F_{i,k} | Sel = 1). \quad (2)$$

Rewriting equation 1 we obtain  $P(Sel = 1 | \mathbf{F}_i) =$

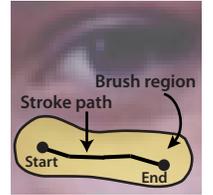
$$\frac{\prod_k P(F_{i,k} | Sel = 1)P(Sel = 1)}{\prod_k P(F_{i,k} | Sel = 1)P(Sel = 1) + \prod_k P(F_{i,k} | Sel = 0)P(Sel = 0)}. \quad (3)$$

To compute  $P(F_{i,k} | Sel = 1)$  and  $P(F_{i,k} | Sel = 0)$  we build histograms of all observed values of the  $k^{th}$  feature within either the selected or unselected pixels of the demonstration images. The prior probability  $P(Sel = 1)$  corresponds to the percentage of selected pixels across all training examples. We then include all pixels of the target image where  $P(Sel = 1 | \mathbf{F}_i) > t_{sel}$  in the selection region for that image. We set the selection threshold  $t_{sel} = 0.27$  for all examples in this paper. Finally, we apply morphological operators to the selection mask to eliminate thin strips and isolated pixels.

The feature vector  $\mathbf{F}_i$  is a  $5 + 2N + L$ -dimensional vector composed of the 5 pixel-level features (3 for color, 2 for contrast),  $2N$  landmark offset features where  $N$  is the number of landmarks in the image and  $L$  label features, where  $L$  is the number of detected labels in the demonstrations. Figure 3 shows how all the features are necessary to adapt selections to new target images

## 4.3 Adapting Brush Strokes

A brush stroke is composed of a brush region and a stroke path (see inset). Brush regions are equivalent to selection regions, and we use the same kind of Naive Bayes classifier as in Section 4.2 to adapt the brush region to a new target image. A stroke path is a curve representing the centerline of each brush stroke. We compute stroke paths using a Markov Chain model in which the next point on the stroke depends only on the state of the previous point on the stroke  $P(S_i | S_{i-1})$ . Markov Chains are commonly used to model time-varying processes and have been successfully applied to example-based synthesis problems in many domains including music [Schwarz 2005], text [Dewdney 1989], and curves [Hertzmann et al. 2002; Kalnins et al. 2002; Simhon and Dudek 2003]. While our approach is similar to previous example-based curve synthesis techniques, unlike the earlier methods which only consider geometric features of the example curves, our approach takes advantage of both geometric features as well as image-based features of the underlying example images. In our stroke model, we define three types of states:  $S_{start} = (x, y)$  is the stroke start point in image coordinates,  $S_i = (x, y)$  is the  $i$ -th point on the stroke and  $S_{end}$  is a state marking the end of a stroke. We proceed in three steps:



**Step 1: Initialize stroke start point.** For each brush region we determine the position of the stroke start point. Since the start point is not preceded by any other state, we pick the point  $p = (x, y)$  within the brush region that maximizes  $P(S_{start} = p)$ . In our model a point  $p$  is fully characterized by its feature vector  $\mathbf{F}_p = [F_{p,1}, \dots, F_{p,N}]$  and  $P(S_{start} = p) = P(\mathbf{F}_p)$ . We assume that each feature  $F_{p,k}$  in the feature vector is independent so that  $P(\mathbf{F}_p) = \prod_{k=1}^N P(F_{p,k})$ . To compute  $P(F_{p,k})$  we build a histogram for all observed values of the  $k^{th}$  feature across the set of known stroke start points in the demonstration brush strokes.

**Step 2: Choose next stroke point.** Once we have computed the start position, we treat the stroke path as a process where the location of the next stroke point is based on the location of the previous stroke point. The state  $S_i$  of the  $i$ -th point on the stroke can either specify the position of the point in image coordinates  $S_i = (x, y)$ , or specify that the stroke should end  $S_i = S_{end}$ . To determine the next state, we compute

$$\operatorname{argmax}_{S_{i+1}, q \in N(p)} \{P(S_{i+1} = q | S_i = p), P(S_{i+1} = S_{end} | S_i = p)\} \quad (4)$$

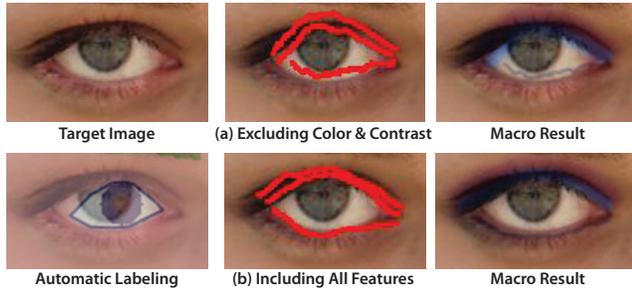


Fig. 4. Landmark offset features provide a rough estimate of the location of the stroke path (a). But because the image labeler slightly mislabeled the eye, color and contrast features are necessary to correctly adapt the strokes to the contour of the eye (b).

where  $p = (x, y)$  is the position of the previous point on the stroke and  $q$  is chosen from the set of points in the neighborhood  $N(p)$  of  $p$ . In practice we set  $N(p) = \{(x \pm 1, y \pm 1)\}$ , which is the 8-connected pixel neighborhood of  $p = (x, y)$ .

Since points are characterized by their feature vectors, we compute  $P(S_{i+1} = q | S_i = p)$  as  $P(\mathbf{F}_q | \mathbf{F}_p)$ . We assume that the individual features  $F_{p,k}$  of each feature vector  $\mathbf{F}_p = [F_{p,1}, \dots, F_{p,N}]$ , are conditioned only on themselves and independent of the other features, so that  $P(\mathbf{F}_q | \mathbf{F}_p) = \prod_{k=1}^N P(F_{q,k} | F_{p,k})$ . To compute  $P(F_{q,k} | F_{p,k})$  we build histograms of the transition probabilities for each feature  $k$  in the training demonstrations. Similarly we compute  $P(S_{i+1} = S_{end} | S_i = p) = \prod_{k=1}^N P(S_{end} | F_{p,k})$  by building a histogram of the feature vectors for the last point of each stroke in our training data.

To reduce the number of training examples needed, it is often assumed that the markov chain is time homogeneous, such that  $P(S_{i+1} | S_i) = P(S_{i+t+1} | S_{i+t})$  for all  $t > 0$ . To better capture the variation of each feature along the path, we assume that the process is time-homogeneous within a limited number of timesteps. We have found that  $t = 5$  works well for color, contrast and landmark offset features, while  $t = 1$  is necessary for stroke orientation and stroke length, since these features vary more quickly along the stroke path. While our approach requires more training data than setting  $t = \infty$ , we have found it to result in better transfers because it better captures the evolution of each feature along the path.

The resulting stroke path gives a greedy estimate of maximizing the likelihood. There are a variety of algorithms that could be used to refine the curve if desired [Kass et al. 1988; Amini et al. 1996], but we have found our approach to work well in practice.

**Step 3: Update brush region.** After computing the stroke path, we dilate the path by the brush diameter  $d$  (we compute  $d$  using the approach of Section 4.4) and subtract the dilated stroke from the brush region mask. We then repeat this process and start a new stroke, until the area of the brush region mask is too small ( $< 10\%$  of the brush area of the smallest stroke in the training data).

We use slightly different feature vectors  $\mathbf{F}_p$  in steps 1 and 2. In step 1,  $\mathbf{F}_p$  includes only three image-based features; color, contrast and landmark offset. For the color and contrast features, we use the median feature values across all pixels that lie within the brush diameter  $d$  of stroke point  $p$  in the training data. In step 2,  $\mathbf{F}_p$  includes two additional geometric features; stroke orientation and stroke length.

Through informal tests we have found that all features provide useful information to the model. For example, Figure 4 shows a brush stroke transfer for an eye makeup manipulation. Because of the color and contrast features, we can correctly transfer strokes even when the eye is slightly mislabeled.



Fig. 5. Adapting the skin-tone manipulation. The 20 training examples contain many images that require shifting the color balance towards yellow. Least Squares overfits the data and exhibits a yellow cast. LARS avoids overfitting and produces a better result. Image credit: *Marc Ducrest*.

#### 4.4 Adapting Adjustment Parameters

To adapt the numerical image processing parameters to new target images, we must learn how the parameter values depend on the underlying image features. Our goal is to learn a function that maps the image features to a parameter value. Suppose  $y_i$  are observations of a parameter  $y$  we wish to learn and  $\mathbf{F} = [F_1, \dots, F_N]$  are a set of image features. Under a linear model,  $y_i = c_1 * F_1 + \dots + c_N * F_N$ , we must compute the set of regression coefficients  $c_i$  that best explain our observations. Linear regression is a simple technique for computing these coefficients, but it can lead to overfitting. To avoid such overfitting, we use Least Angle Regression (LARS) [Efron et al. 2004] which is a variant of linear regression that constrains the sum of the absolute regression coefficients  $c_i$  and thereby causes many coefficients to be zero. This property ensures that only a small set of image features are used for the prediction of a parameter. With 20 training examples we observe that LARS produces better results than least squares for many manipulations including the skin tone manipulation (Figure 5).

Image processing operations are applied to an active selection region or brush region. Often the intent is to strengthen or weaken the characteristics of this region with respect to the surrounding pixels. We therefore compute the pixel-level features for the active region and its complement. We use the median color and contrast features computed over the corresponding regions. However, several adjustment parameters (e.g. brush diameter, scale factor) depend on the size of objects in the image and so we include size features for all labeled regions that overlap the active region. To summarize,  $\mathbf{F}$  is a  $18 + 2N$ -dimensional vector composed of 18 pixel-level features (9 for the selection region and 9 for its complement, consisting of 3 for color, 2 for contrast and 4 for the luminosity histogram) and  $2N$  size features, where  $N$  is the number of labeled regions that overlap with the active region.

## 5. WORKFLOW: FEEDBACK AND CORRECTION

In our framework, the user can create a content-adaptive macro by recording a single demonstration of a manipulation. However, with only one training example, the macro cannot robustly adapt to new target images. Typically, the user needs to demonstrate the manipulation using the same sequence of steps on about 20 different examples to produce a robust macro. To help the user demonstrate and apply the manipulation to new target images, our framework provides two visual feedback and interactive correction panels. The *macro application panel* applies the macro to a new image and then allows the user to correct the transferred selections, brush strokes and adjustment parameters as necessary (Figure 6 left). The *labeling correction panel* helps macro authors correct errors due to our automated image labelers (Figure 6 right).



Fig. 6. (Left) **Macro Application**. The panel shows the target image after each step of the macro. Selection steps show the selection region in gray. Brush stroke steps show the brush stroke in green. Users can click on any step in the panel to load the corresponding images in Photoshop and modify parameters as necessary and then re-executing the remaining steps. (Right) **Labeling Correction**. The panel shows the set of facial features detected. The eyes, nose and lips are clearly mislabeled (b). After the user selects the lips, the current selection is drawn in green (a). To correct the lip labels, the user can check the upper and lower lip in the list below; our framework associates the selection region with all checked labels (c). Image credit: *Perdikopoulos Charalampos (tariq.ante)*.

### 5.1 Macro Application Panel

Suppose an author demonstrates a lip gloss manipulation. After the first demonstration (see inset, Image credit: *Lon & Queta*), our framework generates a macro application panel. Given a new target image, the panel presents the six steps comprising the manipulation and the image that results after applying each step to the new target (Figure 6 left). The visual feedback allows the author to quickly identify and correct errors in individual steps of the macro. Since the sequence of steps is fixed in the panel, it also prevents the author from forgetting steps or changing the order of steps when adding additional demonstrations.



In this example, the panel immediately reveals an error in adapting the selection region in the first step – our image labeler incorrectly labeled the lips. The user can click on any step in the panel and our framework executes all of the operations up to that step. The user can then take over and fix any incorrect parameters. In this case, the user clicks on the first selection step, manually selects the correct lip region and then uses our *labeling correction panel* to mark his new selection as lips (Figure 6 right). Although such manual labeling is not strictly necessary and our framework can learn a macro without it, more accurate labels yield more accurate macro adaptation. Similarly if the automatically generated red lip color were undesirable, the user could adjust the parameters of the color curve that reddens the lips by clicking on step four. After each such correction, our framework automatically executes and re-evaluates the parameters of all remaining steps. Once the user has finished correcting the image, our framework treats the target image with the corrected parameter settings as an additional demonstration.

Our framework immediately reveals an error in adapting the selection region in the first step – our image labeler incorrectly labeled the lips. The user can click on any step in the panel and our framework executes all of the operations up to that step. The user can then take over and fix any incorrect parameters. In this case, the user clicks on the first selection step, manually selects the correct lip region and then uses our *labeling correction panel* to mark his new selection as lips (Figure 6 right). Although such manual labeling is not strictly necessary and our framework can learn a macro without it, more accurate labels yield more accurate macro adaptation. Similarly if the automatically generated red lip color were undesirable, the user could adjust the parameters of the color curve that reddens the lips by clicking on step four. After each such correction, our framework automatically executes and re-evaluates the parameters of all remaining steps. Once the user has finished correcting the image, our framework treats the target image with the corrected parameter settings as an additional demonstration.

### 5.2 Labeling Correction Panel

The labeling correction panel has three components. The top left image shows the current selection region in green, overlaid on the target image (Figure 6a). The top right image visualizes the labeled

regions detected by our image labelers (Figure 6b). Below, the list of checkboxes indicates which of the detected labels correspond to the current selection (Figure 6c). Figure 6 shows the state of the panel after the user has noticed the error in step 1 and selected the correct lip region using Photoshop’s selection tool. To label the current selection as lips, the user simply marks the upper and lower lip checkboxes; our framework then associates these labels with the current selection. Some labelers, such as our face labeler, also provide landmark correspondence points. Although our labeling correction panel does not automatically update such landmark points, we provide an advanced interface for users to manually update the location of these points.

### 5.3 Macro Robustness

One challenge for a macro author is to evaluate when the macro is robust enough to distribute to other users. Thus, our macro application panel provides feedback on the robustness of the current macro. After every new demonstration, we run a leave-one-out cross-validation on the existing demonstration dataset. We use one demonstration as the test data and the remaining demonstrations as the training data. We repeat this process for several rounds until every demonstration has been used once as test data.

For each round, we estimate how well the content-adaptive macro is able to adapt the parameters for the test image. For each step of the manipulation, we compute the mean squared error (MSE) between the image generated by our adapted macro and the ground truth image that we obtained with the original demonstration. Since many steps modify only a small region of the image, we avoid excessively low MSEs by including only the set of pixels modified in either the adapted image or ground-truth image. Since selections do not modify the image, we compute MSE on the selection mask for the selection steps.

The macro application panel displays the average MSE across all rounds for each step. Although MSE does not capture the perceived difference of pixel values, it indicates the level of similarity between the learned result and the ground truth demonstration. Uniformly low MSE values for all steps suggest that the macro is relatively robust and can be shared with other users. We typically

Manipulations	Number of				Dataset Size
	Sel.	Strokes	Params.	Total	
Bag Removal	0	2-4	7	9-11	100/19
Contrast	1	0	4	5	100/18
Eye Makeup	0	6	6	12	100/49
Film Noir	0	5	23	28	100/15
Lip Gloss	1	1	5	7	50/4
Mustache	0	3-10	0	3-10	35/4
Skin Tone	1	0	6	7	50/14
Dark Sky	1	1	4	6	50/14
HDR realistic	1	5-9	26	32-36	50/7
HDR artistic	1	5-9	26	32-36	50/7
Landscape Enhancement	0	5-7	16	21-23	40/8
Rainbow	0	8-12	8	16-20	60/17
Smooth Waterfall	1-5	3-5	11	15-21	50/0
Snow	0	2	24-28	26-30	50/11
Sunset Enhancement	2	2-6	18	22-26	100/62
Black & White	0	0	6	6	100/0
Cross-processing	0	0	17	17	50/0
Lomo	0	0	11	11	50/0
Reflections	1-4	1-2	16	18-22	25/0
Typographic	1	0	2	3	35/0

Fig. 7. Twenty manipulations (face – gray, landscape – green, global – pink) and the number of selections, brush strokes and adjustment parameters adapted for each one. The dataset size column reports the number of images used in the evaluation, followed by the number of images for which we hand-corrected the labeling.

observe low MSEs with 15-20 demonstrations. Note that this MSE measure only indicates robustness with respect to the demonstration images. It remains the task of the author to collect a training set that contains enough variety.

## 6. RESULTS

We have used our framework to generate content-adaptive macros for the 20 manipulations described in Figure 7, some of which are shown in Figures 1, 9, 8 and 11. All of these results were generated using our fully automated approach without any corrections through our application or labeling panels. We chose 7 face manipulations, 8 landscapes adjustments, and 5 global manipulations from popular photo editing books [Huggins 2005] and websites. The manipulations range from adding eye makeup to smoothing waterfalls to making the image look as if it were taken with a Lomo camera. In total the manipulations require adapting between 3 and 36 selections, brush strokes and adjustment parameters. For example, the smooth waterfall manipulation has the following steps (see supplemental website<sup>1</sup> for descriptions of the other manipulations):

- (1) Select waterfalls and copy them onto a new layer (1-5 selections).
- (2) Apply motion blur filter on waterfall layer (2 parameters).
- (3) Create a layer mask and paint over areas where blur effect goes outside the waterfall regions (3-5 brush strokes).
- (4) Adjust opacity of waterfall layer (1 parameter).
- (5) Use warp tool to emphasize shape of waterfall (8 parameters).

To find suitable images for each manipulation we asked photographers and searched the Web (flickr.com) to build a dataset of high-quality images that photographers would want to manipulate. The supplemental website provides specific criteria for each dataset.

<sup>1</sup>All supplemental materials are at <http://vis.berkeley.edu/papers/macros/>

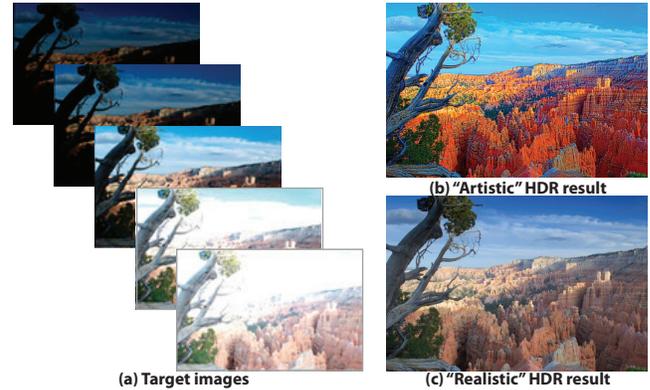


Fig. 8. High dynamic range (HDR) results. Our system successfully transfers both “artistic” (b) and “realistic” (c) HDR manipulations based on the training demonstrations. Image credit: Mark Fairchild’s HDR Photographic Survey.



Fig. 10. Less successful adaptations. Poor parameter estimation results in a washed out image (a). Incorrect selection transfer causes blurring in the region below the waterfall (b). Incorrect labeling results in misplaced brush strokes for the mustache (c) and eye makeup (d) manipulations. Image credits: (b) Ed Yourdon, (c) Jason Hill.

As shown in Figures 1, 9 and 8, in most cases our content-adaptive macros successfully adapt the manipulation to new target images. Figure 9 includes 2 face manipulations (b-c), 2 landscape manipulations (d-e) and 2 global manipulations (f-g). The makeup and mustache manipulations (b-c) demonstrate the precise transfer of brush strokes when our labeler recognizes the faces. The waterfall manipulation (d) edits a semantic region of the image – namely the water. Although our framework does not include a water detector, it is still able to successfully transfer the manipulation to many images using the white color of the water (color feature) and the relative location of the water below the sky (landmark offset feature). The snow manipulation (Figure 1) includes up to 28 adjustment parameters. When correctly transferred, these parameters cause the ground, trees and even the reflected trees in the lake to appear as if they have snow on them. We include two versions of the same HDR manipulation, one trained with examples that produce an “artistic” look and the other trained to produce a “realistic” look. Figure 8 shows that our framework can learn the appropriate parameters for both looks.

Except for the mustache manipulation, all of these results were generated using 20 training demonstrations. Because adding a mustache only involves brush strokes and does not include adjustment parameters, we trained the macro using just 10 demonstrations. Some of the adaptation effects can be visually subtle and we en-

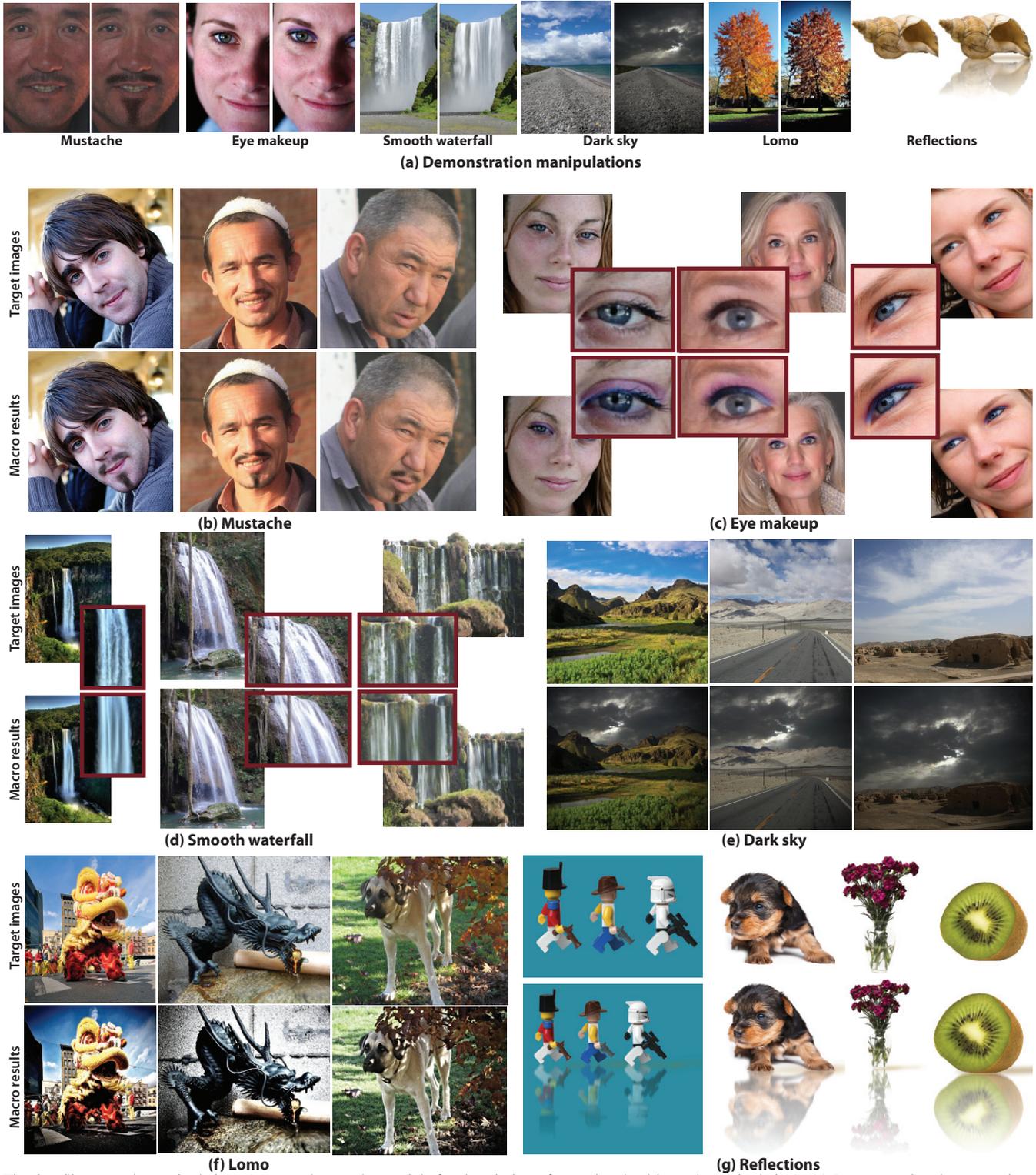


Fig. 9. Six example manipulations (see supplemental materials for description of steps involved in each manipulation). (a) Representative demonstration input image and result pairs. (b)–(g) Our content-adaptive macro results on new target images. The mustache results (b) were generated using 10 training demonstrations; the other results (c)–(g) were generated using 20 training demonstrations. All of these results were generated using our fully automated approach without any corrections through our feedback panels. Some of the adaptation effects can be visually subtle, and we encourage readers to zoom in and look at these results on screen. Image credits (top to bottom, left to right): (a) PC, Steve McFarland, Elizabeth Thomsen, PC, Barbara Miers, Klaus Post, (b) Rob Stanley, PC, PC, (c) PC, James Bardin, Nicolas Couthouis, (d) Luciano Meirelles, Verena Jung, PC, (e) Michael McCullough, PC, PC, (f) Bob Jagendorf, Chris Gladis, PC, (g) pasukaru76, Michal Gajkowski, Klaus Post, Klaus Post. (Note that PC stands for personal photo collection.)

courage readers to zoom in and look at these results on screen. Figure 10 shows a few less successful examples that include misplaced brush strokes and poor parameter estimation. Both types of errors could be corrected using our application panel.

Each manipulation initially takes about 5-15 minutes to demonstrate, but with practice and using our macro application panel this authoring time usually reduces to about 1-5 minutes per demonstration. With 20 training demonstrations and images of size  $600 \times 900$ , our system requires about 3 minutes to learn how to adapt each selection, and about 1 minute to adapt each brush stroke or adjustment parameter. Thus, the training time for our manipulations is between 5 and 40 minutes depending on the number of operations. Applying the resulting macro to a new target image requires a few seconds to adapt each selection region and brush stroke operation, and significantly less than a second to adapt each adjustment parameter. These timings are based on our unoptimized MATLAB implementation.

## 7. EVALUATION

To determine the effectiveness of our framework as a whole, we evaluate both the quality of the images generated by our content-adaptive macros, as well as the utility of our proposed macro authoring workflow.

### 7.1 Macro Results

To evaluate the quality of our macro results, we chose seven representative manipulations: bag removal, contrast, dark sky, eye makeup, film noir, lomo, and sunset enhancement. For each manipulation, we compare four different methods for adapting the manipulation with a dataset of 50–100 images:

- (1) **Ground-truth:** We generate *ground-truth* images by adapting the manipulation manually for each image in the dataset.
- (2) **Average:** As a baseline, we generate *average* images by averaging adjustment parameters across all training demonstrations and copying any selection regions and brush strokes (renormalized to account for differences in image size) from the demonstration image closest in size to the target image<sup>2</sup>.
- (3) **Automatic:** We generate *automatic* images using our content-adaptive macros without any manual correction using our correction interfaces.
- (4) **Corrected:** To factor out the effect of errors due to incorrect image labeling, we also compare against *corrected* images that we generate by manually correcting poor labeling from our automated labelers using our labeling correction panel. Note that we did not use the macro application panel to correct any other parameter adaptations.

Figure 11 shows these adaptations for the film noir manipulation. We do not show our *corrected* result because the image labels were correct and thus the *corrected* and *automatic* results are identical.

As described in Section 4.4, the regression technique we use for learning adjustment parameters requires 20 demonstrations to work robustly. Thus, we train our macros on a random subset of 20 images. In addition, to investigate how our macro results vary with the number of training demonstrations, we choose five of the manipulations (bag removal, contrast, eye makeup, film noir, and sunset enhancement) and compare results with random subsets of 1, 10,

<sup>2</sup>We designed this baseline as a relatively straightforward extension to existing macro systems such as Adobe Photoshop’s *Actions*.

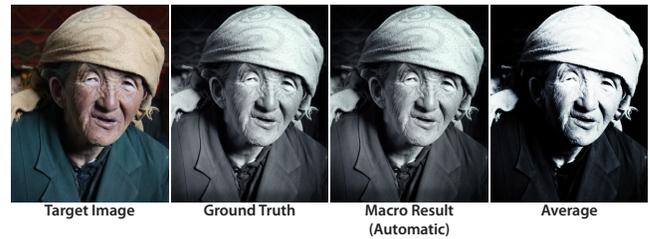


Fig. 11. Three adaptation methods for the film noir manipulation. We do not show our *corrected* result, because the image labels were correct and therefore it is identical to the *automatic* result.

20 and 30 training demonstrations. We use the average parameter values to generate our *automatic* and *corrected* results when we have only 1 or 10 training demonstrations.

We compare the four adaptation methods using an Amazon Mechanical Turk study in which participants rate the differences between the manipulated images. We chose to run a user-based evaluation because most standard image difference metrics such as mean squared error (MSE), are not perceptual measures. However, for completeness, we include MSE comparisons in the supplemental materials. We also compute the absolute difference in parameter space between *ground-truth* and our macro results. Across 11 manipulations we find that 79 out of 89 parameters generated by our *automatic* and *corrected* methods are closer to the *ground-truth* parameters than those generated by the *average* method. We include comparisons our previous method [Grabler et al. 2009] where applicable in the supplemental website.

Finally, to get a sense for the overall quality of our macro results, we manually counted the number of successful and less successful *automatic* results for all 20 manipulations. With 20 training demonstrations, our overall success rate is 82%, ranging from 95% (for lip gloss, skin tone, black & white, and lomo) to 60% (for eye makeup). Our eye makeup macro is less successful because the face labeler did not find any face in 15 of the 80 test images. We report the MSE and parameter difference results in our supplemental PDF document. Our supplemental website includes success rates and sample macro results for all 20 manipulations.

### Mechanical Turk Study Design

We used the Amazon Mechanical Turk to test how well the *automatic*, *average* and *corrected* images match the *ground-truth*. We simultaneously showed Mechanical Turk workers four images marked A, B, C, and D; we asked them to rate the differences between images B, C and D, and image A on a scale of 1 (“indistinguishable from image A”) to 5 (“very different from image A”). In our first study, we labeled the *ground-truth* as A and counter-balanced the ordering of the *ground-truth*, *average* or *automatic* images with respect to labels B, C, and D. In a second study, we replaced the *automatic* images with the *corrected* images. Each task was completed by 5 different workers, and we paid 3 cents per task.

### Findings

There are four notable findings from our user-based evaluation:

#### 1. Macro results are better than average results

The mean difference ratings for all seven manipulations indicate that, in general, our *automatic* images (mean: 2.2 range: 1.8-2.6) and *corrected* images (mean: 2.0 range: 1.8-2.3) match the *ground-truth* (mean: 1.7 range: 1.5-2) more closely than the *average* images (mean: 3.7 range: 2.9-4.7) (Figure 12 left). We find all of the differences in ratings across the four conditions to be significant

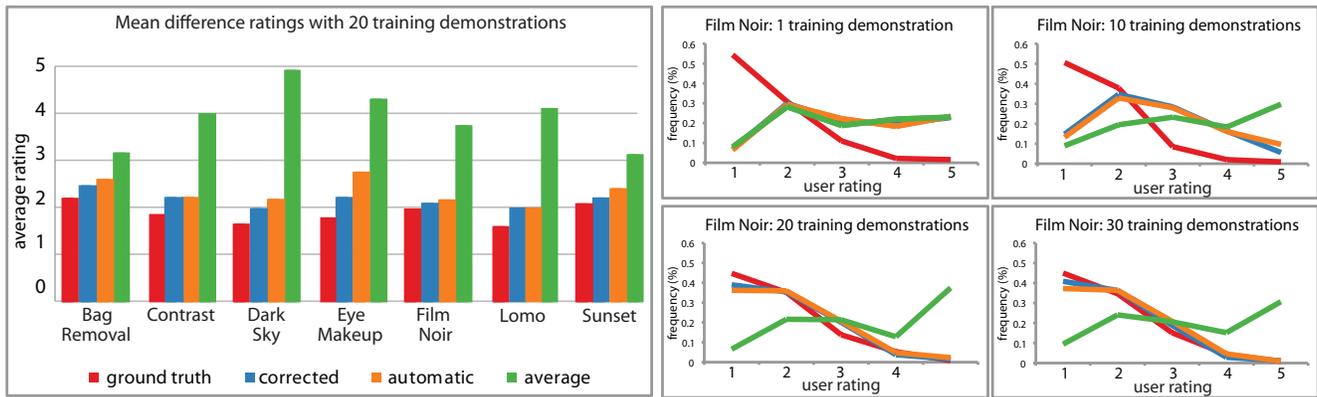


Fig. 12. (Left) The mean difference ratings from the Mechanical Turk experiments, where a low difference rating indicates greater similarity to ground truth. With 20 training demonstrations, our automatic and corrected macro results consistently received lower difference ratings than the average images and were close to ground-truth. (Right) The distribution of difference ratings for the film noir manipulation as the number of training demonstrations increases. With one demonstration, the distributions for automatic and corrected closely match the distribution for average. As the number of training demonstrations increase, the distributions for automatic and corrected shift to match that of the ground truth, while the average distribution remains unchanged.

( $p < 0.0001$ ) using Friedman’s nonparametric test for differences in ranks. Subsequent pairwise comparisons also find significant differences for all pairs of conditions ( $p < 0.0001$ ).

### 2. Macro results improve with up to 20 training examples

The difference ratings for our *automatic* and *corrected* results decrease as the number of training demonstrations increases. Furthermore, the most significant decrease in the difference ratings occurs at 20 training demonstrations for the two manipulations that primarily involve adjustment parameters – contrast and film noir (Figure 12 right). For the other manipulations, our results are already noticeably better than the *average* images after 1 demonstration and show little improvement after 10 demonstrations. This data (see supplemental PDF) suggests that 20 demonstrations are sufficient for learning adjustment parameters and that 10 demonstrations are enough to adapt most selection and brush stroke operations.

### 3. Corrected results are slightly better than automatic results

The automated labelers produced poor labels for 23% of faces and 31% of skies in our image datasets. However, the mean difference ratings for our *corrected* results are only slightly better (i.e., smaller) than the ratings for our fully *automatic* results; with 20 training demonstrations, the discrepancy between the *corrected* and *automatic* ratings ranges from 0 (for the contrast manipulation) to 0.53 (for eye makeup), with an average discrepancy of 0.18 across the six manipulations that required corrections (lomo did not require any corrections).

### 4. Macro results often indistinguishable from ground-truth

Because we include the *ground-truth* image as one of the images in the difference rating task, workers could rate our macro results as a closer match to *ground-truth*, than the *ground-truth* image itself. We count the number of times at least 3 out of 5 workers gave our result a rating less than or equal to their rating for the *ground-truth* image. In such cases it is likely that workers could not visually distinguish our images from *ground-truth*. We also count the number of images for which at least 3 out of 5 workers gave our result a rating greater than or equal to their rating for the average image. These images are the ones for which our approach performs poorly. With 20 training demonstrations, 47% of the automatic and 56% of the corrected images were rated better than or equal to *ground-truth* while just 6% of the *automatic* and 2% of the *corrected* were rated either no better or worse than *average*.

## 7.2 Macro Authoring Workflow

To evaluate the utility of our proposed macro authoring workflow, we conducted a small comparative lab study with three serious photographers who routinely use Photoshop to edit their images. We brought each participant in for a one hour session in which we first described our framework and then asked him to manipulate 5–6 images in Photoshop as if he was generating training demonstrations for our system (i.e., by performing the same sequence of manipulation steps for each image). We chose the lip gloss manipulation as our test case because it includes selections, brush strokes, and adjustment parameters (see Figure 7). Each participant performed the demonstrations under two different conditions: first, using Photoshop without our labeling and macro application panels, and then using Photoshop with our panels. Finally, we asked several questions to elicit feedback about our proposed workflow.

### Feedback

All three participants agreed that the panels were a clear improvement over the no-panel workflow. In particular, there was consensus that the macro application panel made the demonstration process much easier by enforcing the correct sequence of steps and automatically applying each step to new images, even if some of those steps required corrections. Participants specifically mentioned the visualization of steps and the ability to edit intermediate steps as important benefits. In addition, they appreciated that training occurs incrementally every time they apply the macro to a new image and correct the results. The feedback about the per-step macro robustness scores was more mixed. While some found the MSE numbers a bit difficult to interpret, most agreed that it was useful to have the system indicate which steps were likely to require corrections.

To get some feedback on the potential limitations of our workflow, we asked how participants felt about performing the same sequence of steps for each image and whether they would be willing to perform 15–20 demonstrations to train a content-adaptive macro. In response, none of the participants were bothered by having to perform the same sequence of steps (especially given the macro application panel), and two of three respondents said they would be willing to perform 15–20 training demonstrations to use our system — the third respondent felt that the number of required

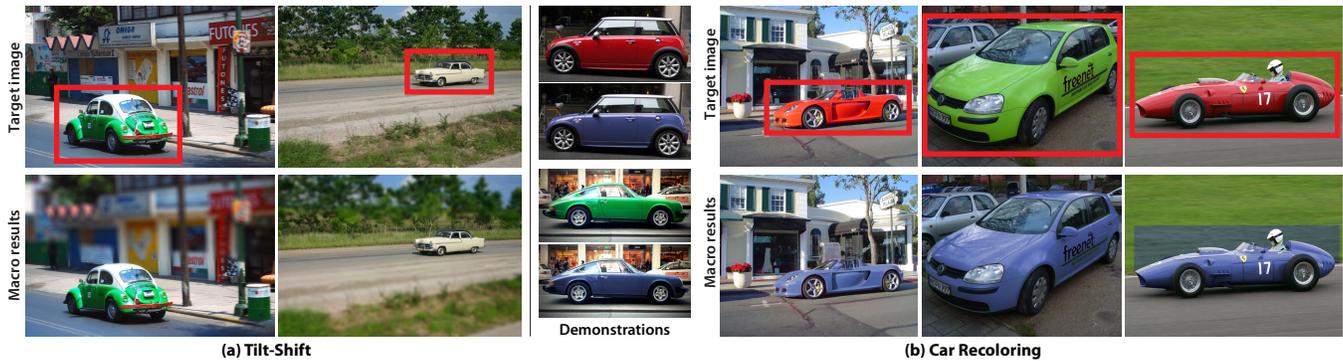


Fig. 13. Two car manipulations. Red boxes indicate cars found by Felzenszwalb et al.’s [2008] detector. (a) We demonstrate 10 tilt-shift manipulations and our framework successfully learns to blur the region above and below the car. (b) We demonstrate recoloring of 5 red cars and 5 green cars to blue. Our content-adaptive macro correctly recolors new target cars, without recoloring red/green elements (e.g. red flowers) that fall outside the car bounding box. But, it fails when red/green background elements, like grass, fall within the bounding box. Image credits: (a) *Martin Garcia, sliabh*, (b) *Axion23, C. Schroeder (vovchychko), John Nuttall*.

demonstrations was reasonable but explained that he does not typically perform the same manipulations on many different images.

Overall, the feedback from this study suggests that our proposed workflow is useful and that our labeling correction and macro application panels make it significantly easier for users to author content-adaptive macros.

## 8. EXTENSIONS TO OTHER IMAGE LABELERS

Although we have demonstrated our framework on common face, landscape and global manipulations, we have designed a general framework that can easily incorporate new image recognizers as they become available. To test this extensibility we have experimented with adding Felzenszwalb et al.’s [2008] car detector to our image labeler. This detector identifies a bounding box for each car and does not provide landmark correspondence points. We transfer two car-specific manipulations: a tilt-shift manipulation that leaves the car in focus while blurring its surroundings and a car recoloring manipulation. The tilt-shift manipulation (Figure 13a) performs very well for our target images because this manipulation only requires an approximate position and size for the car. It does not require a pixel-accurate boundary. The car recoloring manipulation (Figure 13b) performs well if the pixel-level features of the car are sufficiently discriminative. For example, if we train the macro to change red and green cars to blue it learns to recolor only pixels within the car bounding box and does not affect pixels outside the box (e.g. red flowers in first example). However, if there are red/green pixels within the bounding box that are not part of the car (e.g. grass in the third example) it recolors those pixels as well. More accurate landmark points on the boundary of the car would mitigate such problems. We leave it for future work to identify such landmark points when they are not directly provided by the labeler. We show additional examples of macro adaptation with and without accurate landmark points in supplemental material.

## 9. LIMITATIONS AND FUTURE WORK

While our framework is able to adapt many different types of photo manipulation macros to new target images, we have observed a few limitations that we plan to address in future work.

**Poor quality image labels.** Our approach relies on high quality image labeling with consistent landmark points. While our visual feedback and correction interfaces allow users to manually fix

incorrect labeling or landmarks, they also incur additional manual work. Moreover, many kinds of recognizers, including our outdoor scene recognizer, do not provide landmark points. In such cases, our approach of using the bounding box vertices as landmarks yields acceptable results for manipulations like sunset enhancement or tilt shift because they require coarse spatial information about the sky or car. But manipulations like car recoloring (Section 8) that require precise placement of selection regions or brush strokes are unlikely to transfer well using such landmarks.

**Indirect dependencies.** For some image manipulations the adjustment parameters depend on non-local image features. Our framework is only able to learn dependencies between adjustment parameters and image features of the active selection or brush region and its complement. For example, when applying the mustache macro to a light haired person, the result looks unnatural even though the location of the mustache is correct (see inset). In this case our framework is not able to learn the dependency between the color parameter of the mustache brush and the person’s hair color because the hair is not the main element in the brush region or its complement. An interactive machine learning approach where users guide the algorithm by specifying relevant features or image locations containing those features could mitigate such problems. Such an interactive extension could also reduce the number of demonstrations necessary to learn the macro, accelerate the learning process, and extend the number of dependencies the system could recognize.



**Unaligned demonstrations.** Our system requires that users perform all of the example demonstrations for a particular manipulation using roughly the same sequence of operations. While this operation alignment condition is satisfied for most demonstrations, some input images might require additional steps or the user might choose to demonstrate the steps in a different order. One approach may be to use sequence alignment or tree alignment techniques to automatically align operation sequences that differ significantly from one another. Using such alignment techniques, it may also be possible to combine demonstrations from multiple authors and thereby distribute the work of generating example demonstrations.

## 10. CONCLUSION

We have presented a framework for generating content-adaptive photo manipulation macros by demonstration. Our framework uses image labeling and machine learning to learn dependencies between image features and the location of selection regions, the paths of brush strokes, and the values of adjustment parameters. Using this framework, we demonstrate our system on 20 example manipulations and show that we can transfer complex manipulations with only 20 demonstrations. We also provide feedback and correction interfaces so that macro authors and users can find and correct errors introduced by poor labeling or poor parameter estimation. Finally, our evaluation shows that our framework can produce effective content-adaptive macros for a wide range of image manipulations and that our macro feedback and correction interfaces are both effective and practical in the context of real image editing workflows.

**Acknowledgments.** We thank Aaron Hertzmann and Sylvain Paris for valuable feedback on our learning algorithms. We also thank Marc Ducrest for sharing numerous photographs with us, as well as the Flickr users who placed their work under the Creative Commons License. This work was partially supported by National Science Foundation grant CFC-0643552.

## REFERENCES

- AMINI, A., CURWEN, R., AND GORE, J. 1996. Snakes and splines for tracking non-rigid heart motion. *ECCV'96*, 249–261.
- BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. *ACM Trans. Graph. (Proc. SIGGRAPH)* 25, 3, 637–645.
- BITOUK, D., KUMAR, N., DHILLON, S., BELHUMEUR, P., AND NAYAR, S. 2008. Face swapping: automatically replacing faces in photographs.
- BOLIN, M., WEBBER, M., RHA, P., WILSON, T., AND MILLER, R. C. 2005. Automation and customization of rendered web pages. In *Proc. UIST*. 163–172.
- CYPHER, A. AND HALBERT, D. 1993. *Watch What I Do: Programming by Demonstration*. MIT Press.
- DEWDNEY, A. 1989. A potpourri of programmed prose and prosody. *Scientific American*.
- DRORI, I., COHEN-OR, D., AND YESHURUN, H. 2003. Example-based style synthesis. In *In Proceedings of Computer Vision and Pattern Recognition (Madison, USA)*. 143–150.
- EFRON, B., HASTIE, T., JOHNSTONE, I., AND TIBSHIRANI, R. 2004. Least angle regression. *Annals of statistics*, 407–451.
- EFROS, A. AND FREEMAN, W. 2001. Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH*. 341–346.
- FELZENSZWALB, P., MCALLESTER, D., AND RAMANAN, D. 2008. A discriminatively trained, multiscale, deformable part model. In *Proc. CVPR*.
- GRABLER, F., AGRAWALA, M., LI, W., DONTCHEVA, M., AND IGARASHI, T. 2009. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph. (Proc. SIGGRAPH)* 28, 3, 66.
- GUO, D. AND SIM, T. 2009. Digital face makeup by example. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 73–79.
- HASINOFF, S., JÓZWIAK, M., DURAND, F., AND FREEMAN, W. 2010. Search-and-replace editing for personal photo collections. In *Proc. ICCP*. Number 2. 8.
- HERTZMANN, A., JACOBS, C., OLIVER, N., CURLESS, B., AND SALESIN, D. 2001. Image analogies. In *Proc. SIGGRAPH*. 327–340.
- HERTZMANN, A., OLIVER, N., CURLESS, B., AND SEITZ, S. 2002. Curve analogies. In *Proc. Eurographics Workshop on Rendering*. 233–246.
- HOIEM, D., EFROS, A., AND HEBERT, M. 2005. Geometric context from a single image. In *Proc. ICCV*. 654–661.
- HUGGINS, B. 2005. *Photoshop: Retouching Cookbook for Digital Photographers*. O'Reilly.
- JONES, M. AND REHG, J. 2002. Statistical color models with application to skin detection. *International Journal of Computer Vision* 46, 1, 81–96.
- KALNINS, R., MARKOSIAN, L., MEIER, B., KOWALSKI, M., LEE, J., DAVIDSON, P., WEBB, M., HUGHES, J., AND FINKELSTEIN, A. 2002. WYSIWYG NPR: Drawing strokes directly on 3D models. *ACM Transactions on Graphics* 21, 3, 755–762.
- KANG, S., KAPOOR, A., AND LISCHINSKI, D. 2010. Personalization of image enhancement. In *Proc. CVPR*.
- KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1988. Snakes: Active contour models. *International journal of computer vision* 1, 4, 321–331.
- KELBY, S. 2007. *The Adobe Photoshop CS3 book for digital photographers*. Voices That Matter.
- KURLANDER, D. AND FEINER, S. 1992. A history-based macro by example system. In *Proc. UIST*. 99–106.
- LAU, T., BERGMAN, L., CASTELLI, V., AND OBLINGER, D. 2004. Sheepdog: Learning procedures for technical support. In *Proc. IUI*. 109–116.
- LEWIS, D. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. *Machine Learning: ECML-98*, 4–15.
- LIEBERMAN, H. 1993. Mondrian: A teachable graphical editor. In *Watch what I do: Programming by demonstration*. 341–358.
- LIEBERMAN, H. 2001. *Your Wish is My Command: Giving Users the Power to Instruct their Software*. Morgan Kaufmann.
- LITTLE, G., LAU, T., CYPHER, A., LIN, J., HABER, E., AND KANDOGAN, E. 2007. Koala: Capture, share, automate, personalize business processes on the web. In *Proc. CHI*. 943–946.
- LIU, Z., SHAN, Y., AND ZHANG, Z. 2001. Expressive expression mapping with ratio images. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 276.
- MODUGNO, F. AND MYERS, B. 1994. Pursuit: Graphically representing programs in a demonstrational visual shell. In *Proc. CHI*. 455–456.
- NGUYEN, M., LALONDE, J., EFROS, A., AND DE LA TORRE, F. 2008. Image-based shaving. In *Computer Graphics Forum*. Vol. 27. Citeseer, 627–635.
- REINHARD, E., ASHIKHMINE, M., GOOCH, B., AND SHIRLEY, P. 2001. Color transfer between images. *IEEE Computer Graphics and Applications*, 34–41.
- SCHWARZ, D. 2005. Current research in concatenative sound synthesis. In *Proc. of ICMC*. 9–12.
- SIMMON, S. AND DUDEK, G. 2003. Curve Synthesis from Learned Refinement Models.
- ZHOU, Y., GU, L., AND ZHANG, H. 2003. Bayesian tangent shape model: Estimating shape and pose parameters via bayesian inference. In *Proc. CVPR*. 109–116.