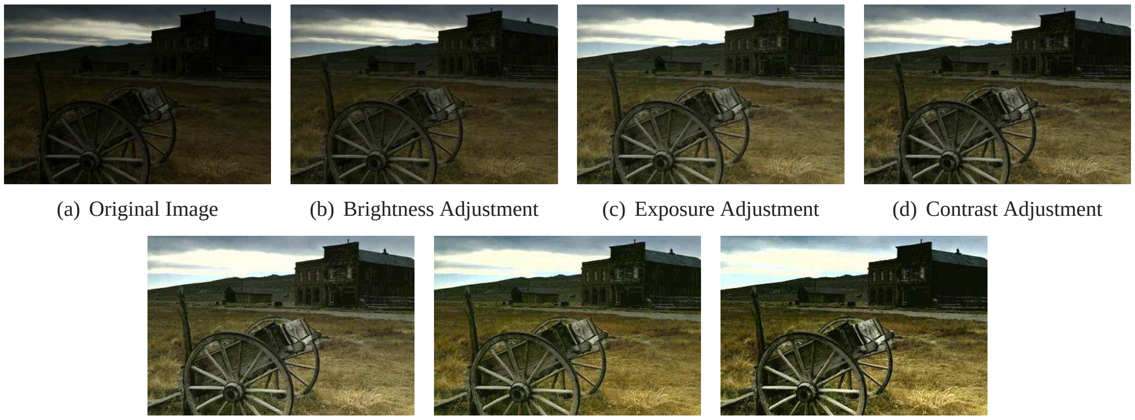
Iterative Learning: Leveraging the Computer as an On-Demand Expert Artist

Armin Samii[*]
University of California, Berkeley

Tim Althoff[†]
University of California, Berkeley



(a) Original Image (b) Brightness Adjustment (c) Exposure Adjustment (d) Contrast Adjustment (e) Brightness Adjustment (f) Saturation Adjustment (g) Black Clipping Adjustment

Figure 1: *Step-by-step suggestions are presented to show the user how to improve the photograph.*

Abstract

We develop a novel method to present a novice photographer with expert suggestions on how to improve a selected photograph. By modeling an expert's artistic decision process, we are able to predict the expert's decisions on a new photograph. We use an Iterative Learning Approach (ILA) in which we learn how to make a series of simple enhancements. First, we train a collection of independent regression models, each of which learns a single type of photograph adjustment (e.g. change in contrast). We then train a sequence model, which chooses the next adjustment to be made. This method allows the novice user to reason about each of the ILA's decisions. We show that our method lends itself to a friendly user interface to facilitate human understanding. Finally, we present examples showing that our method is comparable to recent work.

CR Categories: I.3.7 [Computer Graphics]: Applications—Photo Editing;

Keywords: Computational Photography, Machine Learning, User Interfaces

[*]e-mail:samii@eecs.berkeley.edu
[†]e-mail:althoff@eecs.berkeley.edu

1 Introduction

Applications such as Adobe Photoshop Lightroom[1] and darktable[2] provide users with many options to adjust a photograph after it is taken. For novice photographers, these options may be cryptic and it may be difficult to know which adjustments will improve an photograph. Even experienced photographers may have trouble deciding how to adjust a photograph, similar to "writer's block."

One option is to explore the various available adjustments until something acceptable is found, but this method is not efficient given the dozens of available choices. Another is to apply preset adjustments provided by the application developers, but the presets offered in most applications are not content-dependent and there are too many to efficiently explore. A preset which improves one photograph may look strange on another, even if the two photographs are similar. Further, the presets are not ordered in any meaningful way, so a good preset may be hard to find in the large list available.

With these problems in mind, researchers have developed automatic retouching tools. However, unlike with presets, these methods manipulate the pixels directly and return a compressed image, so the user cannot later modify the parameters without more loss of data ("invasive editing"). Further, these methods are a "black box" which do not provide a photographer with insight into how the final photograph was obtained.

We solve these shortcomings with an Iterative Learning Approach (ILA). This method presents the photographer with options to improve a given photograph. We choose a meaningful type of adjustment based on the image content and present the user with each step of our decision process. This gives the user the opportunity to learn about photograph enhancement in the process. Each adjustment we make is equivalent to one adjustment available in Adobe Lightroom, so they can intuitively reason about each decision and replicate it. An overview of the framework is displayed in Figure 2. One example sequence is shown in Figure 1.

[1]http://www.adobe.com/products/photoshoplightroom
[2]http://www.darktable.org

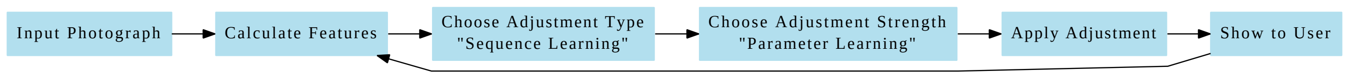

Input Photograph
Calculate Features
Choose Adjustment Type "Sequence Learning"
Choose Adjustment Strength "Parameter Learning"
Apply Adjustment
Show to User

Figure 2: *The Iterative Learning Approach. Given an input image, we repeatedly calculate the features, determine which adjustment to apply, and determine the strength of that adjustment. The user is shown the result of each iteration.*

We show that our results are similar to the adjustments made by expert photographers and comparable to "black-box" approaches.

2 Related Work

Our work can be segmented into two categories: image quality enhancement and machine learning algorithms. We present an overview of the literature in these fields.

2.1 Image Quality Enhancement

Many quality improvement methods require multiple images to fill in missing data, such as optimal relighting for showing detail [Akers et al. 2003; Moreno-Noguer et al. 2005], displaying images with high dynamic range [Fattal and Lischinski 2002; Durand and Dorsey 2002], and reducing noise in an image [Petschnigg et al. 2004]. These methods have good results, but having multiple images is not practical in many applications, including ours. We also aim to subjectively improve the aesthetics of a photograph (e.g. stylizing with false vignetting), instead of only objectively improving the quality of an image (e.g. noise reduction).

Berthouzoz *et. al.* develop a method for improving both the objective quality and subjective aesthetics of an image based on user-defined macros [Berthouzoz et al. 2011]. This method shows good results for applying complicated styles (such as adding snow) to a photograph. Unfortunately, each macro needs to be manually trained by an expert artist that uses their framework. There is also no automatic suggestion of types of macros - the user must decide which style to apply. Although our method is not able to handle complicated local adjustments, it requires no manual training data from the user and automatically decides a style to apply.

Current methods for fully automatic photograph enhancement focus on obtaining a better resulting image [Bychkovsky et al. 2011; Kang et al. 2010]. These approaches are black boxes: they do not allow a novice photographer using the tool to understand how the computer made its decision. Our approach arrives at a comparable resulting image while showing the user intermediate decisions, each of which can be reasoned about. Also, these methods only attempt to learn a few of the most common or useful adjustments, such as a tone curve adjustment and color correction. Our method supports generic operators, such as synthetic vignetting and tone splitting. These methods are also "invasive" in that they work on compressed 8 bit/channel images rather than RAW formats, which limits the amount of adjustments a user can make after obtaining the resulting image. Our method makes adjustments without any loss of data by maintaining a cumulative list of settings to apply onto the RAW file, so the user can adjust these settings after we present the ILA's result.

Because of the lack of training data previously available, learning aesthetic adjustments was difficult. Kang *et. al.* ask the user to make adjustments to a specific set of photographs to train their model [Kang et al. 2010]. To enhance a new photograph, they perform a nearst-neighbor search on the image features and copy the adjustment parameters. This is not appropriate because it does not adapt to the content in the image.

Bychkovsky *et. al* recently improved upon this work by creating a publically-available dataset containing 5000 photographs, each one with five corresponding adjustments by experienced photographers [Bychkovsky et al. 2011]. They are able to train a regression based on the differences in feature-space of the input and output image pairs. Unfortunately, the individual adjustments made by the photographers are not available, which motivates us to use our own dataset of adjustment-history metadata stored in Adobe Lightroom.

2.2 Machine Learning

Our approach uses statistical machine learning techniques to model the expert artist's knowledge and experience. A model based on n-grams [Brown et al. 1992; Chen and Goodman 1996] is used to predict a good adjustment sequence and their parameters are estimated by regression techniques.

Traditionally, sequence learning has been studied in computational linguistics computational biology, communication theory, and data compression. A well understood model in this context is the n-gram model [Brown et al. 1992]. In statistical natural language processing, an n-gram is a contiguous sequence of n items (e.g. phonemes, syllables, letters, words) from a given sequence of text or speech. An n-gram model is a probabilistic language model that can predict the next item in such a sequence based on statistical properties of n-grams. Mathematically, an n-gram models predicts x_i based on $x_{i-(n-1)}, \dots, x_{i-1}$. In probability terms, this is $P(x_i | x_{i-(n-1)}, \dots, x_{i-1})$. When used for language modeling, this is equivalent to the independence assumption that each word depends only on the last $n-1$ words. Therefore, n-gram models are $(n-1)$-order Markov models that approximate the true underlying language. This assumption is important because it simplifies the problem of learning the language model from data.

Naïve n-gram models assign a probability of zero to previously unseen n-grams. In practice, it is therefore necessary to smooth the probability distributions by also assigning non-zero probabilities to unseen words or n-grams. Several smoothing methods have been developed, from simple Lidstone smoothing [Lidstone 1920] to more sophisticated models, such as discounting models [Good 1953; Katz 1987; Ney et al. 1994] or back-off models [Kneser and Ney 1995]. Some of these methods are equivalent to assigning a prior distribution to the probabilities of the n-grams and using Bayesian inference to compute the resulting posterior n-gram probabilities [Chen and Goodman 1996].

Popular choices for regression techniques include Ordinary Least-Squares regression (OLS) [Fu 1998], Ridge regression [Hoerl and Kennard 1970a; Hoerl and Kennard 1970b], Lasso [Tibshirani 1996], Elastic Net [Zou and Hastie 2005], Least Angle regression (LARS) [Efron et al. 2004], and Gaussian Process regression (GPR) [Williams and Rasmussen 1996; Rasmussen 2004]. These techniques are presented in more detail in Section 5.2.

3 Training Data

Adobe Lightroom maintains the history of adjustments applied to every photograph. Our code to extract this metadata is available on our project website[3]. We make the assumption that every adjustment sequence in the photographer's Lightroom metadata improves the image. We explain the benefits of using each adjustment instead of only looking at the original and final image.

While learning the strength of each adjustment, we treat each step of the history as an independent training example. This helps the training data in three ways: ease of access, quantity, and quality. Note that these benefits are applicable specifically to *parameter learning*, not sequence learning (although it is equally easy to obtain sequence learning data).

1. **Easy to Obtain:** Theoretically, we would be able to install our feature extracter as a Adobe Lightroom Plugin on any expert photographer's computer. Without any added effort on their part, we would extract all the features we need and add it to a master training set.

2. **Quantity:** We treat every photograph as an independent set of single adjustments. A photograph with ten adjustments is treated as ten separate photographs, each with a simple adjustment to learn. The caveat here is that we have independent models for each type of adjustment, and a single photograph is unlikely to have a data point for each of them. Still, because of the amount of data we can obtain, we will have sufficient data to train each model.

3. **Quality:** Each of our learning algorithms model a relatively simple function compared to only using the initial and final image, because this function is a single adjustment instead of a complicated sequence of adjustments. Further, the most-used adjustments (such as exposure correction) are trained very accurately because of the amount of available data available.

The dataset we use is obtained from a single photographer. There are 2237 adjustments across 572 photographs. This means that sequence learning (Section 5.1) contains 572 training examples, and all of the paramater learning models combined contain 2237 training examples. More common adjustments have more training examples. For example, exposure has 544 examples, contrast adjustment has 169 examples, and adding fill light had 80 examples. We ignore adjustments with fewer than 20 examples. For now, we also ignore any local adjustments or multi-parameter adjustments (such as a gradient filter or cropping), though it is planned for future work.

4 Features

The ILA requires features to be recalculated at every iteration. Doing so on the full-size image would severely limit the number of features that can be efficiently calculated. We thus choose features that can be calculated on the image thumbnail; fine-scale structure is ignored. We calculate the following features at each iteration:

- **Histograms:** Experimentation has shown that the best set of channels to include in our feature vector are the Hue and Saturation channels in HSV space and the Luminosity channel in CIE LAB space. We split these three channels into 15 bins each.

- **Clipped values:** The percentage of fully overexposed and underexposed pixels.

- **Average luminosity:** The mean luminosity value in CIE LAB space.

- **Global Contrast:** The global contrast using the root mean square contrast metric [Peli 1990]:

$$\sqrt{\frac{1}{MN}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(I_{ij}-\bar{I})^2}$$

  where M and N specify the width and height of the image, I_{ij} is the CIE LAB luminosity of pixel (i, j), and $\bar{I}$ is the average luminosity.

- **Face Detection:** The number of faces present and the percentage of the area that contains a face. These two features theoretically should distinguish between portraits and group shots. We use a set of Haar-like features from the OpenCV library [Bradski 2000] and sum the area of the resulting bounding boxes. Note that this does not need to be recalculated at each iteration, and thus can be calculated on the full-size image.

5 Iterative Learning

In our novel approach, we model the expert artist's knowledge and experience using well-known techniques from statistical machine learning. As motivated in the introduction, we want to predict a good adjustment sequence and their parameters based on the adjustment history and current image features. We solve this problem in two independent steps: first, we predict the next adjustment (sequence learning), and second, given this adjustment we predict the optimal parameter for it (parameter learning).

5.1 Sequence Learning

Intuitively, we are predicting the next adjustment by looking at past adjustments and the current content of the photograph. For example, based on the experience gained from the expert artist (that we reflect in our model) we would usually change contrast after adjusting exposure, but given that a particular photograph already has high contrast we may want to add a increase saturation instead.

Much like previous work used n-gram models for language modeling, i.e. how sentences are modeled by sequences of words, we use them to learn the "language" of expert artists, i.e. how styles are modeled by sequences of photographic adjustments. To do this, it is necessary to extend the $(n-1)$-order Markov model to satisfy our requirements. If we compute an n-gram model based on a corpus provided by an expert artist, we would always suggest the same adjustments for the photograph, irrespective of its content.

However, we want to take the content of the photograph into account, e.g. decreasing exposure for an overexposed photograph but increasing contrast for a correctly exposed but low-contrast photograph. For these reasons, we developed a novel model that include the current image features of the photograph. We call this a feature-augmented n-gram model. Graphically, the model is specified in Figure 3, where $A_1, \ldots, A_N$ is the adjustment sequence and $F_1, \ldots, F_N$ are the image features. In this notation, each F_i is the image feature vector before applying adjustment A_i. The presented model is based on a bigram model (i.e. a first-order Markov model). However, the following derivation can easily be adapted to higher order models as indicated by the dashed arrows in Figure 3.

5.1.1 Mathematical Derivation

Note that Figure 3 is a completely observed graphical model with the factorization

$$p(\{A_i\},\{F_i\}) = p(A_1)\cdot\prod_{i=2}^{N}p(A_i|A_{i-1})\cdot\prod_{i=1}^{N}p(F_i|A_i)$$

[3]http://www.artoonie.com/projects/ila/lightroom-parse.html

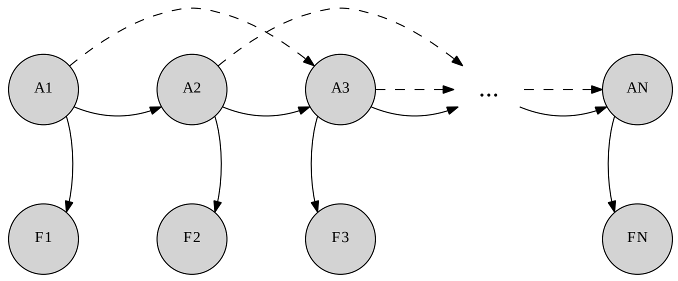


Figure 3: *Graphical model for sequence learning, where $A_1, \ldots, A_N$ is the adjustment sequence and $F_1, \ldots, F_N$ are the image features. The solid arrows describe the first-order Markov model (based on bigram model) and including the dashed arrow leads to a second-order Markov model (based on trigram model).*

where multinomial random variables A_i are the adjustments, continuous random variables F_i are the image features, and A_i^k are indicator variables capturing the state of the multinomial random variable ($A_i^k = 1$ if $A_i = k$ and $A_i^k = 0$ otherwise). We use the following definition for our parameters:

$$\begin{aligned} p(A_1 = k) &\triangleq \pi_k = \prod_{k=1}^{K} \pi_k^{A_1^k} \\ p(A_i = k | A_{i-1} = l) &\triangleq a_{kl} = \prod_{k=1}^{K}\prod_{l=1}^{K} a_{kl}^{A_i^k A_{i-1}^l} \\ p(F_i | A_i = k) &\triangleq \mathcal{N}(F_i; \mu_k, \Sigma_k) = \prod_{k=1}^{K} \mathcal{N}(F_i; \mu_k, \Sigma_k)^{A_i^k} \end{aligned}$$

where $\sum_k \pi_k = 1$, $\sum_k a_{kl} = 1$, and $\mathcal{N}(\mu_k, \Sigma_k)$ is a multivariate Gaussian density with parameters μ_k and Σ_k. Now, we can write our factorization above as

$$\prod_{k=1}^{K} \pi_k^{A_1^k} \cdot \prod_{i=2}^{N}\prod_{k=1}^{K}\prod_{l=1}^{K} a_{kl}^{A_i^k A_{i-1}^l} \cdot \prod_{i=1}^{N}\prod_{k=1}^{K} \mathcal{N}(F_i; \mu_k, \Sigma_k)^{A_i^k}.$$

Let us assume we have S iid samples $(\{A_{i_s}\}, \{F_{i_s}\}$ for $i = 1, \ldots, N$ and $s = 1, \ldots, S)$. Then the loglikelihood of our data is

$$\begin{aligned} l(D) &= \log p(\{A_{1_s}\}, \ldots, \{A_{n_s}\}, \{F_{1_s}\}, \ldots, \{F_{n_s}\}) \\ &= \sum_{s=1}^{S} \log p(\{A_{i_s}\}, \{F_{i_s}\}) \\ &= \sum_{s=1}^{S} \Big[\sum_{k=1}^{K} A_1^k \log \pi_k + \sum_{i=2}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K} A_i^k A_{i-1}^l \log a_{kl} \\ &\quad + \sum_{i=1}^{N}\sum_{k=1}^{K} A_i^k \log \mathcal{N}(F_i; \mu_k, \Sigma_k) \Big] \end{aligned}$$

In this form, the Maximum Likelihood estimates can be obtained by

$$\begin{aligned} \pi_k^* &= \frac{\sum_{s=1}^{S} A_{1_s}^k}{\sum_{k=1}^{K}\sum_{s=1}^{S} A_{1_s}^k} \\ a_{kl}^* &= \frac{\sum_{i=2}^{N}\sum_{s=1}^{S} A_{i_s}^k A_{i-1_s}^l}{\sum_{k=1}^{K}\sum_{i=2}^{N}\sum_{s=1}^{S} A_{i_s}^k A_{i-1_s}^l} \\ \mu_k^* &= \frac{\sum_{i=1}^{N}\sum_{s=1}^{S} A_{i_s}^k F_{i_s}}{\sum_{i=1}^{N}\sum_{s=1}^{S} A_{i_s}^k} \\ \Sigma_k^* &= \frac{\sum_{i=1}^{N}\sum_{s=1}^{S} A_{i_s}^k (F_{i_s} - \mu_k^*)(F_{i_s} - \mu_k^*)^T}{\sum_{i=1}^{N}\sum_{s=1}^{S} A_{i_s}^k}. \end{aligned}$$

If we take a closer look at a_{kl}^*, we can see how they relate to the concept of n-grams (with $n = 2$ in this case):

$$\begin{aligned} a_{kl}^* &= \frac{\sum_{i=2}^{N}\sum_{s=1}^{S} A_{i_s}^k A_{i-1_s}^l}{\sum_{k=1}^{K}\sum_{i=2}^{N}\sum_{s=1}^{S} A_{i_s}^k A_{i-1_s}^l} \\ &= \frac{\#(A_i = k \wedge A_{i-1} = l)}{\#(A_{i-1} = l)} \\ &= \hat{p}(A_i = k | A_{i-1} = l) \end{aligned}$$

which is the empirical probability of the n-gram model based on n-gram counts. Note that the presented derivation can easily be adapted to higher order n-gram models, as well as adapted to emission probabilities $p(F_i | A_i = k)$ that are not Gaussian but, for example, mixtures of Gaussians. However, in practice one needs to be careful with more complex models if the amount of data provided is comparably small because of overfitting. The performance of the presented model is evaluated in Section 7.1.

5.2 Parameter Learning

We want to predict the optimal parameter for a given adjustment based on image features (e.g. the amount by which we want to increase exposure). Mathematically, this is a simple regression problem for which regression techniques like Ordinary Least-Squares regression (OLS) can be used. However, based on the features described in Section 4, for each adjustment, only a subset of our features is correlated with the adjustment strength. For example, to find the right exposure it could suffice to only look at the luminosity histogram whereas for changing vibrance, the color histogram would be important instead. For this reason, we use regression methods that, in addition to fitting a reasonable linear model, assume some sparsity of the parameter vector β. Here, sparsity refers to a lot of components being small or even zero which correspond to the feature selection task we described above. In the following, we go into more detail of the regression task at hand and describe techniques that lead to sparse linear models. We also explored non-linear models such as neural networks, Gaussian process regression, and support vector regression. In our experiments, while being computationally more expensive, we did not observer any improvement in performance.

5.2.1 Relevant Regression Techniques

Mathematically, our parametere learning task is a simple regression problem where the dependent variable y is the parameter that we are predicting (the response) and $x_i = (x_{i1}, \ldots, x_{ip}), i = 1, \ldots, n$ are the p-dimensional feature vectors. In linear regression, data are modeled using linear functions, and unknown model parameters are estimated from the data. In particular, the linear regression problem can be written as $y = X\beta + \epsilon$, where y is an n-vector of random responses, X an $n \times p$ design matrix containing the n stacked feature vectors, β a p-vector of parameters or weights, and ϵ an n-vector of iid random error [Fu 1998].

Ordinary Least-Squares regression (OLS) minimizes the Residual Sum of Squares (RSS) between the observed responses in the dataset, and the responses predicted by the linear approximation

$$\hat{\beta} = \min_{\beta} \|y - X\beta\|_2^2$$

which yields an unbiased estimator

$$\hat{\beta} = (X'X)^{-1}X'y.$$

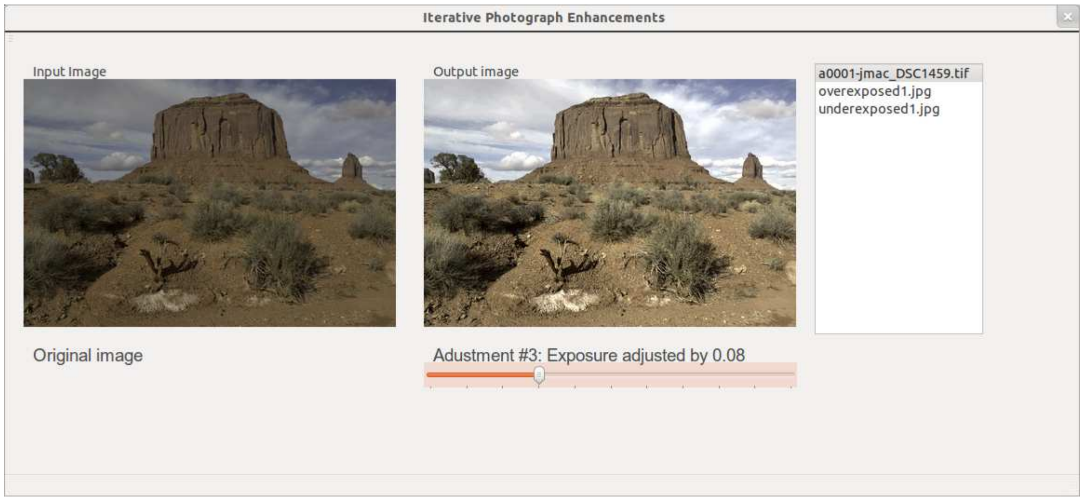


Figure 4: *The current interface. As the user moves the slider to the right, successive iterations are displayed. The text above the slider describes the adjustment that brought the user from the previous state to the one shown.*

one key assumption is that the design matrix X must have full column rank p. For this property to hold it is necessary (but not sufficient) that $n > p$. If this condition fails this is called the multicollinearity in the regressors. Methods for fitting linear models with multicollinearity have been developed [Tibshirani 1996; Hoerl and Kennard 1970a; Efron et al. 2004; Zou and Hastie 2005] based on additional assumptions such as sparsity of the parameter vector. Detailed discussions can be found in [Hoerl and Kennard 1970a; Hoerl and Kennard 1970b]. We use these techniques because we expect the adjustment parameter to depend only on a small subset of features such that sparsity constraints should help in finding the right paramaters. The rest of this section presents some of these techniques in more detail.

ridge regression [Hoerl and Kennard 1970a] imposes a penalty on the size of coefficients, i.e. the ridge coefficients minimize a penalized residual sum of squares:

$$\hat{\beta} = \min_{\beta} \|y - X\beta\|_2^2 + \alpha\|\beta\|_2^2.$$

here, α is a complexity parameter that controls the L_2-regularization. The larger the value of α the greater the amount of "shrinkage" and thus the coefficients become more robust to collinearity.

the Lasso is a linear model that estimates sparse coefficients [Tibshirani 1996]. It is useful in some contexts due to its tendency to prefer solutions with fewer parameter values, effectively reducing the number of variables upon which the given solution is dependent. Mathematically, it minimizes the residual sum of squares with L_1-regularization:

$$\hat{\beta} = \min_{\beta} \|y - X\beta\|_2^2 + \alpha\|\beta\|_1$$

with α controlling the amount of regularization again.

although the Lasso has shown success in many situations, it has been empirically observed that the prediction performance of the Lasso is dominated by ridge regression if there are high correlations between predictors [Tibshirani 1996]. Therefore, the Elastic Net combines the L_2 penalty of ridge regression with the L_1 penalty of the Lasso that essentially does variable selection and produces sparse represenations [Zou and Hastie 2005]. The objective function for the Elastic Net is:

$$\hat{\beta} = \min_{\beta} \|y - X\beta\|_2^2 + \alpha\|\beta\|_2^2 + \gamma\|\beta\|_1.$$

least Angle Regression (LARS) [Efron et al. 2004] is a forward selection algorithm or a "forward stepwise regression". It was shown that a simple modification of this variant of linear regression implements the Lasso in a very efficient way.

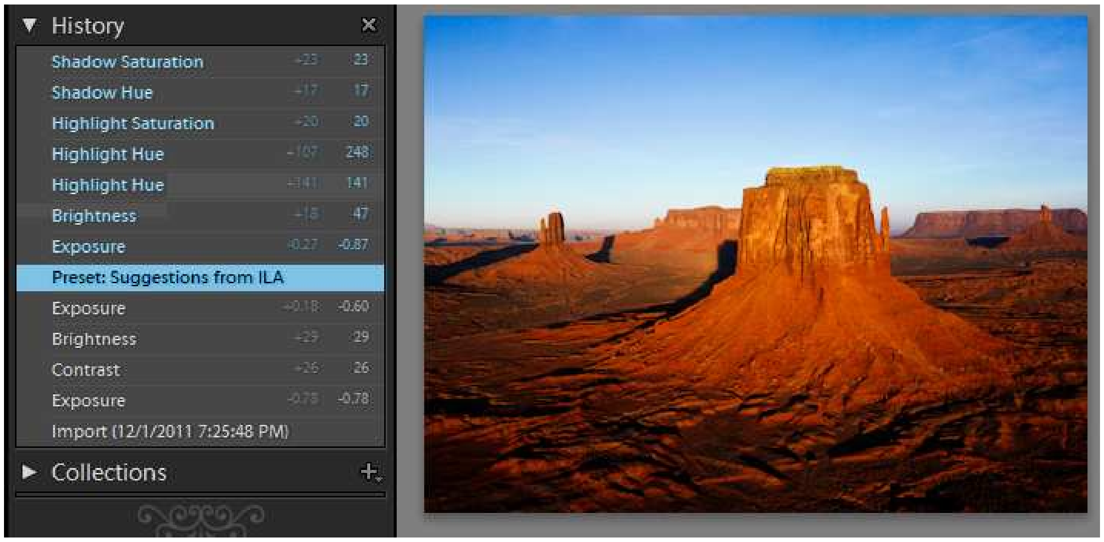
we also explored Gaussian Process Regression (GPR) [Williams and Rasmussen 1996; Rasmussen 2004] which is a powerful but computationally more expensive method where the prediction interpolates the observations, and the prediction is probabilistic (Gaussian) so that one can compute empirical confidence intervals that can be used to refit the prediction in regions of interest.

related work uses a subset of these regression techniques, namely OLS, Lasso, GPR, and LARS [Bychkovsky et al. 2011; Berthouzoz et al. 2011]. We compare the prediction performance of the presented regression techniques in Section 7.2.

6 User Interface

Our goal is to create an interface that allows the user to "ask an expert" for advice. Since our machine learning models have been trained on data extracted from expert photographer's Adobe Lightroom image metadata, showing the decisions of our algorithm models what an expert would advise. Although it is not our goal, this approach also allows a user to accept the ILA's final result without viewing the history. Further, it allows users to explore various adjustment options for creative inspiration, similar to Design Galleries [Marks et al. 1997]. This allows our approach to be used as both an automatic retouching tool and a tool to aid the creative process.

Current interface Through our interface (Figure 4), a novice photographer can get assistance manipulating an image by asking our trained model of an expert artist for help. Our current interface allows users to view the adjustments made at each iteration. The user can select an image and view the adjustment made at each timestep. If the result is acceptable, the user can apply the same adjustment in Lightroom.



```
History
Shadow Saturation          +23     23
Shadow Hue                 +17     17
Highlight Saturation       +20     20
Highlight Hue             +107    248
Highlight Hue             +141    141
Brightness                 +18     47
Exposure                 -0.27  -0.87
Preset: Suggestions from ILA
Exposure                 +0.18  -0.60
Brightness                 +29     29
Contrast                   +26     26
Exposure                 -0.78  -0.78
Import (12/1/2011 7:25:48 PM)
Collections
```

Figure 5: *The planned interface. The History window shows the series of adjustments in chronological order, from bottom to top. After the user chooses to get suggestions from the ILA (shown with a blue background in the History window), a series of new adjustments will be displayed (typeset in blue). The user can then click through each adjustment to see what was changed, or accept the final result and ignore the intermediate steps.*

Planned interface Although the current interface successfully shows the user each decision of the ILA, it does not integrate well with a photographer's workflow. In light of this, we propose an interface with the same benefits but also integrated with Adobe Lightroom (see Figure 5). This interface will display suggested adjustments in the History window of Adobe Lightroom. When the user selects an option to get suggestions from the ILA, a series of suggested adjustments is displayed. From there, the user can see how each iteration affects the image.

7 Results

In this section, we present an evaluation of the proposed iterative learning approach. We provide empirical results for sequence and parameter learning as well as qualitative results. Because Lightroom does not have an interface for using their adjustments, and they do not advertise the algorithms they use for each type of adjustment, we have implemented our best approximation of them. Since they are not equivalent to Lightroom's adjustment, our feature vectors are not fully accurate. For example, the image we generate after the adjustment ”Exposure +1” is different than the one Lightroom generates, so the feature vector is slightly different. Therefore, until we can use the same algorithm that Lightroom uses, our results on real-world data are accompanied with proof-of-concept results on synthetic datasets.

7.1 Sequence Learning

To evaluate sequence learning, we used several adjustment sequences $A_1, \dots, A_N$ from an expert artist and tried to predict the next adjustment given previous adjustment steps and current image features. Note that the adjustments do not always have to be in the same order as given by ground truth to produce good results. If we predict an adjustment $\hat{A}_i$ but the artist did another adjustment $A_i \neq \hat{A}_i$ then we still count $\hat{A}_i$ as a true positives if the artist applied this adjustment at some later point in his adjustment sequence $A_{i+1}, \dots, A_N$. Under this constraint, we measure the prediction accuracy for varying “lookahead” l (i.e. looking whether $\hat{A}_i$ appears in $A_i, \dots, A_{i+l}$). Formally, we define accuracy as:

$$ACC(l) = \frac{1}{N} \sum_{i=1}^{N} \lfloor \hat{A}_i \in \{A_i, \dots, A_{i+l}\} \rfloor$$

Under this definition, $ACC(0)$ corresponds to the fraction of predictions that are correctly made at the same point in the sequence as in the ground truth.

7.1.1 Proof of Concept

To evaluate our feature-augmented n-gram model as well, we created an synthetic dataset containing 1000 adjustment sequences (750 for training, 250 for testing) using the scitkit-learn package for machine learning [Pedregosa et al. 2011]. We tried to replicate properties that we expect from our real dataset, i.e. we have 20 different adjustment possibilities and 20-dimensional features that are modelled by “overlapping” multivariate Gaussians with a large standard deviations (we assume a very cluttered feature space). The adjustment sequences themselves were sampled from a first-order Markov model with random transition parameters. In this case, we compare our proposed feature-augmented n-gram model (for $n = 2, 3$ to different baselines, i.e. only using features to predict the next adjustment, using only the n-gram models (without feature-dependence), and a random baseline as before. We compare these methods for different lookaheads by averaging the accuracy metric defined above over the 250 adjustment sequences for testing. The other 750 sequences were used to estimate the parameters described in Section 5.1.

The results for this dataset are shown in Figure 6. Our proposed model outperforms all baselines showing that it can be beneficial to predict the next adjustment based on the adjustment history as well as current image features. Given that this evaluation was performed on synthetic data, although the results look very promising, it does not readily prove a good performance of our model on the real world dataset.

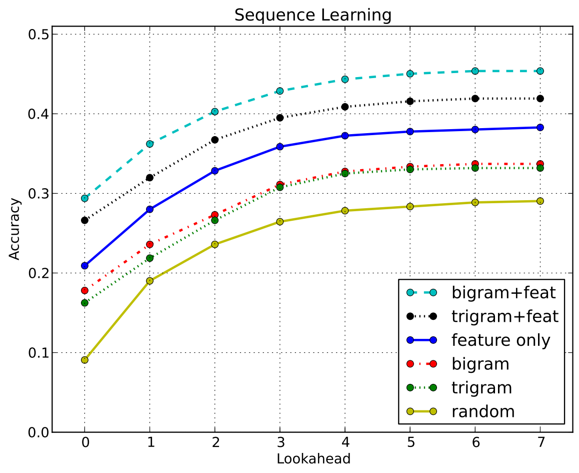


Figure 6: *Proof of concept for our feature-augmented n-gram model. Our proposed model outperforms all baselines: only using features to predict the next adjustment, using only the n-gram models (without feature-dependence), and a random baseline.*

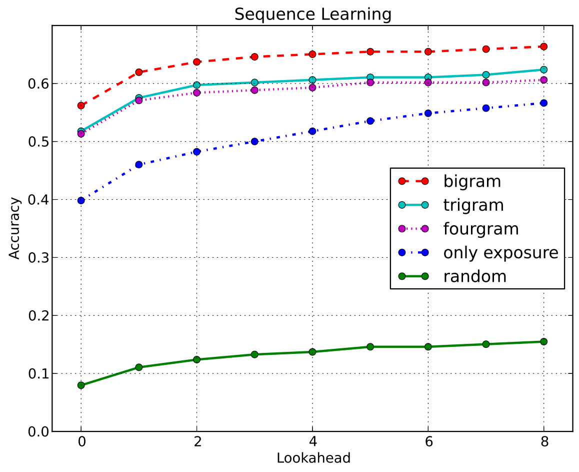


Figure 7: *Results for sequence learning on the real-world dataset. The bi- and trigram model consistently outperform the "always exposure" baseline as well as the random baseline.*

7.1.2 Real-world Data

Our real-world dataset contrains 270 adjustment sequences for training and 100 sequences for testing. We compare our non-feature-dependent n-gram models to two baselines, one always predicting exposure adjustments (because this was the most common adjustment in our dataset), and another one doing random predictions (uniformly over all possible adjustments. The results can be found in Figure 7. The best n-gram model (for $n = 2$) consistently outperforms the exposure-baseline as well as the random baseline. For a lookahead of zero, the difference is about 16% accuracy.

7.2 Parameter Learning

We evaluated parameter learning using two common metrics in regression analysis, the mean squared error (MSE) and the coefficient of determination r^2 [Menard 2000]. MSE represents the amount by which the predicted values differ from the quantities being estimated. Formally, it is the value of the squared deviations of the predictions $\hat{y}_i$ from the true values y_i (usually over an out-of-sample test space):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

In the case of linear regression, the coefficient of determination r^2 is the square of the sample correlation coefficient between the outcomes and their predicted values. Also, its value can be seen as the amount of variance explained by the data. Let again y_i be the values to be predicted, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ their sample mean, and $\hat{y}_i$ the modelled or predicted values. Then, the coefficient of determination is defined by:

$$\begin{aligned} SS_{tot} &= \sum_i (y_i - \bar{y})^2, \\ SS_{err} &= \sum_i (y_i - \hat{y}_i)^2, \\ r^2 &= 1 - \frac{SS_{err}}{SS_{tot}}, \end{aligned}$$

where SS_{tot} is the total sum of squares (proportional to the sample variance), and SS_{err} the residual sum of squares.

7.2.1 Proof of Concept

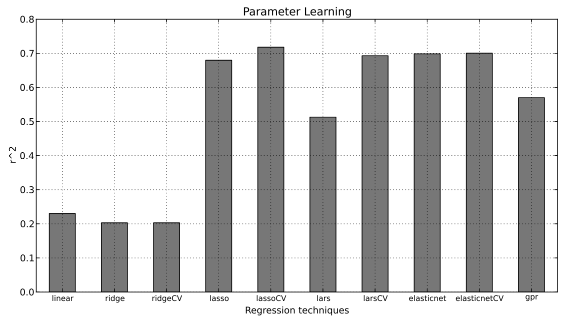
To prove the concept of sparse linear models for regression, we created and evaluated on an synthetic dataset that has higher dimensionality than our current features but a and a low effective rank (as motivated in Section 5.2). We used the scitkit-learn package for machine learning [Pedregosa et al. 2011] to create a synthetic dataset that resembles some of the properties we expect our data to have. The dataset has 200 samples and 200 features. However, the effective rank (the approximate number of singular vectors required to explain most of the input data by linear combinations) is only 20. This sparsity makes sense because we expect that for any given adjustment, the adjustment strength only depends on a small subset of features. Furthermore, we applied Gaussian noise with a standard deviation of 3 to the output to create a more challenging regression problem. The results from comparing all presented regression techniques on this dataset can be found in Figure 8. Clearly, methods that enforce sparsity on the parameter vector β, like Lasso (L_1-penalty) outperform the rest. We expect these methods to work even better on the real world data once we are able to transform the image using Adobe Lighroom's adjustments (as noted in the beginning of this section).

7.2.2 Real-world data

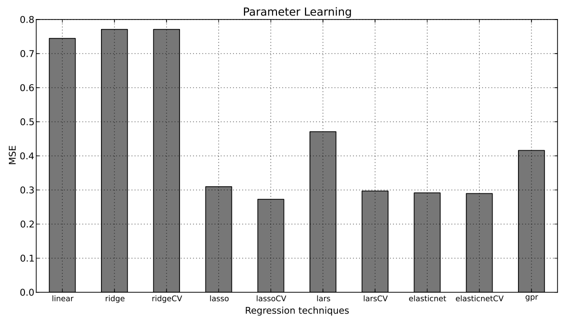
The results on the real-world dataset are shown in Figure 9. Observe that all regression techniques perform similarly. The techniques involving a regularization of the parameter vector β do not have an advantage here because the feature dimension is rather small (51). Because we plan to use a lot more features in the future, we expect the regression techniques creating sparse linear models to eventually outperform simpler techniques on real-world datasets as well.

7.3 Qualitative results

Figure 10 shows successful example outputs of our method. Each of the examples displayed are chosen for illustration purposes from a sequence of five to fifteen adjustments. The first image (Figures 10(a)-10(c)) slowly increases contrast and decreases exposure,

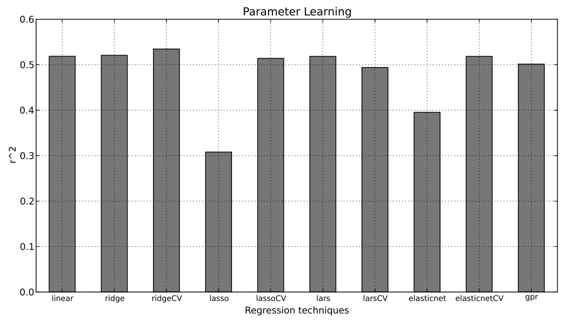


(a) r^2 error on the synthetic regression dataset

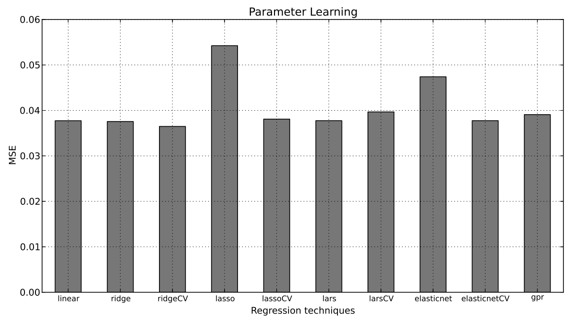


(b) Mean squared error (MSE) on the synthetic regression dataset

Figure 8: *Proof of concept for our sparse regression techniques on synthetic data. The suffix "CV" indicates that the respective parameter was selected using cross validation. Clearly, methods that enforce sparsity on the parameter vector β, like Lasso (L_1-penalty) outperform the rest.*



(a) r^2 error on the real-world regression dataset



(b) Mean squared error (MSE) on the real-world regression dataset

Figure 9: *Results on our regression dataset. The suffix "CV" indicates that the respective parameter was selected using cross validation. One can observe that all regression techniques perform about the same. The techniques involving a regularization of the parameter vector β do not have an advantage here because the feature dimension is rather small (51).*

along with other minor adjustments in between. The second image (Figures 10(d)-10(f)) iterates several times, slowly decreasing exposure and brightness, before adjusting the black level clipping and saturation, which greatly improves the aesthetics.

Figure 11 shows some failure cases for our system. Both images have lost data through either clipped highlights or clipped shadows. The first image (Figures 11(a)-11(c)) oscillates between the original image and a faded image. Each iteration attempts to fix the mistakes of the previous, but is unable to, and thus our system loops through the same set of adjustments. The second image is overexposed in most of the image, so our system attempts to correct this by adjusting the exposure. However, this only affects the region of the image with detail, and thus each iteration removes more detail from the bridge until it has fully deteriorated. We expect that supporting local edits will alleviate this problem (see Section 8).

Furthermore, sometimes underexposed photographs are intentionally dark, such as as in low-key photography. We plan to identify these photographs by unsupervised style modeling as described in Section 8.

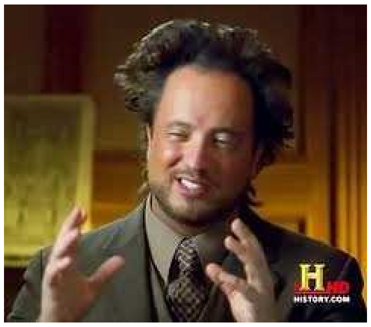
We believe these failure cases occur because the training data is unfamiliar with the adjustments that would be needed for these photos. Further analysis showed that the quality of photos in the training corpus provided by the expert photographer was generelly much better than these example images. We expect that a larger training corpus will at least partially remedy this problem.

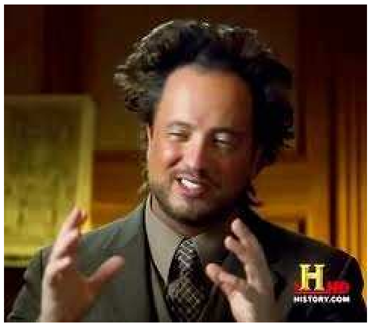
8 Future Work

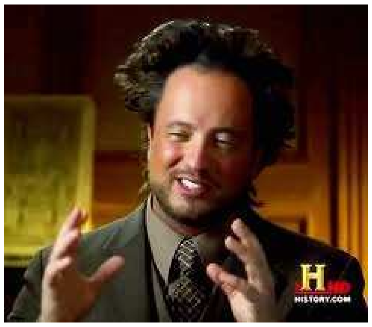
We have just touched the surface in the field of Iterative Learning. There are many potential directions that this research can go. We present some of these below.

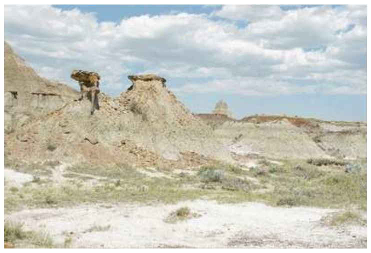
Style Modeling Through Clustering We need a way to group entire sequences together so that a training example that applies one style (such as black-and-white) does not mix with one that applies a different style (such as highly-saturated). Our current method treats all sequences of adjustments the same, and thus averages together the various styles. This often leads to errors such as iteratively lowering saturation to near-grayscale, and a few iterations later, increasing it to normal. One method is to take the difference in feature-space between the original and final images in the training set. This will result in a difference-feature-vector which describes how the image was modified overall. We would then cluster data points in this difference-feature-space, and would train independent models on each cluster.

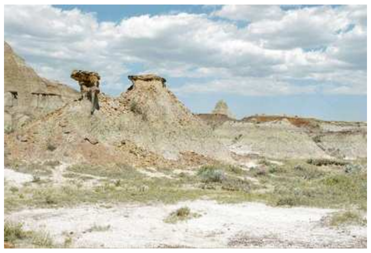
Cleaning Training Data The above clustering approach assumes that each iteration in the training data has the same style in mind, and each step gets closer to the final product. This is not the case in real data. To partially remedy this, we would have to remove any "loops" in the training data. A loop is a sequence of adjustments, $A_1, ..., A_N$, such that the feature vector before applying adjustment A_1 is similar to the feature vector after applying A_N. Applying this loop does not get us any closer to the desired result.


(a) Original Image


(b) Intermediate Result


(c) Final Result


(d) Original Image


(e) Intermediate Result

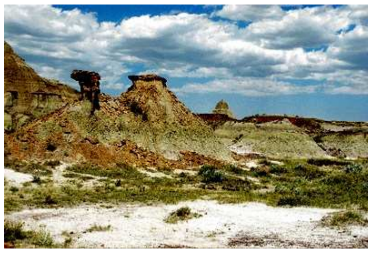
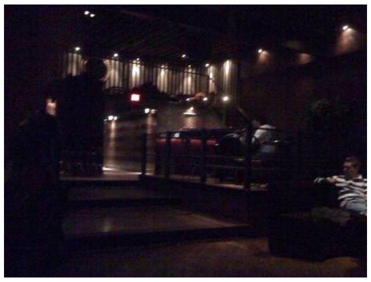
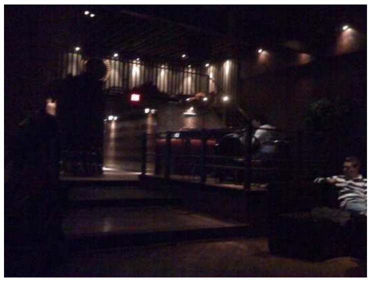
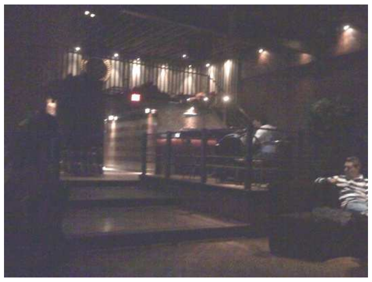
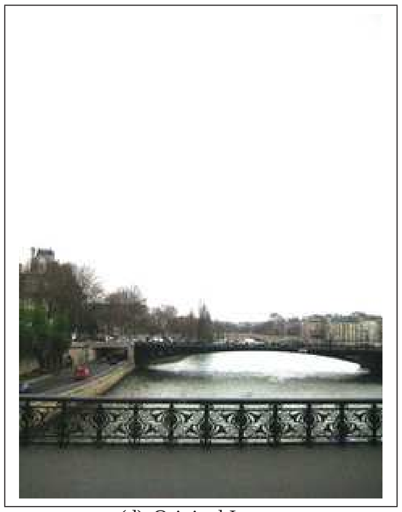

(f) Final Result

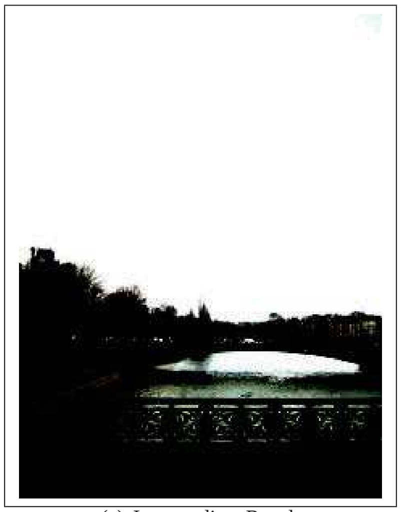
Figure 10: *Some example results of our method. The first photograph in each row is the input. The second photograph is an intermediate result. The third is the final result. In our proposed Iterative Learning approach we keep adjusting the photograph until it the user is pleased with the final result. The first image shows a progressive increase in quality. The second image changes very slightly at first (notice the decreased exposure in the clouds), but the last several steps greatly increase the quality. See Section 7.3 for a discussion.*


(a) Original Image


(b) Intermediate Result


(c) Final Result


(d) Original Image


(e) Intermediate Result

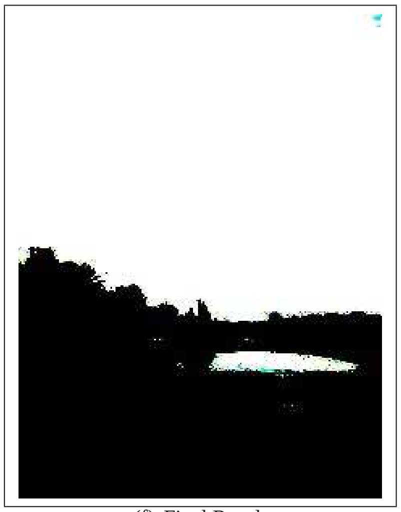

(f) Final Result

Figure 11: *Some example* ***failure cases*** *of our method. Notice in the first image, our method slowly increases the brightness until the photograph is washed out. In the second image, the brightness is decreased such that the bridge immediately deteriorates and remains in that state. We show a border around the second image to illustrate the amount of fully saturated pixels. See Section 7.3 for a discussion.*

Personalization by Adapting Style Priors to User Preferences Because aesthetics are very subjective, we need a way of taking into account an individual user's personal preferences. This should be more straightforward than the complicated approaches recently proposed [Bychkovsky et al. 2011; Yeh et al. 2010] because we have access to the user's Lightroom data as well. We can leverage this data by weighting each cluster (mentioned above) by the styles the user has applied in the past. This approach requires no extra effort for the user, unlike the other methods. It should also possible to personalize the parameter and sequence models that were described in Section 5.

Features Our feature vector is missing local data. By segmenting the foreground and background, we can extract more useful features. For example, we currently calculate global contrast on the entire image, but if the background is artistically out of focus, this feature is averaged between a sharp foreground and blurry background. Treating the two areas of the image separately alleviates this problem. We can also leverage the photograph metadata such as shutter speed, ISO, and f-stop. In general, more features will be useful, especially because each regression model selects the most useful features for the type of adjustment it is modeling.

Lightroom Connection We are currently using our own implementations of the Lightroom adjustments. They are similar but not equivalent to Lightroom's own implementation (which they do not specify). This makes our training data noisy: we read the Lightroom adjustment history and apply each successive transformation onto the original image, and use that to calculate the feature vector. Further, having everything cleanly inside of Lightroom will greatly simplify the interface, as described in Section 6 and shown in Figure 5.

Terminating Condition Currently, our algorithm asks the user when to stop iterating. Our system would be more robust if we could automatically decide when to terminate, and allow the user to ask the iteration to continue. One method would be to determine when there is little change between adjustments, or if the adjustments oscillate (e.g. increase in brightness, followed by an equivalent decrease in brightness).

Local Adjustments Many photographers apply adjustments only to specific parts of the image, such as skin or sky. To model this, we should be able to identify the region on which a local adjustment is made. We can then decide whether to make a local adjustment (and which type of local edit) or a global adjustment during another step of Sequence Learning.

Results Comparison To prove that our results are comparable to recent work, we will need a formal comparison of our application's final output. We plan to do this with a crowdsourced rating system, where participants rank our output compared to other methods. We also plan to evaluate the usefulness of the ILA to a novice photographer by conducting a controlled user study.

Applying to other datasets Iterative Learning will be applicable to many more datasets if we are able to incorporate the time domain. As an example, for videos, each iteration would need to also predict the frames on which to apply the adjustment.

9 Conclusion

We have presented a method to train a model of an expert artist. Through a combination of a sequence learning model and a set of parameter learning models, we are able to present a user with a step-by-step demonstration of how to improve an image. Our dataset is easy to gather and robust against noise. Learning the underlying model is easier using our dataset as opposed to datasets containing only input/output image pairs. We show promising results using only a very limited amount of training data, and describe a user interface that facilitates a efficient and educational workflow for novice photographers using Adobe Lightroom.

Acknowledgements

We would like to thank Igor Proskurin for his work on connecting this application to Adobe Lightroom and Jennifer Gross for collecting test data.

References

AKERS, D., LOSASSO, F., KLINGNER, J., AGRAWALA, M., RICK, J., AND HANRAHAN, P. 2003. Conveying shape and features with image-based relighting. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 349–354.

BERTHOUZOZ, F., LI, W., DONTCHEVA, M., AND AGRAWALA, M. 2011. A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Transactions on Graphics (TOG) 30*, 5, 120.

BRADSKI, G. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

BROWN, P., DESOUZA, P., MERCER, R., PIETRA, V., AND LAI, J. 1992. Class-based n-gram models of natural language. *Computational linguistics 18*, 4, 467–479.

BYCHKOVSKY, V., PARIS, S., CHAN, E., AND DURAND, F. 2011. Learning Photographic Global Tonal Adjustment with a Database of Input / Output Image Pairs. *IEEE Computer Vision and Pattern Recognition (CVPR)*.

CHEN, S., AND GOODMAN, J. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 310–318.

DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*.

EFRON, B., HASTIE, T., JOHNSTONE, I., AND TIBSHIRANI, R. 2004. Least angle regression. *The Annals of statistics 32*, 2, 407–499.

FATTAL, R., AND LISCHINSKI, D. 2002. Gradient domain high dynamic range compression. *ACM Transactions on Graphics*.

FU, W. 1998. Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 397–416.

GOOD, I. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika 40*, 3-4, 237–264.

HOERL, A., AND KENNARD, R. 1970. Ridge regression: applications to nonorthogonal problems. *Technometrics*, 69–82.

HOERL, A., AND KENNARD, R. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 55–67.

KANG, S. B., ASHISH, K., AND DANI, K. 2010. Personalization of Image Enhancement. *CVPR*, 1799–1806.

KATZ, S. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on 35*, 3, 400–401.

KNESER, R., AND NEY, H. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1, IEEE, 181–184.

LIDSTONE, G. 1920. Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries 8*, 182-192, 13.

MARKS, J., ANDALMAN, B., AND BEARDSLEY, P. 1997. Design galleries: A general approach to setting parameters for computer graphics and animation. *on Computer graphics*, 389–400.

MENARD, S. 2000. Coefficients of determination for multiple logistic regression analysis. *American Statistician*, 17–24.

MORENO-NOGUER, F., NAYAR, S. K., AND BELHUMEUR, P. N. 2005. Optimal illumination for image and video relighting. *ACM SIGGRAPH 2005 Sketches on - SIGGRAPH '05*, 75.

NEY, H., ESSEN, U., AND KNESER, R. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language 8*, 1, 1–38.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND E., D. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research 12*, 2825–2830.

PELI, E. 1990. Contrast in complex images. *J. Opt. Soc. Am. A 7*, 10 (Oct), 2032–2040.

PETSCHNIGG, G., SZELISKI, R., AGRAWALA, M., COHEN, M., HOPPE, H., AND TOYAMA, K. 2004. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics 23*, 3 (Aug.), 664.

RASMUSSEN, C. 2004. Gaussian processes in machine learning. *Advanced Lectures on Machine Learning*, 63–71.

TIBSHIRANI, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

WILLIAMS, C., AND RASMUSSEN, C. 1996. Gaussian processes for regression.

YEH, C. H., HO, Y. C., BARSKY, B. A., AND OUHYOUNG, M. 2010. Personalized photograph ranking and selection system. ACM, Firenze, Italy, MM '10, 211–220.

ZOU, H., AND HASTIE, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*, 2, 301–320.