Personalized Image Enhancement

Gregory Luppescu
Electrical Engineering
Stanford University
gluppes@stanford.edu

Raj Shah
Electrical Engineering
Stanford University
shahraj@stanford.edu

***Abstract*—We implement an auto-enhancement framework that can learn user preferences to enhance images in a personalized way. Our method finds a maximally representative training subset (20 images) out of a large dataset, allowing for efficient training. The parameters chosen in the training phase can then be applied accordingly to other images in the dataset, automatically creating an entire library of personally customized images. To test this method, we performed eight user studies, and the results suggest that the personalization of image enhancement parameters is sometimes better than conventional enhancement methods.**

I. Introduction

Image enhancement is almost always a necessary step after taking a picture with a digital camera. Many different photo software packages, like Google Photos or Photoshop, attempt to automate this process with various auto enhancement techniques. None of these methods, however, take user preference into account. Social media outlets like Instagram supply their users with many different types of filters to add some level of customization, but the user must manually choose a filter every time he or she uploads a photo. While manually customizing every image in a library is not a problem for a small amount of photos, the process can quickly become tedious for larger photo libraries. In this paper, we implement a system that can learn a user's preferences and apply those preferences to the rest of his or her photo library.

II. Related Work

Image enhancement has been an active field of research, especially, automatic image enhancement. Kaufman et al.[1] propose an automatic photo enhancement pipeline that is content-aware. Unlike other automatic photo enhancement methods, they take the local semantics of the image into account and specifically, attempt to enhance faces, blue skies, and underexposed regions in the image. Celik and Tjahjadi [2] use Gaussian mixture modeling to model the gray-level distribution in the image and then, using this distribution, automatically enhance the contrast of the image. Dale et al. [3] develop a visual search technique to find the closest images to the input image from a large web database. The image enhancement pipeline uses these images to define the input image's visual context, and then uses this visual context for a number of image enhancement operations like white balance, auto-exposure, and contrast correction. While these methods try to make the auto enhancement pipeline more content-aware, they do not address the problem of incorporating user preferences. The work of this project is a modified version of the method developed in [4].

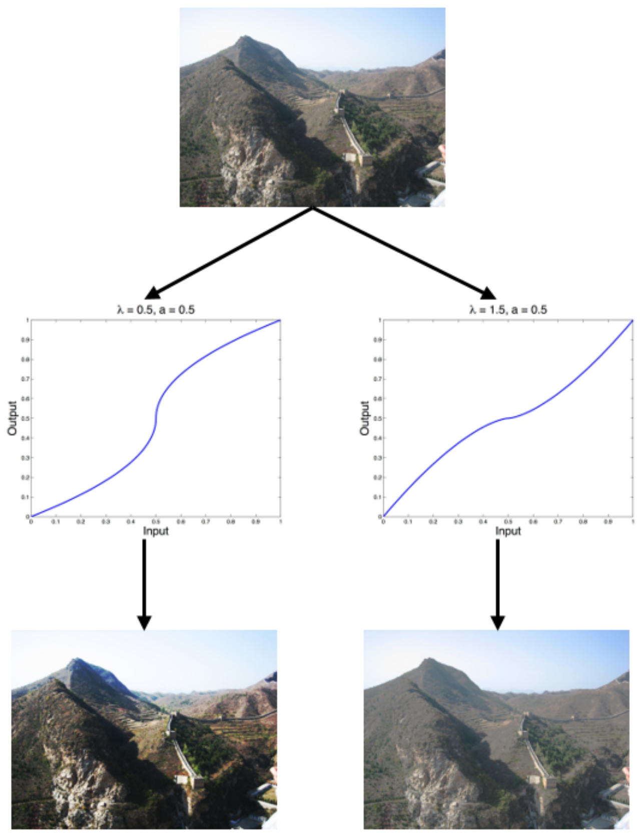


Fig. 1: Example results after applying two different S-curves. The left image results after applying an S-curve with parameters $a = 0.5$ and $\lambda = 0.5$. The right image results after applying an S-curve with parameters $a = 0.5$ and $\lambda = 1.5$.

III. Enhancement Parameters

We decided to focus on two types of image enhancements: contrast manipulation and color correction. Contrast manipulation was achieved by applying an S-curve to the images, and color correction was performed by modifying the color temperature and the tint of the images.

A. *S-Curve*

The equation for the S-curve is given as

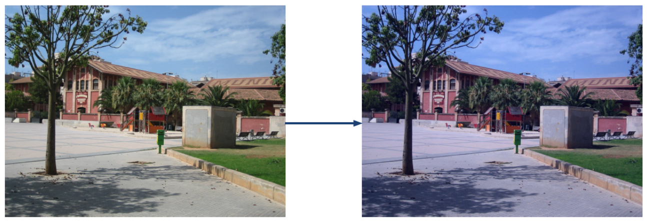


Fig. 2: Example results after modifying T and h

$$y = \begin{cases} a - a(1 - \frac{x}{a})^{\lambda} & \text{if } x \leq a \\ a + (1-a)(\frac{x-a}{1-a})^{\lambda} & \text{otherwise} \end{cases} \tag{1}$$

where a determines the inflection point of the S-curve, λ determines the shape of the S-curve, x is a normalized input pixel value, and y is a normalized output pixel value. For $\lambda \geq 1$, applying an S-curve maps pixel values that are greater than a to lower values, and maps pixel values that are less than a to higher values. For $\lambda \leq 1$, the opposite is achieved. Examples of applying various S-curves to an image are shown in figure 1.

B. Color Correction

Color correction can be achieved by modifying the color temperature T and the tint h. The color temperature is defined as the wavelength of light emitted by an ideal blackbody radiator heated to some temperature, where warmer colors are more blue, while cooler colors are more red. In essence, modifying the color temperature of an image simply changes how warm or cool (blue or red) an image looks. Orthogonal to color temperature is the tint, which determines the amount of green in an image. To apply color correction, we simply calculate

$$\begin{aligned} R_{out} &= R_{in} - \Delta T \\ G_{out} &= G_{in} + \Delta h \\ B_{out} &= B_{in} + \Delta T \end{aligned} \tag{2}$$

where R_{in}, G_{in}, and B_{in} are the input RGB values for a given pixel, ΔT is the change in color temperature, Δh is the change in tint, and R_{out}, G_{out}, and B_{out} are the output RGB values for a given pixel. Examples of changing color temperature and tint are shown in figure 2.

IV. Methodology

A. Dataset

The dataset used for this project consisted of 500 photos. The images were selected to represent a typical user photo library, consisting of a wide variety of contexts. The categories of photos included rural landscapes, urban areas, faces and people, bodies of water, etc. The images were taken from [5].

B. Pre-processing

To deal with low quality images in the dataset, all images were auto enhanced before applying personalized enhancement. The auto enhancement procedure consists of two steps:

1) Auto-white balancing: The images are white-balanced using the gray-world assumption for the brightest 5% of pixels. This step uses 3 scaling factors to transform each color channel.

2) Auto-contrast stretch: Here, we first convert the image into grayscale and then find intensities I_L and I_H, where I_L is greater than at most 0.4% of the intensities and I_H is less than at most 1% of the intensities. Finally, the pixel values are linearly transformed such that I_L is mapped to 0 and I_H is mapped to 1.

C. Distance Metric

One of the critical steps in this project is finding an effective metric to measure the similarity between a pair of images. The distance metric is formulated on the basic assumption that if the images are similar, then their auto enhancement parameters are also similar. Thus, we define the sum of the absolute difference between the 5 auto enhancement parameters (found in section IV-B) for a pair of images as

$$D^{params}(i,j) = \sum_{k=1}^{5} |p_{ik} - p_{jk}| \tag{3}$$

where p_{ik} is the k^{th} auto enhancement parameter for image i. We define our distance metric as

$$D_{\alpha}^{images}(i,j) = \sum_{n=1}^{25} \alpha_n D^n(i,j) \tag{4}$$

where

$$\alpha^* = \arg\min_{\alpha} \sum_{i,j} \| D_{\alpha}^{images}(i,j) - D^{params}(i,j) \|_2^2 \tag{5}$$

24 of the distance metrics we used were the KL Divergence, L_1, L_2, and L_∞ norms of the differences between:

- Two color images
- The R, G, and B histograms of two color images
- The grayscale values of two color images after being converted to grayscale
- The grayscale histograms of two color images after being converted to grayscale

The last distance metric is the entropy of the difference between two color images. To find the value of α^*, we used the BFGS algorithm to minimize the objective function given in equation 5. In order for equation 5 to be well conditioned, it was necessary to normalize the individual distance functions such that they had comparable values. The L_1 norm and KL Divergence of the differences of the color images were divided by $3 *$ number of pixels. The L_1 norm and KL Divergence of the differences of the grayscale images were divided by the number of pixels. The L_2 norm of the differences of the color images were divided by the square root of $3*$number of pixels. The L_2 norm of the differences of the grayscale images were divided by the square root of number of pixels. The L_1 norm of the differences of histograms were normalized by $256 *$ number of pixels, as the number of bins in the histograms was

256. Finally, the L_2 norm of the differences of histograms were normalized by the square root of $256 *$ number of pixels.

D. Training Set Selection

Since the purpose of this pipeline is to make image personalization as efficient as possible, it is imperative to have a small training set to ensure a seamless user experience. Ideally, each training image would represent a specific category in the dataset. For instance, if our dataset consisted of images of faces, cats, and trees, our training set should at least contain one face, one cat, and one tree. Then, the parameters chosen for each of these training images would be applied to all other images within their respective categories. Ideally, the distance metric learned in IV-C will determine similarity between images such that images in the same category are closest to each other. Practically, however, with a large unlabeled image set containing images not belonging to specific categories, we must select a training set that is maximally representative of the dataset. This problem can be interpreted as a sensor placement problem [6], where the top n sensors (training images), when combined, share the most amount of information with the rest of the dataset. To achieve this objective we employ a greedy selection procedure found in [6], which considers a Gaussian process with covariance matrix

$$k_{i,j} = \exp\left(-N^2 \frac{D_\alpha^{images}(i,j)}{\sum_{k,l} D_\alpha^{images}(k,l)}\right) \qquad (6)$$

where $D_\alpha^{images}(i,j)$ is the distance described in IV-C between images i and j, and N is the total number of images in the dataset. The covariance matrix K has an intuitive interpretation: each entry (i,j) encodes the similarity between image i and image j, where a value of 0 signifies the two images are infinitely far apart, and a value of 1 signifies the images are at the same location in the image space.

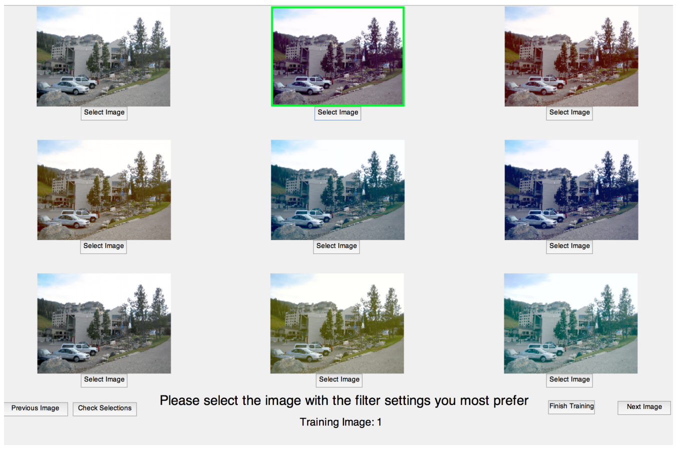
As we are employing a greedy algorithm, each selection step chooses the image that maximizes the gain in mutual information among the unselected images:

$$I^* = \arg\max_{I_i \in U} f(i) \qquad (7)$$

where

$$\begin{aligned} f(i) &= MI(U-i; S \cup i) - MI(U-i; S) \\ &= \frac{1 - k_{S,i}^T K_{S,S}^{-1} k_{S,i}}{1 - k_{U-i,i}^T K_{U-i,U-i}^{-1} k_{U-i,i}} \end{aligned}$$

In equation (7), $MI(x,y)$ is the mutual information between x and y [7]. S is the set of images already selected to be a part of the training set, and U is the set of unselected images. $K_{S,S}$ is the similarity matrix among images in S, whose values are calculated according to (6). $K_{U-i,U-i}$ is the similarity matrix among images in U excluding image i. Similarly, $k_{S,i}$ and $k_{U-i,i}$ are the similarity vectors between images in S and $U-i$ with image i. It can be intuitively seen that if the the similarity of image i with $U-i$ is high, then the denominator is small. Conversely, if the similarity of image i with S is low, then the numerator is large. Thus, finding i such that it maximizes $f(i)$ implies that image i is very similar to the unselected images, but at the same time, dissimilar to selected images, which is what we are trying to achieve. Greedily finding an image i at each iteration gives us a ranking of images that maximally represent the dataset. For this project, we choose the top 20 images as our training set. The rationale behind choosing 20 images is to make the training convenient and time-efficient for the user.



Please select the image with the filter settings you most prefer

Training Image: 1

Fig. 3: Training GUI implemented in MATLAB

E. Parameter Set Selection

As mentioned in section III, we learn 4 enhancement parameters in this project. For each of the enhancement parameters, we decided to use 3 different values, which gives us a total of $3^4 = 81$ different parameter combinations for each training image. The values used for each parameter are listed in table I. To decide the 3 values per parameter, we applied various changes to see how resulting images were affected. As was expected, the more aggressive the change, the more unnatural the image would look. We ended up choosing parameters that noticeably changed the images, while keeping the images looking reasonably natural.

Asking the user to choose the best parameter combination out of 81 different choices is simply impractical. Hence, we decided to select an optimal subset of parameter combinations to make the training process less tedious. To achieve this, per image, we apply all 81 different parameter combinations and then use the same sensor placement optimization procedure in section IV-D to choose the top 8 parameter combinations. As a result, the 8 parameter combinations found maximally represent the parameter space for that image. Now, instead of choosing among 81 versions of the same image, the user only has to choose between 9 versions: the original auto enhanced image, and the 8 images enhanced using the selected parameters.

F. Training

Once the training images and parameter combinations per training image are selected, the system is ready to learn a

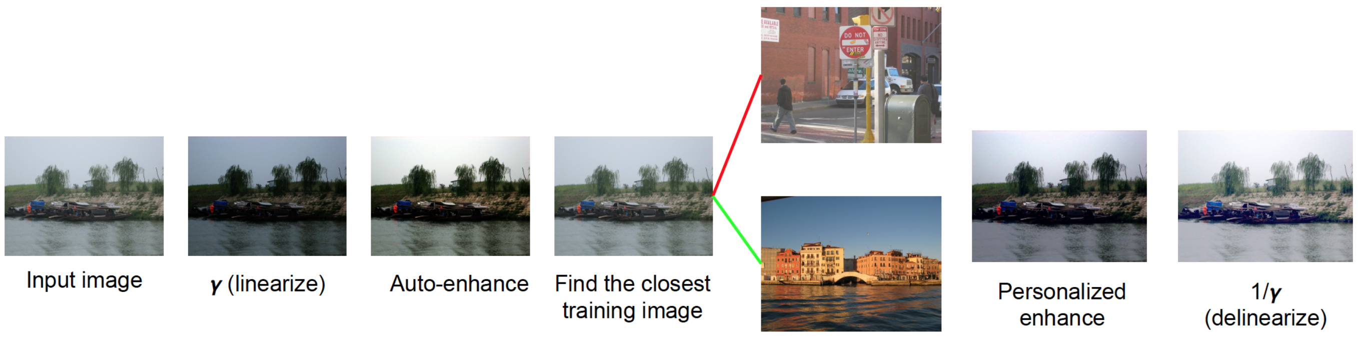


Fig. 4: Image Processing Pipeline to Personally Enhance Newly Added Images

Parameters	Values
λ	0.75, 1, 1.5
a	0.3, 0.5, 0.7
ΔT	-7.5, 0, 7.5
Δh	-7.5, 0, 7.5

TABLE I: Values used for each parameter

user's preferences. For training, we implemented a GUI in MATLAB, which can be seen in figure 3. For each training image, a user chooses which enhanced photo he or she prefers the most. The GUI signifies the image chosen by highlighting it with a green border. The user can traverse the training set using forward and backward buttons, and can press the ”Check Selections” button to check which training images still have unspecified parameter combinations. We tried to make the GUI very easy to use so that the training phase would be as smooth as possible. After a user has selected all preferences for each training image, he or she can click ”Finish Training” to save the preferences chosen.

V. Processing Pipeline

After the user preferences are learned, all other images in the dataset can be personally enhanced. The enhancement pipeline (figure 4) is as follows:

1) Transform the image into the perceptually linear domain (we used $\gamma = 2.2$)
2) Auto enhance the image using the procedure described in IV-B
3) Find the closest training image to the auto enhanced input image using the distance metric learned in IV-C
4) Get the enhancement parameters chosen by the user during the training phase associated with the closest training image
5) Apply these enhancement parameters to the test image
6) Perform the inverse operation from step 1) to produce the final image

VI. Results

A. Testing Procedure

To test the efficacy of our method, we took 10 test images from the dataset, each with varying contexts (faces, landscapes,

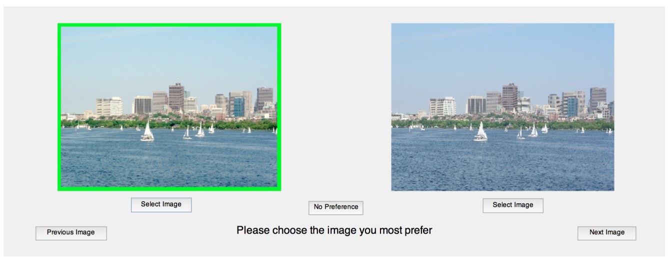


Fig. 5: Testing GUI implemented in MATLAB

etc.). We enhanced these images three different ways: using our personalized enhancement method, the Google Photos auto enhancement feature, and Photoshop's auto contrast and color correction tools. We then performed one-on-one comparisons between the personalized images and the original images, the personalized images and the Google enhanced images, and the personalized images and the Photoshop enhanced images. To perform the comparisons, we created another GUI in MATLAB which can be seen in figure 5. Note that there is an option for ”No Preference,” as some pairs of images may be indiscernible from one another.

B. User Study

The results of 8 user studies can be seen in table II. A little more than 50% time, users preferred the personalized images to the original images, had no preference about 20% of the time, and preferred the original images about a 25% of the time. These results are very reasonable, as more often then not, some type of enhancement is better than no enhancement at all. When compared to Google Photos and Photoshop, our method also yielded reasonable results, as users preferred personalized images about 30% of the time, had no preference about 25% of the time, and preferred the professional enhancement tools about 45% of the time. Since we are only modifying four parameters, our method is limited as to how effectively it can enhance an image. We expect that the auto enhancement features for Google Photos and Photoshop both modify many other parameters, which could be one reason why those photos were preferred a majority of the time. Another source of error could come from the fact that the learned distance metric is imperfect. For example, if according to our distance metric a test image of a face is closer to a training image of a rural landscape than to a training image of a face, parameters associated with the training image of the rural landscape would be applied to the test image of the face, leading to undesirable enhancement results. We also noticed that people who preferred more artistic looking photos (like, for instance, a photo put through an Instagram filter) liked the personalized results better than the professionally auto enhanced results. This observation also makes sense, as the professional auto enhancement algorithms are most likely trying to keep images looking as natural as possible. Overall, we were pleased with the results.

Personalization vs. Original

Preferred Personalization	54.3%
No Preference	18.6%
Preferred Original	27.1%

Personalization vs. Google Photos

Preferred Personalization	28.6%
No Preference	21.4%
Preferred Google Photos	50.0%

Personalization vs. Photoshop

Preferred Personalization	30.0%
No Preference	30.0%
Preferred Photoshop	40.0%

TABLE II: User Study Results

VII. Conclusions & Future Work

In this project, we implement an end-to-end pipeline to learn user preferences to enhance images in a personalized way. The five major components of this project are: computing a distance metric, finding a training set that maximally represents the dataset, finding an optimal parameter set for each training image, training, and finally, enhancing the images. The efficiency of this approach lies in the fact that almost all of the processing can be done offline, so the user is involved only in a short training phase. To test the validity of this method, we carried out user studies, and the fact that our method was preferred over professional software for some images shows the potential of this approach.

Due to limited time and computational resources, we decided to work with a smaller dataset of images and to also learn fewer parameters. However, we feel that a larger dataset of images could yield a more accurate distance metric, which would in turn yield a training set that is more representative of the dataset. Also, applying more enhancements and thus, learning more parameters can help achieve better personalization. Lastly, exploring different ways of learning distance metrics and finding training images that hold maximum information could go a long way in making this pipeline more effective.

Acknowledgment

We would like to thank Prof. Gordon Wetzstein and the TAs of EE 368 for guiding us through the course of this project and for providing the necessary resources.

References

[1] Liad Kaufman, Dani Lischinski, and Michael Werman. Content-aware automatic photo enhancement. In *Computer Graphics Forum*, volume 31, pages 2528–2540. Wiley Online Library, 2012.

[2] Turgay Celik and Tardi Tjahjadi. Automatic image equalization and contrast enhancement using gaussian mixture modeling. *IEEE Transactions on Image Processing*, 21 (1):145–156, 2012.

[3] Kevin Dale, Micah K Johnson, Kalyan Sunkavalli, Wojciech Matusik, and Hanspeter Pfister. Image restoration using online photo collections. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2217–2224. IEEE, 2009.

[4] Sing Bing Kang, Ashish Kapoor, and Dani Lischinski. Personalization of image enhancement. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1799–1806. IEEE, 2010.

[5] Labelme, the open annotation tool. http://labelme.csail.mit.edu/Release3.0/.

[6] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[7] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley, New York, 2006.