Automatic Image Colorization

Harrison Ho (h2o@stanford.edu) and Varun Ramesh (vramesh2@stanford.edu)

***Abstract*— Image colorization is a difficult problem that often requires user input to be successful. We propose and evaluate a new approach to automatically colorize black and white images of nature without direct user input. Our approach uses Support Vector Regressions (SVRs) to predict the color of a region locally, then a Markov Random Field (MRF) to smooth color information across the image. We find that our algorithm is able to recognize textures that correspond to objects such as trees and water, and properly color them.**

I. Introduction

Fig. 1. An example usage of the algorithm.

Image colorization is the process of adding color to grayscale or sepia images, usually with the intent to modernize them. By hand, this can be a time consuming and repetitive process, and thus would be useful to automate. However, colorization is fundamentally an ill-posed problem - two objects with different colors can appear the same on grayscale film. Because of this, image colorization algorithms commonly require user input, either in the form of color annotations or a reference image.

We propose an algorithm to automatically colorize black and white images in restricted circumstances, without requiring any user input. Our technique works by training a model on a corpus of images, then using the model to colorize grayscale images of a similar type. We use Support Vector Regressions (SVRs) and Markov Random Fields (MRFs) in our approach.

In order to define the problem, we represent images in the YUV color space, rather than the RGB color space. In this color space, Y represents luminance and U and V represent chrominance. The input to our algorithm is the Y channel of a single image, and the output is the predicted U and V channels for the image.

II. Background

A. Previous Work

Existing work with image colorization typically uses one of two approaches. The first approach attempts to interpolate colors based off color scribbles supplied by an artist. Levin et al develop an optimization-based approach which colors pixels based on neighbors with similar intensities [4]. Luan et al further build on this work by not only grouping neighboring pixels with similar intensities, but also remote pixels with similar texture [6]. This addition was designed to facilitate colorizing images of nature.

The second approach to colorization has the user supply a reference image. The algorithm then attempts to transfer color information from the reference onto the input grayscale image. These algorithms typically work by matching up pixels or image regions by luminance. Bugeau and Ta propose a patch-based image colorization method that takes square patches around each pixel [2]. They then extract various features such as variance and luminance frequencies in order to train a model. Charpiat et al develop a color prediction model which takes into account multimodality - rather then returning a single prediction, they predict a distribution of possible colors for each pixel [3]. They then use graph cuts to maximize the global probabilities and estimate the colored image.

For our algorithm, we expand on the second approach, training a model over a corpus of images rather than a single image. Our goal is that, once the model is trained, users will not need to provide any input at all to the algorithm.

B. Related Work

Our work here is largely inspired by the work of F. Liu, et. all, who used Deep-Convolutional Neural Networks (DCNNs) to estimate a depth channel given individual monocular images [5]. Their approach constructs an MRF over the superpixels of a given image and estimates the potentials of the field using a DCNN. We apply a similar approach, modeling an MRF over the superpixels in an image, and using SVRs to provide local estimates.

III. Dataset and Features

A. Dataset

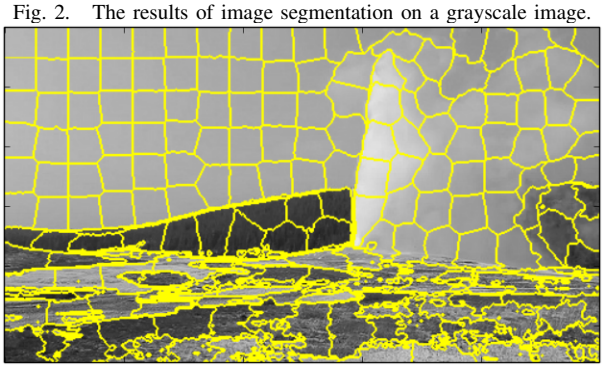
To train and test the classifier, we decided to focus on images of Yellowstone National Park. We downloaded photos on Flickr with the tags “yellowstone” and “landscape,” as this gave a good selection of photos that did not include animals, humans, or buildings. We scaled the images in our dataset to a constant width of 500 pixels, and eliminated any already grayscale photos. Additionally, we split the images into a training set (98 images) and a test set (118 images).

B. Image Segmentation

In order to constrain the problem, we segment the images into sections using the SLIC superpixel algorithm [1]. We chose the SLIC algorithm over other segmentation algorithms due to its effectiveness in creating uniform, compact

segments. We used the implementation in scikit-image [8]. Figure 2 shows the results of SLIC clustering on one of our test images.

Fig. 2. The results of image segmentation on a grayscale image.



C. Image Representation and Training

For our model, instead of predicting color pixel-by-pixel, we predict two real values (U and V channels) for each segment of the image. This allows us to color segments based on image structures. Additionally, we assume that the U and V channels are independent given an image segment.

We utilize Support Vector Regressions, a generalization of Support Vector Machines. Given training vectors $\phi(x^{(i)}) \in \mathbb{R}^p$ for $i = 1...m$ and a vector $y \in \mathbb{R}^m$, a Support Vector Regression specifies the following minimization problem, where C and ϵ are chosen constants:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^\top w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$$
$$\text{subject to } y_i - w^\top \phi(x^{(i)}) - b \leq \epsilon + \zeta_i,$$
$$w^\top \phi(x^{(i)}) + b - y_i \leq \epsilon + \zeta_i^*$$
$$\zeta_i, \zeta_i^* \geq 0, i = 1, ..., n$$

This optimization problem can be performed efficiently using scikit-learn's SVR function, which performs Epsilon-Support Vector Regression optimization with an underlying liblinear implementation [7].

For our model, we train two separate SVRs - one for each output channel. To extract feature vectors for each of the image segments, we take the square of 10×10 pixels around each centroid. We then perform a 2D Fast Fourier Transform (FFT) on these squares, giving us our feature vectors $\phi(x^{(i)})$. For both regressions, the feature vectors $\phi(x^{(i)})$ are used as the input, while the average U and V values of the segment $x^{(i)}$ are used as the output. We used the default Gaussian kernel for training the SVRs.

Fig. 3. Conversion of superpixels to feature vectors.

D. Image Testing and ICM Smoothing

To estimate the chrominance values of a test image, we perform the same process of segmentation and feature extraction described above. We then run the two SVRs over the segments, giving us an initial estimate of the color.

To smooth out the shading of similar, adjacent superpixels, we model every test image as two Markov Random Fields (MRFs), with one MRF per predicted channel. We represent local potentials for the hidden chrominance value of a superpixel as a Gaussian $\sim \mathcal{N}(\mu_i, \sigma)$, where μ_i is the result of the SVR on that superpixel. We additionally represent neighboring potentials between hidden chrominance values of adjacent superpixels with a distance function.

The total energy E_i of a single superpixel $x^{(i)}$ with a hidden chrominance c_i can hence be represented as

$$E_i = \frac{||c_i - \mu_i||_2^2}{2\sigma^2} + \gamma \sum_{x^{(j)} \in N(i)} ||c_i - c_j||_2^2$$

where γ weights the relative importance of neighboring pixels compared to single pixels, and $N(i)$ denotes the set of adjacent superpixels to $x^{(i)}$. For any two pairs of adjacent superpixels $x^{(i)}$ and $x^{(j)}$, we only include $x^{(j)}$ in the set $N(i)$ if $||\phi(x^{(i)}) - \phi(x^{(j)})||_2$ lies below a threshold value. Finally, to minimize the total energy of the MRF, we run Iterated Conditional Modes (ICM) until convergence.

At this point, we have the original Y channel and estimated U and V channels for a target image. Converting these channels to the RGB space yields our final colorization estimate.

Fig. 4. Flowchart of the algorithm.

E. Hyperparameter Tuning

We evaluated the hyperparameters of the SVR and ICM over a range of values, and selected those that produced the minimum error. In our formulation we define the error as the average distance between our predicted RGB values and the actual RGB values. After evaluation, we selected $\epsilon = .0625$ and $C = .125$ for the SVRs, and $\gamma = 2$ for the ICM.

IV. Results

Fig. 5. The original color image, the de-colorized input, and our algorithm's prediction, respectively.

Despite the ill-posed nature of the problem, our algorithm performs well on the test set. We display some of our results in Figure 5. While the method does not reproduce all colors correctly, it in general produces plausible coloring results. Moreover, our method successfully colors environmental features differently. In the third example, our method notably shades the reflection of trees in the water differently from the actual trees.

In Figure 6, we compare the average error for the grayscale image with the original, the average error prior to running ICM smoothing, and the average error after running ICM smoothing. Implementing the SVRs decreased the error by 68.1%, while implementing the ICM smoothing decreased the error by an additional 6.5 percentage points. Visually, the ICM smoothing helps to reduce anomalies within an image; as an example, in Figure 4, segments which are erroneously colored a red shade are changed to blue and white shades in our prediction.

Fig. 6. Average Error Comparison Before and After ICM Smoothing

Before Processing	With SVR only	With SVR and ICM
$1.83 * 10^{-3}$	$5.84 * 10^{-4}$	$4.81 * 10^{-4}$

Our method still has room for improvement. Our method fails to reconstruct the full range of colors present in the original image, and fails to include more shades of yellow and brown. This may be due to the limited range of colors present in our data set, which primarily consists of shades of blue and green. Training the model on images with a balanced variety of colors may alleviate this behavior.

Another issue with our method is that shades intended for one section of the image often bleed into neighboring sections. For example, in the third image, the water adjacent to the large tree on the right is shaded similarly to the tree itself. This may be because for small superpixels, taking 10×10 squares around the centroid of the superpixel often takes in more pixels than contained within the superpixel itself. Thus, the estimated color is strongly affected by neighboring superpixels, which can introduce error if neighboring superpixels are significantly different shades of color.

Finally, our method often produces images with low saturation, an issue particularly noticeable with the second example. This may be a result of our assumption that each superpixel has a single most likely coloring, despite the fact that superpixels may take on one of several, equally plausible colorings. This assumption can cause the SVRs to average the chrominance value outputs, decreasing the saturation of the estimated color. A multimodal approach may help to increase the saturation of the predicted colors, which has been explored in depth by Charpiat et al [3].

V. Conclusion

Automatic image colorization by training on a corpus of training data is a feasible task. We envision a potential system where a user can colorize grayscale images by simply declaring relevant tags for a target image. A hypothetical system can then automatically retrieve photos with the relevant tags and train models specific to these tags.

Our project code is available at `https://github.com/harrison8989/recolorizer`.

VI. Future Work

There are a number of measures than can be taken to improve on the performance of the algorithm.

- Currently, we simply use a Gaussian distribution for local potentials in our MRFs - we use the SVR prediction as the mean, and a fixed variance. A Gaussian is too simplistic, as a given texture can be indicative of two or even three colors. For example, leaves vary between green and red, but are unlikely to be other colors. In order to capture this, we need to predict a distribution of values for each color, not just a single value. SVRs are insufficient for this task.
- We can represent images as a pixel-wise MRF, with local potentials on each pixel and pairwise potentials between adjacent pixels. With this model, we may be able to prevent color bleeding and provide other visual improvements, such as allowing backgrounds to poke

through trees. This could also be coupled with pixel-wise SVR prediction, and the elimination of the image segmentation step.

- While SVRs were reasonably successful, many state of the art algorithms use DCNNs to great effect. Early experimentation with DCNNs did not prove promising for us, but adjustments in layer definitions and an increase in training data might make them more viable.
- We can incorporate additional features in the SVRs to increase the model's expressiveness. Bugeau et al incorporate three features: the variance of a pixel patch, the 2D discrete Fourier transform, and a luminance histogram computed over the patch [2].
- We currently use the YUV color space to separate luminance and chrominance. Alternate color spaces exist that may do this more effectively (CIELAB).

REFERENCES

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[2] Aurélie Bugeau and Vinh-Thong Ta. Patch-based image colorization. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3058–3061. IEEE, 2012.

[3] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. Automatic image colorization via multimodal predictions. In *Computer Vision–ECCV 2008*, pages 126–139. Springer, 2008.

[4] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, August 2004.

[5] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *CoRR*, abs/1502.07411, 2015.

[6] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural image colorization. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 309–320. Eurographics Association, 2007.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.