Generating Images with Neural Networks

RORY SOIFFER, University of Washington, USA
JIANYANG ZHANG, University of Washington, USA

Artists often want to edit global properties of an image such as hue and saturation without changing how it appears in small areas. Current tools usually don't attempt to prevent modifications to the perceptual content of an image. We propose a deep learning based method for color transfer problem that modifying the image to change the desired parameter while attempting to keep the detected high-level features as constant as possible and requires no user input besides the input image and the target color.

CCS Concepts: • **Computing methodologies → Neural networks**; **Image processing**;

Additional Key Words and Phrases: style transfer, color transfer, image processing, image parameter editing, neural network, Generative Adversarial Network

ACM Reference Format:
Rory Soiffer and Jianyang Zhang. 2018. Generating Images with Neural Networks. 1, 1 (March 2018), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

One common task in digital image editing is changing the overall color of an image. Applications for this task are commonplace: modifying an image to better fit with a color scheme, copying one image to another while making the content blend naturally, and so on. It is also generally desireable to perform this edit so naturally that the audience is not able to notice that an edit occured. As such, want to create a tool that can perform color transfer between images while maintaining the perceptual realism of the image.

Most current tools for color transfer work on simple pixel manipulations, such as applying a linear transformation to the pixels in some color space. Other more advanced tools try to take statistical or semantic properties of the image into account. These methods often require user input to annotate images and colors. In addition, they often fail to understand higher-level properties of an image, and so produce strange-looking and unrealistic results.

Recent advances in deep learning allow for computers to automatically determine the perceptual content of an image, with no user input at all. Our goal is to apply these ideas from deep learning to the problem of color transfer. We hope to create a tool that is able to transfer color between images while producing perceptually realistic results.

Authors' addresses: Rory Soiffer, University of Washington, Paul G. Allen School of Computer Science & Engineering, 185 E Stevens Way NE, Seattle, WA, 98195, USA, rorys4@cs.washington.edu; Jianyang Zhang, University of Washington, Paul G. Allen School of Computer Science & Engineering, 185 E Stevens Way NE, Seattle, WA, 98195, USA, jianyz@cs.washington.edu.

2 RELATED WORK

Color transfer is an active research area in the communities of image processing and computer graphics. We review some previous work proposed for color transferring between images, together with several deep learning methods for image processing and image texturing that related to our approach.

2.1 Statistical Color Transfer Methods

In the literature, early color transfer methods are mainly statistics-based algorithm. Reinhard et al. introduced a method that model color distributions as Gaussian distributions. They define color transfer as transfer the means and standard deviations along each channels in $L\alpha\beta$ color space using per-axis scaling and shifting from one image to another [Reinhard et al. 2001]. Since the RGB color space is highly correlated and does not account for human-perception, they first transform pixel values from RGB space to $L\alpha\beta$ color space according equation 1. The $L\alpha\beta$ color space can minimizes correlation between channels and is proved to be more perception-based [Ruderman et al. 1998]. Then, they calculate the mean and standard deviations separately on each of the three channels. They transfer these statistics from the input image to the output image as scales and offsets along each axis for each pixel in its $L\alpha\beta$ color space. Last, they transform each pixel value back to RGB space. For images contains multiple color regions, users can provide additional annotations to divide the image into color clusters.

The major difficulty of color transferring often lies in the correct segmentation of semantic regions of the images. Levin et al. derived an optimization framework for colorized a gray scale image or movie by assuming that neighboring pixels in space-time with similar intensities should be assigned with similar colors [Levin et al. 2004]. This approach uses a using a quadratic cost function and obtain an optimization problem that can be solved efficiently using standard techniques. The users are required to annotate the image with color scribbles, and the indicated colors are automatically propagated in both space and time to produce a fully colorized image or sequence. In later works, Huang et al. extended this method to reduce color blending over object boundaries with edge detection techniques [Huang et al. 2005]. Yatziv et al. extended this method using weighted distance between user annotations to produce a color blending [Yatziv and Sapiro 2006].

Zeng et al. proposed a method that transfer color based on local semantic segmentation [Zeng et al. 2011]. This approach parses an input image into semantic objects according to different material properties by human interactions. Based on the segmentations, a proper reference image is selected from the library. The dominant colors of the input image and reference image are computed by clustering, and then transferred on to the input image.

While these approaches all produced promising results in color transferring, the problem of these approach is, they all require considerable amount of human interactions, especially when the color setting of the image is complex. A lot of annotations need to be specified by users and such annotations may not result in a desired color segmentations of the image. In addition, these method is restricted to linear color transformations but often time color transfer procedures require non-linear matching between colors.

2.2 Deep Learning Based Methods

Techniques based on deep learning has been widely used in computer graphics and computer vision tasks, such as image processing and image generation.

Deep Dream is an application based on deep-learning convolutional neural networks (CNNs) [Krizhevsky et al. 2012] that blend visual qualities from source images to create a new output image[Mordvintsev et al. 2015]. Deep Dream algorithms uses pre-trained CNNs to perform gradient descent on a target image to maximize the perceptual features, generating a transformed version of a source image with perceptual qualities from target image.

Generative Adversarial Networks (GANs) have achieved significant results in image generation [Goodfellow et al. 2014] and image to image translation [Radford et al. 2015]. The basic idea behind GAN is to simultaneously train two adversarial networks, a discriminative network and a generative network, competing against one another. The generator produces fake samples by passing random noise variables $p_z(z)$ as input through a multilayer perceptron G. The discriminator D, also a multilayer perceptron, learns a loss by distinguishing between the output samples of G from real world samples. We use $D(x)$ represents the probability that a sample x came from the real world data. The generator receives the gradient of the output of D with respect to the fake sample and uses it to minimize the adversarial loss $\log(1 - D(G(z)))$. In another word, the two networks play a two ply min-max game, and the utility function can be written as

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

[Goodfellow et al. 2014]. This method have led to great success because the generator produces imitation of data and the discriminator captures the visual realism of generator output. We thus adapted this architecture to support color transferring. In addition, we added a color loss and extended the architecture to convolutional network, which will be elaborated in section 3.

Yi et al. extended the idea of GANs to unlabeled and unpaired image-to-image translation [Yi et al. 2017] inspired by dual learning in machine translation [He et al. 2016]. The architecture employs two GANs, a primal GAN and a dual GAN. Given two sets of unlabeled and unpaired images from two domains U and V, the primal GAN learns a generator $G_A : U \longrightarrow V$ and a discriminator D_A that discriminates output of G_A from samples in V. The dual network trains an inverse generator $G_B : U \longrightarrow V$ and a discriminator D_B. During the training, an image $u \in U$ is transferred to domain V with G_A. $G_A(u,z)$ is then transferred back to domain U with G_B. In addition to the adversarial loss, the architecture minimizes two reconstruction losses $\|G_A(G_B(v,z^{'}),z)) - v\|$ and $\|G_B(G_A(u,z),z^{'} - u)\|$. The reconstruction losses force the inputs to be re-constructable from outputs which strengthens feedback signals that encodes the targeted distribution [Yi et al. 2017]. Zhu et al. proposed a similar loss function, cycle consistensy loss, as a way of using transitivity in GAN network to supervise training [Zhu et al. 2017]. Both methods were proved to have significant performance improvement over the work of art, which can be attributed to the addition of the reconstruction loss. We thus adapted the reconstruction loss into our approach as a similarity loss to capture backwards color transferability between input and output images, which will be elaborated in section 3.

Building on ideas from these many previous works, we developed a generative adversarial based approach for color transfer. Our main contribution is applying new ideas from deep learning to the old problem of color transfer. The method changes the color of an image in a way that preserves the perceptual content of the image and require no user input besides the input image and the target color.

3 APPROACHES

We attempted two approaches for color transfer that preserves the perceptual content of the image. Our first approach performs gradient descent on an image to minimize the difference between the features detected by a convolutional neural network. Our second approach trains a generative adversarial network to create realistic-seeming images with the desired target color.

3.1 Feature Matching

Our first approach tries to change the color of an image while matching the features detected by a neural network to the original image. We implemented this method in a python script, which we have made available in our project's code repository. Essentially, this method starts with some input image, defines a loss function that measures the quality of the output, and then performs gradient descent on the input image in order to minimize the loss function. Our loss function includes the following components:

- Color Loss
- Feature Matching Loss
- Blur Loss

The **Color Loss** measures the difference between the current color of the image and the target color the program is attempting to reach. It is defined as $|C - C_T|^2$, where C is the current color of the image and C_T is the target color. Both colors are expressed as 3-d vectors in the RGB color space. As this loss decreases, the image approaches the target average color.

The **Feature Matching Loss** measures the difference between the current perceptual features of the image and the perceptual features of the input image. Let $N(I)$ be a function that takes as input an image and returns as output a vector of features detected by the neural network. Then the Feature Matching Loss is defined as $|N(I) - N(I_0)|^2$, where I is the current image and I_0 is the input image. As this loss decreases, the perceptual content of the image (as measured by the neural network) approaches the perceptual content of the input image. In our implementation, we used a Tensorflow copy of Google's Inception network as our neural network, and we extracted features from the 'mixed3' layers.

We found that a simple change to the Feature Matching Loss to make it scale-invariant drastically improved the quality of the output. The intuition behind this change is that the neural network only detects features at a given scale, so if we want to detect features of any scale, we need to run the neural network multiple times on scaled version of the input image. We created several copies of the input image (which we call *octaves*), each downscaled by a factor of 2^n for $n \in [1, ..., 3]$. We then computed the Feature Matching Loss for each octave, adding together the results, while more heavily weighting the loss of higher-resolution images. Our final definition for the Feature Matching Loss was

$$\sum_{i=0}^{3} |N(D_i(I)) - N(D_i(I_0))|^2 * (.9)^i$$

where I is the current image and I_0 is the input image, and $D_i(I)$ is a function that downscales the image I by a factor of 2^i. To allow for backpropagation of gradients, we defined $D_i(I)$ via strided convolution with a gaussian kernel to smoothly blur and downsample the image.

The **Blur Loss** measures how much high-frequency noise is in the current image. This term in the loss is required to avoid generating an image with a lot of noise that happens to randomly match the filters in the neural network. Let $B(I)$ be a function that takes as input an image and returns as output a blurred version of that image (in our case, we used a simple 3x3 constant matrix to blur the image). Then the Blur Loss is defined as $|B(I) - I|$, where I is the current image. As this loss decreases, the image contains less and less high frequency noise.

One advantage of this method is that we can produce a wide variety of effects simply by changing the terms of the loss function. One especially interesting additional term was the **Color Variance Loss**, which measures how much each RGB component varies across the image. From a mathematical perspective, adding this term makes the program approximate the color of an image as a pseudo-gaussian distribution instead of as a single average value. Let $V(I)$ be a function that takes an input an image and returns as output the variance of each RGB component in the image. We got optimal results by defining $V(I) = mean[\sqrt{abs(I - C)}]^2$, where C is the average color of I, though there are many other definitions of the color variance that performed about as well. Then the Color Variance Loss is defined as $|V(I) - V(I_0)|^2$, where I is the current image and I_0 is the input image. As this loss decreases, the image approaches the target color variance. By changing the target color variance, we can generate images with increased or decreased saturation.

3.2 Generative Adversarial Network

Our second approach used a generative adversarial network to change the color of an image while making the output of the network seem as realisitic as possible. We implemented this method in a python script, which we have made available in our project's code repository. This method uses a standard GAN to generate images, but adds additional terms to the loss function to constrain the GAN to generating images with the desired target color. Specifically, we add a term to the generator loss that measures the difference between the color of the output image and the target color the program is attempting to reach. Similarly to the Color Loss in the Feature Matching approach, the generator's color loss is defined as $|C - C_T|$, where C is the color of the output image of the generator and C_T is the target color. Both colors are expressed as 3-d vectors in the RGB color space. As this loss decreases, the output of the generator approaches the target average color.

We based our implementation on the Improved Training for Wasserstein GAN sample code here: https://github.com/wiseodd/generative-models. We used the Adam Optimizer with a learning rate of 10^{-5}, and trained the discriminator for 5 steps for each training step of the generator. We used a minibatch size of 32. We trained our network on 24x24 crops of CIFAR-10 images, and later on 64x64 crops of a small subset of Imagenet images.

Our generator network took as input the starting image, a noise vector, and the target color, and returned as output another image of the same size. Our generator network consisted entirely of four 5x5 convolution layers with 64 filters each. Our discriminator network took as input an image, and returned as output a scalar representing its confidence that the image was real, as opposed to a an output of the generator (positive numbers mean real, negative numbers mean fake). Our discriminator consisted of three 5x5 convolution layers with 64 or 128 filters, each followed by a max pooling layer. These layers were followed by two fully-connected layers with 256 and 128 neurons each.

We also tried including a **reconstruction loss** term in the generator network to measure how well the network could run in reverse to recreate the original image. Adding a reconstruction loss term has been shown by methods like DualGAN and CycleGAN to greatly improve the performance of GANs for image-to-image translation. While color transfer is a different problem than image-to-image translation, we felt that there was enough similarily between the two that reconstruction loss might be helpful. Let $G(I, C)$ be a function that takes as input an image and a target color and returns the output of the generator network for that image and target color. If we use the generator network to transfer the original color back to the image, we can define $I_{recon} = G(G(I, C_1), C_0)$, where C_0 is the average color of I and C_1 is any color. Then the reconstruction loss is defined as $mean(abs(I - I_{recon}))$. As this loss decreases, the output of running the generator twice approaches the original image. In practice, we did not include the reconstruction loss when training the GAN, as we found it had very little effect on the output.

4 PERFORMANCE EVALUATION

The feature matching approach worked fairly well overall. It was successfully able to change the color of images while preserving their perceptual content. For example, in Figure 1 to Figure 3, note that the output of this process has much more vibrant colors than the baseline image. Also note that the sky and clounds in the output image are a clear blue and white respectively, while in the baseline image they're a strange shade of green. Because it corrects these anomalies, the output image seems subjectively more realisitic and higher quality than the baseline image.

However, the feature matching approach suffers from some significant drawbacks. For one, the script is very slow, as it's performing gradient descent on an image with millions of pixels. This is mathematically very similar to performing gradient descent on a neural network with millions of parameters, and as expected, the script takes a similar amount of time to run - roughly a few hours per

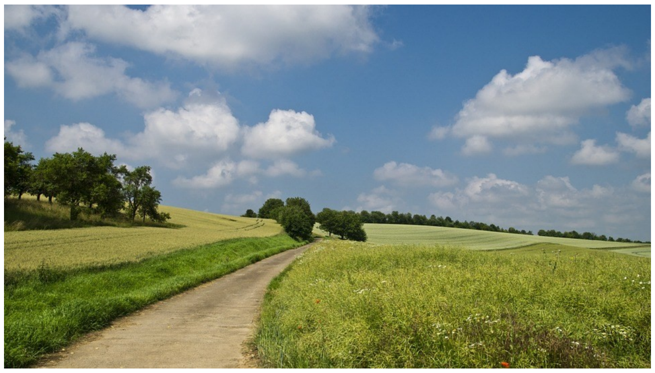

Fig. 1. Start image

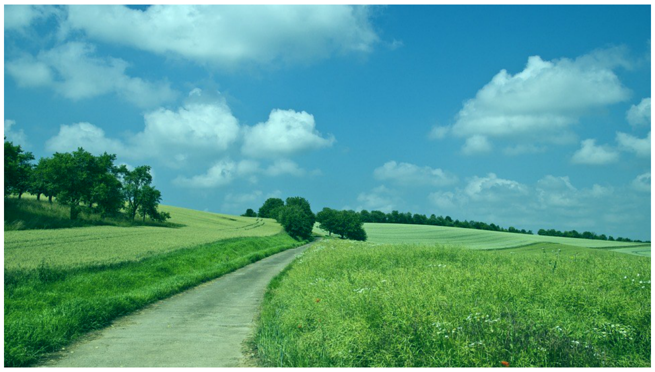

Fig. 2. Edited image

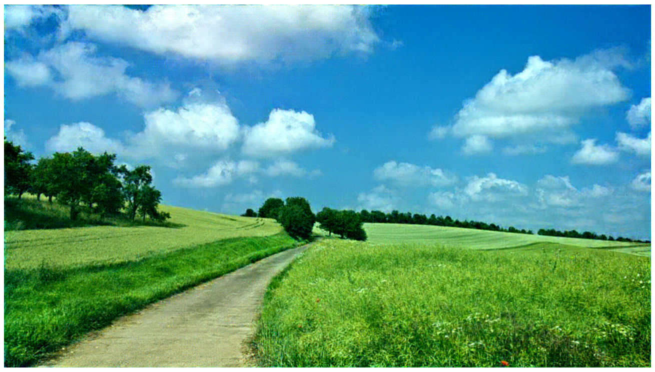

Fig. 3. Output image from feature matching network

image. In addition, the script often produced noticeable artifacts in the form of noisy rainbow patterns. This is likely a failure of the Blur Loss described above to completely remove high-frequency noise from the image.

The GAN approach did not work very well. When training, the network failed to converge to a stable solution on any run. The discriminator and generator loss were constantly jumping up and down, even after thousands of batches. On some runs, the losses would diverge to NaN. Working around these training challenges took a lot of effort restructuring the network and trying new possibilities.

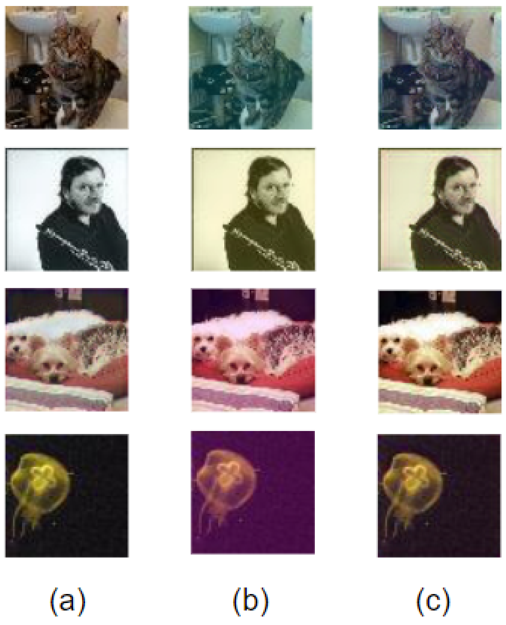

Fig. 4. GAN output: (a) start image, (b) edited image, (c) GAN output

In the end, the network had partial success in training. The results appear somewhat correct, and the generator and discriminator losses were usually better than a baseline comparison. However, the results are noticeably low quality. In some cases, the network completely failed to match the target color of the image. The network sometimes even made the color of the image farther from the target! Below are a few outputs of the GAN:

The second row in figure 4 shows a success case of the network. It attempts to change the overall color of the image to be more yellowish. The GAN strikes a balance between matching this color and preserving the realism of the image. The reader can see that the output of the network appears both yellowish and perceptually realistic.

The third row in figure 5 shows a failure case of the network. The network was asked to change the color of the image to be more purplish, but instead, it changed to the color of the image to be more grayish. This is likely a case of the color loss not being strong enough to push the output of the network towards the correct target color.

5 CONCLUSIONS

In this work, we explored two different methods based on deep learning to solve the color transferring task. The feature matching approach and generative adversarial based network. The feature matching network is adapted from the Deep Dream architecture. Our main contribution is by incorporating the blur loss to reduce

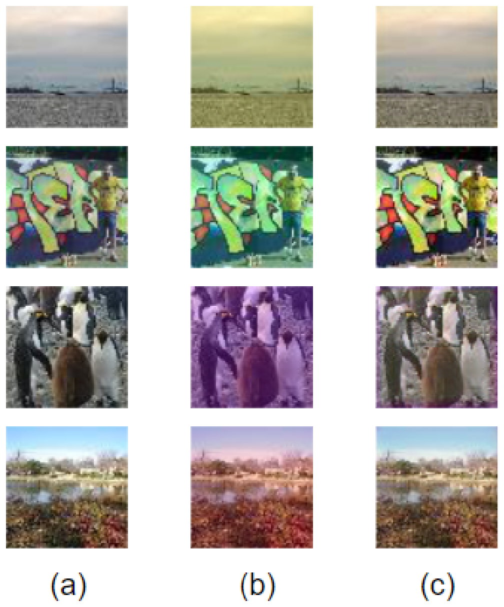

Fig. 5. Failure cases: (a) start image, (b) edited image, (c) GAN output

noise in the image and the feature matching loss to capture perceptual features at different scales together with the color loss to modifying the image parameter while matching high level perceptual information of the image. Our second approach is inspired by Generative Adversarial Networks. We introduced a novel loss function consists of three parts, a discriminative loss, a reconstruction loss and a color loss, in order to solve the color transfer problem.

Comparing the outputs and performance of the two architectures, we can see the feature matching method is simple to understand and implement and can be easily adapt to other image editing problems such as modifying blurriness or brightness of an image. The GAN based network incorporate more information and construct a much more complex architecture. It gradually converges to the equilibrium of modifying the image parameters and maintain the image similar to the start image. But due to the complex structure of GAN, the system only works well for low resolution images such as MNIST [LeCun et al. 2010] or Cifar-10 [Krizhevsky et al. 2014] but will generate considerable amount of parameters for a larger dataset.

6 FUTURE WORK

In the future, we plan to train the GAN with higher-resolution images. Due to the complex structure of GAN network, we only had chance to tune the network with low-resolution images such as crops from ImageNet [Krizhevsky et al. 2012] and Cifar-10[Krizhevsky et al. 2014]. Experimenting with images of different resolutions and different scales may lead to different results. Furthermore, if we want this approach to be useful in practice as an image-editing tool, it needs to be able to produce high-quality output on images of any resolution.

Another challenge we could address with future work is the difficulty of objectively evaluating the output of our approaches. Currently, we mostly evaluate the results visually, which is easy to perform, but is subjective and impossible to exactly measure. In the future, we could conduct a more formal user study on visual realism of the synthesized images as a quantitative metric for evaluation.

Beyond simply color transfer, there are many other probelms in computer graphics and vision that our approaches could be applied to. For instance, we could train our models to modify the saturation, blurriness, or brightness of an image. The feature matching approach is very general, and could be applied to a wide variety of style transfer problems. The generative adversarial network method is more difficult to generalize, but it could be possible to do so with a more sophisticated model and a larger dataset.

ACKNOWLEDGMENTS

We are fortunate and thankful for all the advice and guidance we have received during this project from our instructors Alex Colburn, Deepali Aneja, Gary Faigin, and Barbara Mones.

REFERENCES

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*. 820–828.

Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. 2005. An adaptive edge detection based colorization algorithm and its applications. In *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM, 351–354.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html* (2014).

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010).

Anat Levin, Dani Lischinski, and Yair Weiss. 2004. Colorization using optimization. In *ACM Transactions on Graphics (ToG)*, Vol. 23. ACM, 689–694.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June* 20, 14 (2015), 5.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. 2001. Color transfer between images. *IEEE Computer graphics and applications* 21, 5 (2001), 34–41.

Daniel L Ruderman, Thomas W Cronin, and Chuan-Chin Chiao. 1998. Statistics of cone responses to natural images: implications for visual coding. *JOSA A* 15, 8 (1998), 2036–2045.

Liron Yatziv and Guillermo Sapiro. 2006. Fast image and video colorization using chrominance blending. *IEEE transactions on image processing* 15, 5 (2006), 1120–1129.

Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint* (2017).

Kun Zeng, Ruimao Zhang, Xiaodan Lan, Yan Pan, and Liang Lin. 2011. Color style transfer by constraint locally linear embedding. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 1121–1124.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593* (2017).