Learning Large-Scale Automatic Image Colorization

Aditya Deshpande, Jason Rock and David Forsyth
University of Illinois at Urbana-Champaign
{ardeshp2, jjrock2, daf} @illinois.edu
http://vision.cs.illinois.edu/projects/lscolor

Abstract

We describe an automated method for image colorization that learns to colorize from examples. Our method exploits a LEARCH framework to train a quadratic objective function in the chromaticity maps, comparable to a Gaussian random field. The coefficients of the objective function are conditioned on image features, using a random forest. The objective function admits correlations on long spatial scales, and can control spatial error in the colorization of the image. Images are then colorized by minimizing this objective function.

We demonstrate that our method strongly outperforms a natural baseline on large-scale experiments with images of real scenes using a demanding loss function. We demonstrate that learning a model that is conditioned on scene produces improved results. We show how to incorporate a desired color histogram into the objective function, and that doing so can lead to further improvements in results.

1. Introduction

We describe a method that learns to colorize grey-level images. Our method learns a cost function that evaluates local predictions of color, spatial consistency, and consistency with an overall histogram. There are two reasons to be interested in colorization. First, solutions have some practical applications (colorizing old movies or photographs; correcting color in legacy images). Second, the problem is a good model for a wide range of problems. In many cases, we wish to take an image and predict a set of values at each pixel in the input image, using information from the input image. Our predictions should have significant long-scale spatial structure. Problems like predicting albedo, shading, depth, denoised images, and so on have this form. One advantage of colorization as a model is that immense colorization datasets are easily available, and they are organized in interesting ways. We use the SUNS dataset [20], which is organized by scene.

It is natural to predict image maps by using image data and prior knowledge to set up an optimization problem, which is solved to recover the desired representation. Rather than using domain knowledge to set up prior or likelihood terms, we train an optimization problem by requiring it to produce good colorizations of training data.

Contributions: Our colorization method is learned from data, using a novel variant of LEARCH to balance pixelwise accuracy and spatial error. Comparable methods for training Gaussian Random Fields must impose positive definiteness constraints on the inverse covariance matrix, and encounter practical limits on the scale of spatial terms in the inverse covariance matrix; our method avoids these difficulties. Our method significantly outperforms the best baseline we are aware of, in the first quantitative colorization experiments we are aware of. We show how to exploit a target histogram to apply global constraints. We show that possessing a scene label at run-time always provides a target histogram that results in improved quantitative performance performance; this scene label could come from an oracle, from application logic, or from a scene classifier applied to the grey-level image.

1.1. Related Work

The problem most like ours is predicting an intrinsic image (one predicts albedo and shading instead of the color layers). The traditional approach splits an image into shading and albedo components [9]. Good strategies should have three properties. First we wish to correctly predict individual pixels. Second we wish to avoid bad spatial patterns in the output, even over long scales. Third we should be capable of predicting multiple channels, even when those channels have complex interactions. The properties are usually in tension. For example the best independent prediction of pixel values generally contains bad patterns. Traditionally, this tension is managed by an optimization problem. A learned data term attempts to predict each pixel correctly based on some local information while hand chosen priors enforce spatial and channel coherence. While the data terms are often portable, priors are often specific to particular problems, and can be hard to identify. For example,

Barron and Malik provide a good review and an extremely strong method for decomposing images into albedo, shading and shape fields [1]; however, their results depend delicately on a good choice of prior, and their priors require considerable domain knowledge to produce.

Data: We choose to study colorization because very large datasets are easily obtained by dropping the color representation of any collection of color images. There are datasets for shading and albedo decomposition, but these have disadvantages. The pioneering dataset of Grosse et al. has been extensively studied, but is small and shows isolated objects of quite limited material complexity [6]. Bell et al's dataset does not annotate entire images [2].

Notation: We write vectors as $\mathbf{b}$ and matrices as $\mathcal{W}$. I is the input grey-level image and $\mathbf{c}$ is the set of color layers we wish to infer, rearranged into a vector.

Learning to Optimize: A Gaussian random field (GRF) models the log-likelihood of a colorization (or other set of intrinsic image layers) $\mathbf{c}$ as $-\left[(1/2)\mathbf{c}^T\Sigma^{-1}(I)\mathbf{c} - \mathbf{b}^T(I)\mathbf{c}\right] + K$, where K is a constant of no interest; the first application to intrinsic images is by Tappen et al [18]. Maximum likelihood inference involves solving $\Sigma^{-1}\mathbf{c} = \mathbf{b}$. Learning by maximizing likelihood is impractical, because the term in $\det\Sigma$ is difficult to manage; instead, one learns by maximizing pseudolikelihood. An important difficulty of GRFs is obtaining models of Σ^{-1} that control long-scale spatial effects (have many non-zero terms) without introducing unmanageably many parameters and keeping Σ^{-1} positive definite. Jancsary et al. use a regression tree model of Σ^{-1} and $\mathbf{b}$; the practicalities of computing pseudolikelihood limit the range of spatial support possible, and they must adapt the learning algorithm to ensure the estimate is positive definite [8]. Like Jancsary et al. we learn a quadratic optimization problem in $\mathbf{c}$, but we apply no probabilistic interpretation. In contrast, we extend LEARCH [17], a framework for learning an objective function from examples, in a manner that allows us to control long spatial scales and provides a positive definite Hessian without difficulty.

Colorization: Producing a color image from a monochrome image is again a standard problem. Most current solutions are intended to be part of an authoring pipeline, and have an interactive component. We are not aware of a standard quantitative measure of performance or of quantitative studies. A good review appears in [12]. Jancsary et al. show that GRF's can be applied to colorization [8]. Charpiat et al. predict multiple colors for each pixel by estimating conditional probabilities over texture features and enforce smoothness using graph-cuts to find globally optimal colors [4]. Similarly, Bugeau et al. perform energy minimization using variational methods to find optimal color from multiple predictions [3]. Hertzmann et al. demonstrated that their image analogies method could be used to colorize [7], and the approach was extended by Welsh [19] by introducing different normalization and matching step. Morimoto et al. [13] showed how to choose a good exemplar for [19] automatically; we use this method as our baseline.

2. Learning an Objective Function

We wish to learn an objective function $\Phi(\mathbf{c}, I)$ such that $\operatorname{argmin}_{\mathbf{c}}\Phi$ is close to the correct colorization of I. We expect $\mathbf{c}$ to be very large, so it is natural to restrict our attention to problems quadratic in $\mathbf{c}$. It is also natural to require the Hessian be positive definite, yielding a single solution. Such a problem is equivalent to a GRF. The primary issues here are (a) obtaining a parametric representation of the Hessian that allows long scale control with few parameters and (b) ensuring the Hessian is positive definite. We drop the probabilistic interpretation, because it is not required to attack these problems.

Write d for the dimension of $\mathbf{c}$, $\mathbf{b}(I)$ for a vector that is a function of the image, and $\mathcal{A}(I)$ for a matrix, with column rank at least d, that is a function of the image. Then the most general objective function that meets our constraints is $\frac{1}{2}\|\mathbf{b}(I) - \mathcal{A}(I)\mathbf{c}\|^2$ It can be helpful to think of $\mathcal{A}(I)\mathbf{c}$ as a set of image-dependent linear features of $\mathbf{c}$ and $\mathbf{b}(I)$ as predictions of the features using I. There are too many parameters for feasible learning.

To limit the number of parameters, one could assume that effects in images are contained within some neighborhood. We write Π_u for the matrix which selects such a patch about pixel u. Then the form $\sum_{u\in\text{pixels}}\frac{1}{2}\|\mathbf{b}(I,u) - \mathcal{A}(I,u)\Pi_u\mathbf{c}\|^2$ is a simplification that exposes a unity between existing methods. Assume that $\mathcal{A}(I,u)$ is the identity then $\mathbf{b}(I,u)$ makes a prediction of the patch about u; we get a patch-matching approach like that of [7] and [19] (though these have a data-dependent prior on $\mathbf{c}$). If $\mathcal{A}(I,u)$ is the identity and $\mathbf{b}(I,u)$ returns a filtered version of the image I at u, then we have a filter forest [8]. However, for unconstrained $\mathcal{A}(I,u)$ and for large patches there are still too many parameters to learn.

Now define a set of f filters which are applied at each pixel. This allows us to limit the dimensionality of the problem without blinding our method to long-scale effects. Specifics of the filters chosen can be found in section 3.2, but we require one filter to be the identity. We write the linear operator that implements the filters as $\left[\mathcal{I}, \mathcal{F}^T\right]^T$, using this notation to keep track of the fact that one filter is always the identity. Now interpret Π_u to be the matrix that picks out all filter responses located at the center of the patch u. Consider

$$\Phi(\mathbf{c}; I) = \sum_{u\in\text{pixels}}\frac{1}{2}\|\mathbf{b}(I,u) - \mathcal{W}(I,u)\Pi_u\begin{bmatrix}\mathcal{I}\\ \mathcal{F}\end{bmatrix}\mathbf{c}\|^2 \quad (1)$$

where $\mathcal{W}(I,u)$ is $n \times f$, $n < f$, the first row of $\mathcal{W}(I,u)$ is $[1,0,\ldots 0]$ and so picks out the pixel value at u, and the rows of $\mathcal{W}(I,u)$ are orthonormal. Here $\mathcal{W}(I,u)$ can be thought of as projecting the many filter responses at u to a lower dimensional summary, which must be predicted by $\mathbf{b}(I,u)$. The column rank of $\mathcal{W}(I,u)$ is clearly n.

This notation is clumsy, so we drop the device of projection onto patches, and build $\mathcal{W}(I)$ by stacking the per-patch row orthonormal matrices, and similarly form $\mathbf{b}(I)$ to obtain

$$\Phi(\mathbf{c};I) = \frac{1}{2}\|\mathbf{b}(I) - \mathcal{W}(I)\begin{bmatrix}\mathcal{I}\\ \mathcal{F}\end{bmatrix}\mathbf{c}\|^2 \qquad (2)$$

where $\mathcal{W}(I)$ is now $(nd)\times(fd)$ and is obtained by padding the rows of each $\mathcal{W}(I,u)$ with zeros and stacking appropriately. We must have that the column rank of $\mathcal{W}(I)$ is (nd), because each column is obtained by taking a column of an appropriate $\mathcal{W}(I,u)$, and padding with zeros above and below. It follows that the Hessian of this objective function is positive definite (more detail in supplementary material). Qualitatively, $\mathcal{F}$ is a list of potentially significant patterns in $\mathbf{c}$, $\mathcal{W}$ identifies combinations of those filters to predict, and $\mathbf{b}$ predicts the filters. In what follows, we write $\mathcal{A}(I)$ for $\mathcal{W}(I)\left[\mathcal{I},\mathcal{F}^T\right]^T$.

2.1. Learning

We use LEARCH to learn appropriate $\mathcal{W}(I,u)$ and $\mathbf{b}(I,u)$ for pixels u independently [17]. Write $\Phi(\mathbf{c};\theta,I,u)$ for an objective function with parameters θ; in our case $\theta = \{\mathcal{W}(I,u),\mathbf{b}(I,u)\}$. Write $\{(\mathbf{c}_i^*,I_i)\}$ as a set of input ground truth color images and their corresponding grey-level image, and $H(\cdot,\cdot)$ for a margin. Then LEARCH requires colorizations which are further away from the ground truth (i.e. $H(\mathbf{c}^*,\mathbf{c})$ is large) should be given larger scores. This yields the objective in θ:

$$\sum_i \left[\Phi(\mathbf{c}_i^*;\theta,I_i,u) - \min_{\mathbf{c}}\left\{\Phi(\mathbf{c};\theta,I_i,u) - \lambda H(\mathbf{c}_i^*,\mathbf{c})\right\}\right] \qquad (3)$$

In our case the parameters θ are **functions** of the image, $\mathcal{W}(I,u),\mathbf{b}(I,u)$. The standard strategy for learning under these conditions is functional gradient descent on the objective function.

An important nuisance of solving LEARCH-style problems with functional gradient descent is that every step requires solving an inner optimization problem ($\min_{\mathbf{c}}\{\cdots\}$ in eq (3)) for every example. For an appropriate choice of margin this can be avoided. In particular, we chose.

$$H(\mathbf{c}^*,\mathbf{c}) = \|\mathcal{A}(I,u)(\mathbf{c}-\mathbf{c}^*)\|^2 \qquad (4)$$

With this margin, we can complete the square to retrieve a closed form solution of eq (3) (supplementary section 1).

Such a margin may not be appropriate for all learning problems because $\mathcal{A}(I,u)$ has a non-trivial nullspace. Therefore, $\Phi(c;\theta,I,u)$ can possibly grow only in some (rather than all) dimensions of the image patch (Refer supplementary section 1). However, in our case 1) our patch filters form a sufficient (even if incomplete) representation of the diversity in real image patches and 2) $\mathcal{W}(I,u)$ identifies the important combination of those filters for the specific image patches we are considering. Furthermore, we constrain $\mathcal{W}$ to be orthonormal which eliminates the trivial solution.

3. Implementation

3.1. Learning in practice

We represent $\mathcal{W}(I)$ and $\mathbf{b}(I)$ as a sum over regression trees, as in [5]. There are n rows of each for each pixel location. Assume there are t regression trees, write orth for the operator that orthonormalizes the rows of a matrix, $\mathcal{W}^i(I,u)$ for the estimate of the n rows corresponding to the u'th pixel location computed by the first i trees, and $\Delta\mathcal{W}^{(i+1)}(I,u)$ for the contents of the leafs of the $i+1$'th tree reached by passing the features at the u'th pixel location down the tree. Then we have the update

$$\mathcal{W}^{(i+1)}(I,u) = \text{orth}(\mathcal{W}^{(i)}(I,u) + \Delta\mathcal{W}^{(i+1)}(I,u)) \qquad (5)$$

Each leaf of each tree also contains an affine function predicting an update to the values of $\mathbf{b}(I)$ from $\Psi_r(I,u)$, the regression features evaluated at pixel location u (see section 3.3). Using the notation of the previous paragraph with the exception that $\Delta\mathcal{B}^{(i+1)}(I,u)$ is now an affine function, we have the iteration

$$\mathbf{b}^{(i+1)}(I,u) = \mathbf{b}^{(i)}(I,u) + \Delta\mathcal{B}^{(i+1)}(I,u)(\Psi_r(I,u)) \qquad (6)$$

We depart from tradition here in our computation of trees as we perform line search at each leaf independently. This allows us to make maximal progress on each leaf, regardless of the state of the tree. We believe this is an important feature for colorization as we expect the error to be dominated by a small number of difficult to predict patches. We also differ from traditional regression trees due to the orthonormalization which means that during inference we must traverse the trees and accumulate their effects in the same order they were learned.

3.2. Constructing Filters

In defining a set of filters $\mathcal{F}$ for our regression, there is no point in controlling effects that do not occur in images. A natural vocabulary for an image representation is bars and spots at various scales and orientations. We also learn filters created from eigen-patches corresponding to the largest eigenvalues. These eigen-patches attempt to encode specific dataset peculiarities. An obvious question is which vocabulary is best, however, we do not currently have a satisfactory answer. Detailed information on the filters we used in supplementary.

3.3. Features

We seek to define two sets of features: split features (Ψ_s), as the name suggests determine the splits in our regression trees and regression features (Ψ_r) are used as predictors. Split features should provide a good description for the classification of pixels with similar characteristics, and thus similar color. We use grey-level value, blurred grey-level value, grey-level gradients, and average color and variance for this. Average color and variance are computed for a query grey-level image by retrieving the top-k most similar images from an image dataset. We use bag-of-features retrieval using SIFT features computed on the grey-level image [14]. A standard vocabulary tree is used to quantize SIFT features to visual words and we find the top 9 images with nearest tf-idf vectors. We compute mean and variance at each pixel.

Regression features (Ψ_r) should embody properties of the neighborhood and exhibit a strong correlation to the color. For this we use LM filter bank responses (scaled between 0 and 1), since they are good at discriminatively identifying the material and texture of swatches [11].

3.4. Inference

In general, minimizing a quadratic objective on a large non-sparse matrix is difficult because minimization requires solving a large linear system. In our case, inference requires solving the linear system

$$[\mathcal{A}(I)^T\mathcal{A}(I)]\mathbf{c} = \mathcal{A}(I)^T\mathbf{b}(I) \tag{7}$$

but we cannot form or store $\mathcal{W} = \mathcal{A}(I)^T\mathcal{A}(I)$ because it is too large and non-sparse. However, we can compute the product of $\mathcal{W}$ with a vector $\mathbf{x}$: form $\mathbf{x}$ as an image, convolve it with the filters, multiply by a sparse matrix and then filter again. This structure allows us to use pre-conditioned conjugate gradient to solve this linear system (see supplementary section 2).

3.5. Histogram Correction

The color image $\mathbf{c}$ inferred above, henceforth called the source image, can be improved further by enforcing global properties (e.g. beach scenes have many blue pixels for sky/water, indoor scenes have white walls, effects of yellow lighting etc.). A known method to perform this in image manipulation literature is histogram adjustment [16]. We develop a novel histogram correction step.

We model the desired target histogram (t) as Gaussian mixture model (GMM) obtained using the EM algorithm. The number of components (M) in GMM are equal to the modes obtained by performing mean-shift clustering on the target histogram. We then find the corresponding modes in the histogram of the source image (s). This is done by initializing mean-shift to the modes of the target histogram and allowing it to shift up to a threshold distance. The source image histogram is then modeled by a GMM, now with a known correspondence of the M components of the target histogram and the M components of the source histogram.

Write $\mu_i, \sigma^2 I, w_i$ for the mean, covariance and weight of the i^{th} Gaussian component in GMM. We distinguish between source and target histogram using superscripts s, t respectively. A standard measure for the divergence between two GMMs is the Bhattacharyya distance, which in the case of constant spherical covariance becomes.

$$\Phi_B(s||t) = \sum_{i=1}^{M} \frac{1}{8\sigma^2}||\mu_i^s - \mu_i^t||^2 - \frac{1}{2}ln(w_i^s w_i^t) \tag{8}$$

Notice that correspondence between the components must be known in the Bhattacharyya distance above, as in our case. μ_i^s and w_i^s are functions of the source image $\mathbf{c}$ as per equations of EM algorithm. This allows us to perform a steepest gradient descent to find optimal $\mathbf{c}$. Closed form derivatives with respect to $\mathbf{c}$ can be obtained and we update the soft assignments to Gaussian components after every descent step, details are in supplementary section 3.

Our final objective function is a weighted sum of the LEARCH objective Φ_L and the Bhattacharyya distance Φ_B. This ensures spatial coherence while performing histogram correction. The complete objective function is: $\Phi = \Phi_B + \lambda_L\Phi_L$. The weight λ_L is learned by search using a validation set, as discussed in Section 4.2.

3.6. Scene Histograms

Histograms can be estimated automatically from training data by taking the normalized histogram of all training images. We refer to this as the mean histogram for a scene, and use it as the target histogram in some of our experiments.

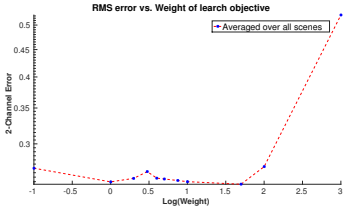
3.7. Scene Classification

Scene labels can be provided automatically [10, 20] at high accuracy. We use a scene classifier which uses GIST features [15] to provide scene labels. We verify that our classifier produces results comparable to those reported in [20] for a GIST scene classifier on the 15 scene dataset. Predicted scene labels are used to create a fully automatic scene specific colorization method.

4. Experiments

4.1. Dataset

We perform colorization on 6 scene categories of the SUN dataset, viz. *beach, castle, outdoor, kitchen, living room, bedroom*. We chose 3 indoor and 3 outdoor categories with maximum number of images. All images are rescaled to have height of 256, with aspect ratio maintained.



(a) Weight λ_L vs. 2-channel error

(b) $\lambda_L = 0$

(c) $\lambda_L = 5$

(d) $\lambda_L = 1000$

Figure 1: Large weight (λ_L) for the LEARCH objective prevents modification of colors by histogram correction. Lowering λ_L, makes colors vivid, e.g. sand becomes yellowish. Very low λ_L can cause artifacts as it downweighs the spatial coherence.

For each scene category, we randomly select 40 color/grey-level image pairs as training data, 20 image pairs for validation and 40 grey-level images for testing. For scene independent training, we merge the training images of all the 6 categories together. We perform a parameter search on validation and use the optimal parameters in test (Section 4.2). The remaining images in each category are used as a database to obtain the top-k matching images for generating average color image (Section 3.3).

4.2. Parameter Search

Learch. Our model has hyper-parameters which determine the tree structure, sampling of training data, and LEARCH objective function parameters. The number of trees (t_n) and the maximum depth (t_d) define the forest parameters. The number of samples per tree (t_s), the minimum number of samples per leaf (l_s), and the number of samples from each training image (i_s) determine how to handle training data. The inner dimension of $\mathcal{A}$, the LEARCH margin λ determine the function we will learn. We perform a search over these parameters and use the optimal values.

We search values which affect the objective function first, since improvements should be independent of the tree parameters. We found large inner dimensions of $\mathcal{W}$ improve performance but cost memory. We use an inner dimension of 12. A margin $\lambda = .25$ provides a good tradeoff between enforcing the margin without allowing it to dominate.

We then search over the tree parameters. Rather than limiting our trees by depth, we find that allowing very deep trees $t_d = 60$, and enforcing a large minimum samples per leaf $l_s = 100$ works well. We find that a relatively small number of trees $t_n = 8$ works well due to their expressiveness. We set $t_s = 7000$ and $i_s = 4000$ for 40 images.

Histogram Correction. We vary the weight of the LEARCH objective (λ_L) with respect to Bhattacharyya Distance between source and target GMMs. In Figure 1, we vary it from 0 to 1000 and observe error is lowest for values between 1 and 10, we set it to 5.

4.3. Error Metric

Since intensity information ($I = \frac{R+G+B}{3}$) is already present in the grey-level image, we only estimate 2 out of the 3 channels. During training, we transform $RGB-$color space to a de-correlated 2-channel *normalized opponent* color-space. The 2 channels are $I_a = \frac{B}{I} - \frac{(R+G)}{2I}$ and $I_b = \frac{R-G}{I}$.

In the 2-channel image, the intensity information is suppressed and values represent colors. We measure the average *root mean squared error* of the 2-channel images compared to the ground truth. In addition to average error, we display cumulative histograms of error values for pixels and images (Figure 2). Cumulative histograms allow us to evaluate the distribution of errors that a colorization makes.

Note that, our error metric is particularly harsh because believable colors different from the ground truth are heavily penalized, while small spatial oddities are not. Still, on a set of images our evaluation provides a comprehensive picture of the performance of colorization.

4.4. Algorithms used for Evaluation

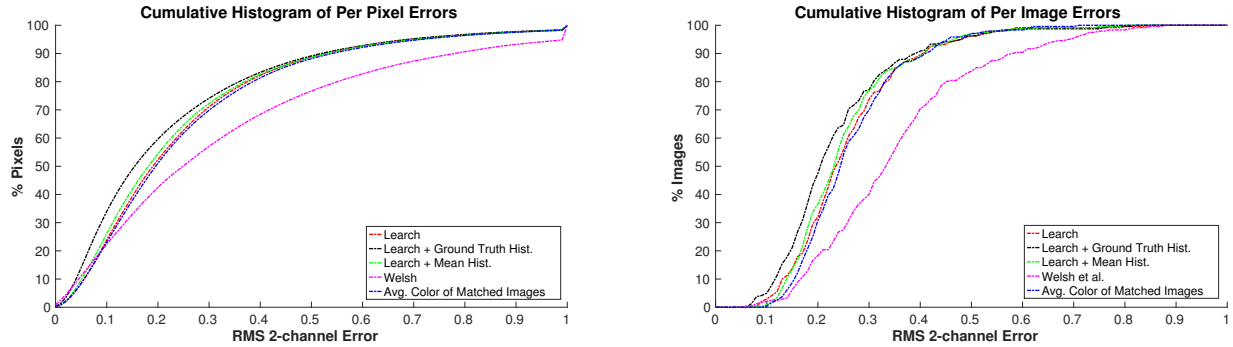
Baseline. We use two colorization methods as baseline: (i) Welsh et al. [19] which transfers color to a grey-level image from a carefully selected reference image. We use the most similar image from the top-k retrieved images as reference image (Section 3.3). This is similar to the method proposed by [13]. (ii) Average color image, where color is transferred by averaging color channels of top-k matching images.

Scene independent training. We train a single LEARCH image regressor from a scene independent training set. We either report the LEARCH result directly, or apply the histogram correction using the ground-truth histogram from oracle.

Scene specific training. We train a LEARCH image regressor for each scene category. We either report the LEARCH result directly, or apply the histogram correction using the ground-truth histogram from oracle or using the scene specific mean histogram (i.e. the normalized histogram of all

	Baseline: With scene label		Training: Without scene label		Training: With scene label					
					Testing: Scene Classification			Testing: Oracle Scene Label		
	Avg. Color	Welsh et al.	LEARCH	LEARCH + GT Hist	LEARCH	LEARCH + Hist		LEARCH	LEARCH + Hist	
						Mean	GT		Mean	GT
Averaged over scenes	0.265	0.353	0.284	0.271	0.270	0.262	0.242	0.260	0.254	**0.236**

Table 1: Comparison of average RMS error for different configurations of our method. Training a regressor specific to each scene shows an improvement over scene independent regressor. This improvement is for both using oracle scene label and scene classification for test images. Histogram correction step reduces errors significantly for both ground truth and mean histograms. Completely automatic configuration – offline training with scene labels, scene classification for test image and histogram correction with mean histogram – outperforms the baseline. Average color image gives good performance on our metric because it does not specifically penalize spatially odd distributions such as isoluminant edges, which are clearly visible in column 2 of Figure 4. Scene-wise split of these results in supplementary.



(a) Cumulative histogram of pixel errors (0.01 wide bins)

(b) Cumulative histogram of image errors (0.01 wide bins)

Figure 2: Our method gives the dual benefit of higher % of pixels and images with low errors. In contrast, Welsh et al. gives lower % of pixels and images with low errors. Though, average image gives similar % of low error pixels as our method, its per image errors are higher than our method. Higher per image errors lead to bad spatial artifacts when using average color (see Figure 4), which our method avoids.

train images). Here we assume that the scene label for test images is provided by the oracle.

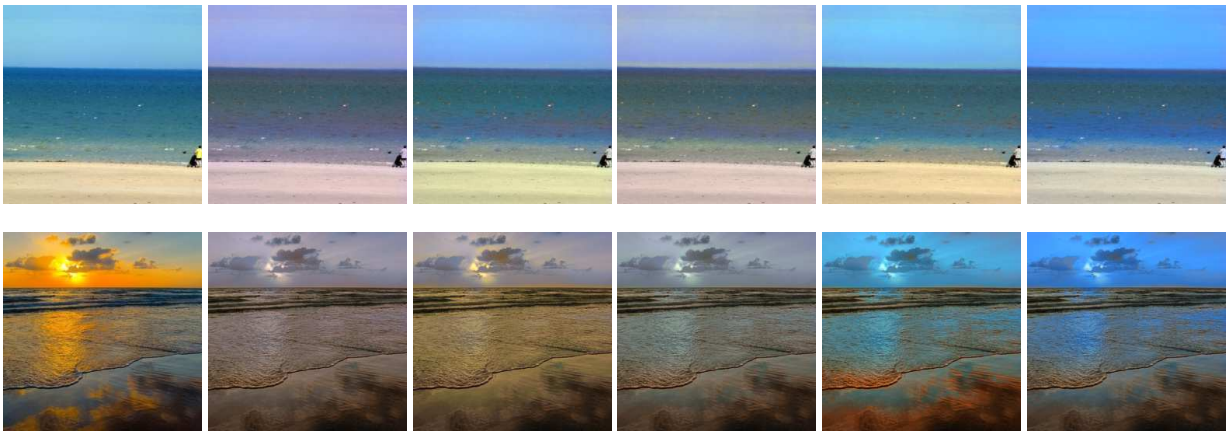
Scene classification while testing. As above, we train LEARCH image regressors for each scene category according to ground truth labels and compute a scene specific histogram. During testing, the scene labels are not provided, instead we predict them and then reconstruct using the regressor associated with the predicted label. Scene specific histograms are used to perform histogram correction.

4.5. Results

4.5.1 Large-scale learned colorization possible

As shown in Figure 4, our method produces good color images as output, in fact use of ground-truth histogram allows us to output strikingly similar looking images to the ground-truth. The output color images with LEARCH followed by correction with mean histogram also show good resemblance to ground-truth. They are free from spatial oddities, unlike Welsh et al. and average color image. Generally large regions are assigned close to ground-truth colors, but smaller regions/objects are assigned spatially coherent but incorrect colors. This is likely because they are not sampled frequently.

We achieve these results by leveraging large datasets of images for learning colorization. This is in stark contrast to the practice of using a single or a few carefully selected reference images for colorization. For large datasets an RMS error provides a valid error metric. Furthermore, at test time we can provide the ground truth histogram, ensuring our prediction shares the same color palette as ground truth. Reasonable quantitative comparisons (Table 1 and Figure 2) can be performed as opposed to previous methodology of qualitatively comparing the output of a few test images.



(a) Ground Truth (b) Learch O/P (c) GT Hist. (d) Mean Hist. (e) Exemplar# 1 (f) Exemplar# 2

Figure 3: Different shading obtained with histogram correction.

4.5.2 Scene information makes a big difference

In Table 1 using scene specific training of LEARCH with an oracle scene label at test time improves performance by 8.4% over scene independent training. An improvement of 4.9% is observed if instead we predict this label by scene classification. The results show that training on a particular scene category, helps the LEARCH objective exploit the underlying structure within the data and learn the optimal function parameters. Scene information is thus vital for learning methods for colorization.

4.5.3 Histogram correction helps

Table 1 compares the impact on LEARCH error when different kinds of histograms are used in histogram correction. To test for best possible improvement with histogram correction, we use the ground truth histogram of the test image. We also report results for mean histogram of all train images, of the given scene. In all experiments, we observe a decrease in error with histogram correction. This demonstrates the importance of optimizing the regressed output to take into account global properties of the scene.

Figure 3 shows use of histogram correction to generate different shades from the same regressed output. Exemplar histograms are sampled from training images of the scene category. Thus, the histogram correction step allows for an authoring pipeline, wherein an expert user modifies the target histogram as needed.

4.5.4 Practical Colorization Methods

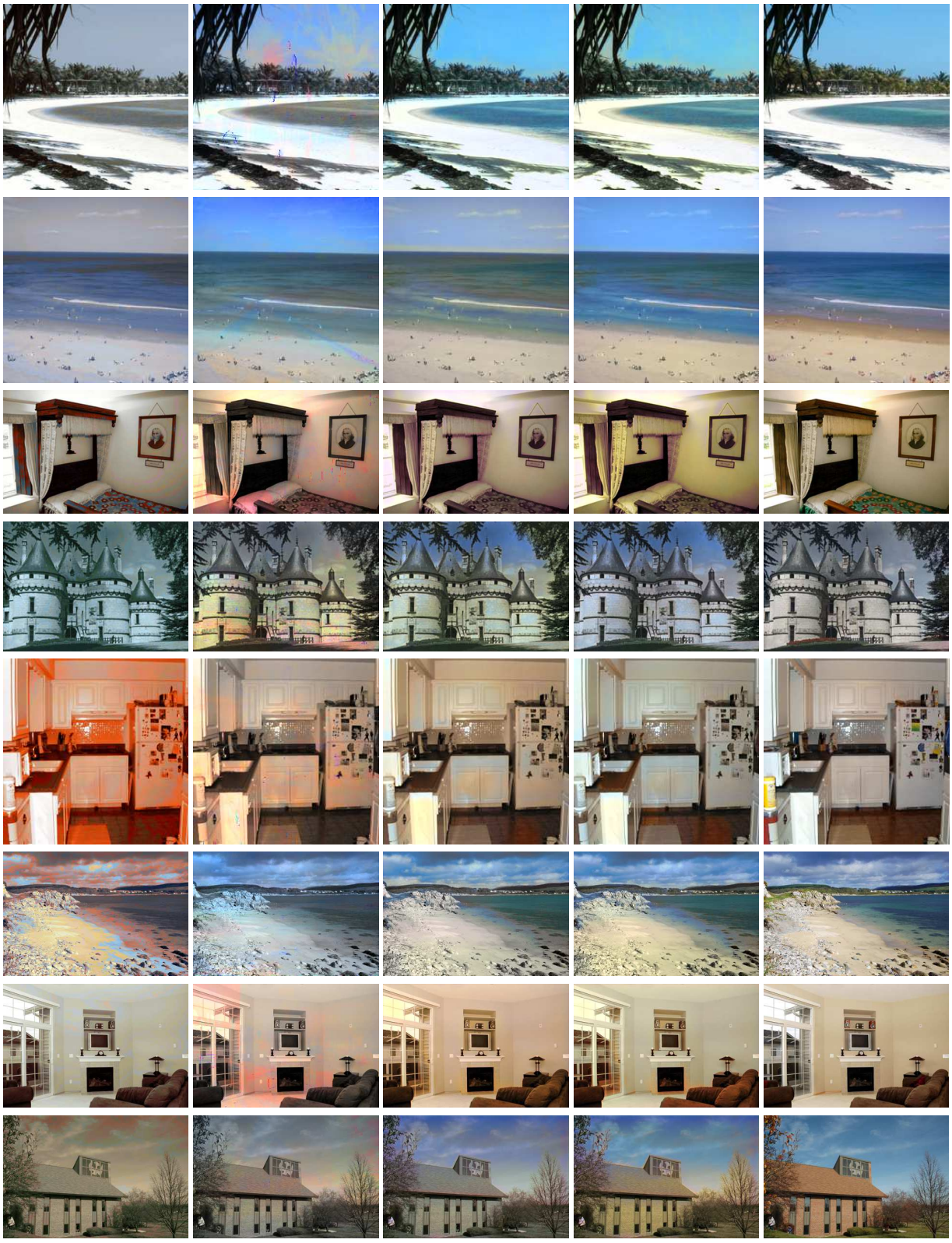
There are two use cases in colorization: either a user wants to colorize one or a handful of images; or a user wants to colorize a movie or a similarly large collection of images. In the first case, it is reasonable to expect the user to provide a scene label. For this, we run scene specific LEARCH using the oracle label and mean histogram. In the second case, it is necessary that the colorization be fully automatic. There are two ways to perform automatic colorization, either we use scene independent LEARCH or we generate scene labels using scene classification and pick the appropriate scene specific regressor. Scene classification LEARCH with histogram correction outperforms scene independent LEARCH (Refer Table 2 for comparison).

Baseline With scene label		**Our Method**		
Avg. Color	Welsh et al.	Scene-indep. Training	Scene-specific Training + Mean Hist. *Classif -ication*	*Oracle Label*
0.265	0.353	0.284	0.262	**0.254**

Table 2: Comparison of errors for practical colorization methods. Our method outperforms baseline, both with and without the availability of scene label for test images.

5. Conclusions

We propose a method to predict colorization using an objective automatically learned by LEARCH. We demonstrate that the method produces spatially coherent colorization, and when augmented with histogram correction produces visually appealing and convincing colorizations. Our method performs best when scene information is available from an oracle, but our fully automated approach, which uses scene classification, produces near optimal results.



(a) Welsh et al. (b) Avg. Color (c) Learch + Mean Hist. (d) Learch + GT Hist. (e) Ground Truth

Figure 4: Qualitative comparison of colorization output of different methods. (Best viewed in color and high resolution)

References

[1] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015. 2

[2] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Trans. Graph.*, 33(4):159:1–159:12, July 2014. 2

[3] A. Bugeau, V.-T. Ta, and N. Papadakis. Variational Exemplar-Based Image Colorization. *IEEE Transactions on Image Processing*, 23(1):298–307. 2

[4] G. Charpiat, M. Hofmann, and B. Schölkopf. Automatic image colorization via multimodal predictions. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 126–139, Berlin, Heidelberg, 2008. Springer-Verlag. 2

[5] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001. 3

[6] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. Ground-truth dataset and baseline evaluations for intrinsic image algorithms. In *International Conference on Computer Vision*, pages 2335–2342, 2009. 2

[7] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH*, 2001. 2

[8] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fields - an efficient, non-parametric approach to image labeling problems. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, April 2012. 2

[9] E. Land and J. J. Mccann. Lightness and retinex theory. *J. Opt. Soc. Am.*, 61(1):1–11, Jan 1971. 1

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, pages 2169–2178, 2006. 4

[11] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43(1):29–44, June 2001. 4

[12] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, Aug. 2004. 2

[13] Y. Morimoto, Y. Taguchi, and T. Naemura. Automatic colorization of grayscale images using multiple images on the web. In *SIGGRAPH 2009: Talks*, SIGGRAPH '09, New York, NY, USA, 2009. ACM. 2, 5

[14] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. 4

[15] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001. 4

[16] T. Pouli and E. Reinhard. Progressive histogram reshaping for creative color transfer and tone reproduction. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, pages 81–90, New York, NY, USA, 2010. ACM. 4

[17] N. D. Ratliff, D. Silver, and J. A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1), July 2009. 2, 3

[18] M. F. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning Gaussian Conditional Random Fields for Low-Level Vision. In *CVPR*, 2007. 2

[19] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. In *SIGGRAPH*, 2002. 2, 5

[20] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva. SUN Database: Exploring a Large Collection of Scene Categories. *International Journal of Computer Vision*, Aug. 2014. 1, 4