Automatic Image Colorization Using Image Analogies

Stephen Cook
Princeton University
Princeton, NJ 08544
stcook@princeton.edu

Abstract

*This paper describes a framework for automatic colorization of grayscale images. The framework involves four stages: a **classification phase**, in which the image is categorized and given a label based on its contents; an **exemplar search phase**, in which we find the nearest neighbors within the category of an image dataset; the **image analogies phase**, in which the image analogies algorithm is carried out using the nearest neighbor exemplars as training data; and a **colorization combination phase**, in which all colorizations from the image analogies are combined in a visually appealing manner. The use of image analogies permits a straightforward color filter to be learned for the grayscale image. Since this method relies heavily on the range of colors available in the training pair, multiple exemplars are used to provide a fuller palette for the grayscale image. A careful combination of the resulting images creates an accurate colorization.*

1. Introduction

Often times it is desirable to give color to traditionally black-and-white images, such as historical photographs, medical imaging scans, or photos taken with a black-and-white camera. Color can give life to remastered photos and make them easier for viewers to understand. Color can help provide depth cues and other useful visual information. There is a wealth of black-and-white images in existence to motivate an accurate colorizer. In this paper we explore a framework for automatic colorization of grayscale images using image analogies. In particular, we attempt to solve the following problem:

Problem("COLORIZE IMAGE"): Given a grayscale image A (and access to an image dataset), synthesize a new image A' such that it is a valid and aesthetically pleasing colorization of A.

In other words, we aim to interpolate color from only black-and-white information. This is a difficult problem in the sense that we are trying to guess *something* from *nothing*. The best colorization algorithms generally to try figure out *what* the image is, then use one or more "training" images to transfer similar colors to the correct areas of the picture.

However, for colorization there is never exactly a "correct" answer. Multiple colorizations can all be valid. For instance, a result in which a car is colored blue can be just as valid as one in which it is colored red. The sky above a city may be colored with a blue gradient or hues of a sunset. The color of grass may appear green instead of yellow. This makes evaluation of colorization results highly subjective and rather difficult. Since there is no true "best" answer or benchmark to meet, attempting to fine-tune and improve a colorization system is tricky.

Generally there are four stages to our COLORIZE IMAGE function:

1. **Classification phase**: The input image is classified into a category and given a label (e.g. a black-and-white photo of New York is labeled as "city").

2. **Exemplar search phase**: We search through all color images in that category in an image dataset. The top nearest neighbors to the input image are chosen as exemplars (e.g. we find two color pictures of Los Angeles and San Francisco that are similar to our image).

3. **Image analogies phase**: We obtain a grayscale version of each exemplar image to create a training pair: grayscale $\rightarrow$ color. Each training pair is used in an image analogy process to create a colorization of the input image (e.g. we use grayscale San Francisco $\rightarrow$

Figure 1: An image analogy. Hertzmann *et al.*'s algorithm for learning and apply a filter between pairs of images[1].

color San Francisco as a guide of how to color our image of New York).

4. **Colorization combination phase**: We combine the resulting colorizations of each image analogy (e.g. we take the colorization of New York in which we use San Francisco as a guide and combine it with the one in which we use Los Angeles as a guide).

This purpose of this project is mainly to evaluate the performance of using Hertzmann *et al.*'s image analogies algorithm as the core of a colorization framework [1]. Image analogies is unique in that it is capable of learning a filter applied from one image to another; that is, when it examines an unfiltered image A and a filtered image A', it is able to "learn" the transformation and apply the filter to any new image B to create a filtered version B'. While this allows for many interesting applications such as copying image processing filters, artistic filters, texture synthesis, and super-resolution, we are interested in particular in colorization.

In its most basic form, image analogies uses luminance information to analyze the change from image A to A'. Since luminance is one of the few features available to us in grayscale images, this is a good fit. However, this algorithm performs poorly on many colorization cases for that exact reason: it is limited only to luminance feature vectors and thus cannot "learn" the filter transformation successfully. Section 6 discusses the limitations of using image analogies and suggests further areas of development.

2. Related Work

There have been numerous proposed methods to colorize grayscale images. For example, Welsh *et al.*'s [6] method matches the color "mood" of a target image by matching luminance and texture information. Many techniques aim to segment the images into objects by semantic meaning and color each segment according to training images of similar semantic meaning [7]. Big data techniques try to average the color data from closest images [8]. There have also been numerous color transfer proposals such as Reinhard *et al.* [5] to transform the three-dimensional color distribution of a picture to match an input image.

Nearly all of these techniques work well in their own regard – there is no immediate need for a new system. However, there are some points to take into consideration when comparing methods. For the "training" set, in terms of color palette, it much more desirable to have images that are similar semantically, not simply in shape. A picture of an Egyptian obelisk might be closest in shape to one of a pencil, but the color palette would not match. Instead we would want the colors of other Egyptian monuments. To leverage the entire color palette, we plan to use the image analogy method proposed by Hertzman *et al.* [1].

While image analogies have been used in various other applications, to our knowledge is has not been used as the heart of any previous major colorization systems. Perhaps this is due to the fact the algorithm can be quite slow and does not perform segmentation like traditional colorizers would. However, we believe it is worth exploring image analogies for automatic colorization due to promising initial results of a case-by-case basis.

3. Image Colorization

Here we describe the algorithm and elements necessary to support our image colorizer.

3.1. The algorithm

First, in the classification phase, the a classifier is trained on an image dataset, if not already, and used to label the input image. The classifier used in our framework utilizes the bag-of-words technique to learn the image categories by extracting SURF features of all categories and using k-means to reduce the number of features.

Second, in the exemplar search phase, a large subset of the category is searched to find structurally similar images. The Structural Similarity (SSIM) index was used to measure similarity between two images. This method was used since it is purported to be more consistent with human eye perception than methods such as PSNR and MSE. Since image analogies can be sensitive to the relative position of objects in the scene, SSIM index is a useful metric. The top n ranked images are chosen to be used as exemplars. Mul-

tiple exemplars were used because image analogies often only colors *certain* sections of images well, based on the exemplar provided. Thus you can introduce more essential colors to the image by using multiple training images. With more than one exemplar used, and the results combined, it is more likely to have a valid colorization with an aesthetically pleasing color palette.

Third is the image analogies phase. Each exemplar image is converted to grayscale. The grayscale image is then used as the A image and the colored exemplar is used as the A' image. The input image is used as B. This paper will not go into the image analogies algorithm in detail. You can do so on the NYU Media Research Lab website: `http://www.mrl.nyu.edu/projects/image-analogies/`.

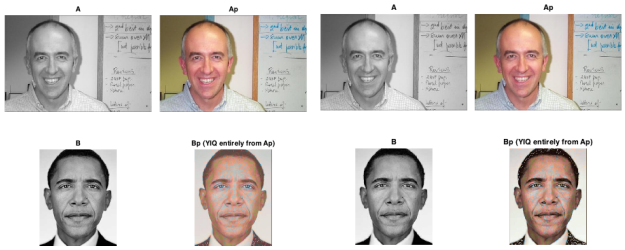
In short, image analogies is provided images A, A', and B and is tasked with synthesizing image B'. At each pixel synthesized in B', the function finds the best matching pixel in A/A' based on the surrounding neighborhood of features (in this case, luminance). In terms of the YIQ color space, the synthesized pixel uses the IQ values from the matched pixel in A' and the Y value from the corresponding pixel in B.[1] In order to account for the various size of visual features, this process is carried out at l different resolutions, or "levels", of A, A', and B using a Gaussian pyramid approach (see Figure 3).

Luminance remapping is a suggestion by the original paper to remap the luminance if A and A' to match the mean and standard deviation of luminance in B. The remap reportedly improves feature vector matching during pixel synthesis. Depending on the lighting or contrast difference between A and B, one of the two results from luminance remapping and non-luminance remapping is usually blown-out while the other looks valid (see Figure 2). An average of both of these two results typically gives in an acceptable coloring for all luminance variations. Thus, image analogies is carried out twice for each exemplar, once with luminance remapping and once without.

The size of the neighborhood for pixel scanning was 5x5, and 3x3 for lower resolutions, as suggested by the paper. This was tested against different neighborhood sizes and 5x5 indeed appeared to give the best balance between colorization quality and runtime. It was able to understand larger features in the image, especially with more levels.

Fourth, in the colorization combination phase, all colorizations from the previous phase are combined using a special "vibrance enhancer". Instead of a straight averaging of colors, the COMBINECOLORIZATIONS function weights the color of each picture based on its vibrance. Thus colorizations that perform poorly due to an inferior exemplar do not muddy the other valid colorizations. This is also

[1]This is for the colorization application of image analogies specifically. Other applications may use different YIQ combinations.



(a) Luminance remapped (b) Non-Luminance remapped

Figure 2: Luminance remapping vs. non-remapping. In this example, non-remapping performs better. The results are averaged.

due to the fact that some exemplar images will simply not have the palette necessary to color certain sections of the image. For instance, an exemplar may not be able to color the sky blue because it has no blue colors itself; the result is usually a washed-out alternative color. With the vibrance combination method, the blue sky from a valid colorization would be weighted more heavily than the washed-out sky. The final image is then smoothed, strictly to fix artifacts (i.e. specks) that sometimes result from erroneous results of image analogies.

The full algorithm for coloring an image B can be described in pseudocode as follows:

function COLORIZE IMAGE(B, n, l)
  $label \leftarrow$GETIMAGECLASSIFICATION(A)
  $e_1, ..., e_n \leftarrow$RANKNEARESTNEIGHBORS($label, B, n$)
  for $i \leftarrow 1...n$ **do**
    $gray_e_i \leftarrow$TOGRAYSCALE(e_i)
    $lum \leftarrow true$
    $Bp_1 \leftarrow$IMAGEANALOGIES($gray\_e_i, e_i, B, l, lum$)
    $lum \leftarrow false$
    $Bp_2 \leftarrow$IMAGEANALOGIES($gray\_e_i, e_i, B, l, lum$)
    $Color_i \leftarrow$AVERAGEIMAGES(Bp_1, Bp_2)
  end for
  $finalColor \leftarrow$COMBINECOLORIZATIONS($Color_1$, ..., $Color_n$)
  return $finalColor$
end function

3.2. Implementation

Except for the Matlab wrapper [4] of the `C++` ANN library [2][3], which uses *kd*-trees to find approximate nearest neighbors, the Matlab SSIM library, and the Matlab category classifier, all code for this project was written in Matlab entirely from scratch.

Given that you have an image dataset (e.g. Caltech 101) in the `dataset` directory where this file is executed, COL-

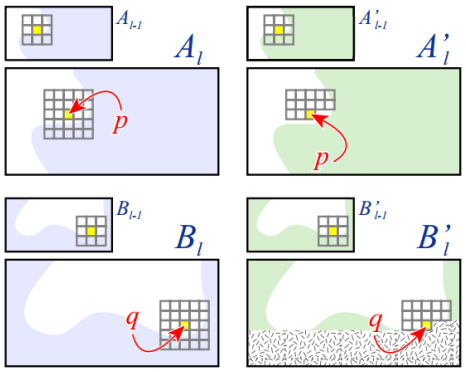


Figure 3: An illustration of the image analogy algorithm, from Hertzmann *et al.*'s paper[1].

ORIZEIMAGE can run out of the box with the following commands:

```
ann_class_compile;
colorizeImage(imagePath,train,testMode,
numExemplars,levels);
```

where `imagePath` is a filepath to the grayscale image, `train` is a boolean where `true` will train an entirely new classifier on the dataset, `testMode` is a boolean where `true` means the input image is already in color, `numExemplars` is the number of exemplars to use, and `levels` is the number of levels to use during multi-resolution synthesis in image analogies.

The following would be an example command to colorize `'GrayCity.jpg'` using the existing classifier, 2 exemplars, and 3 levels in image analogies:

```
colorizeImage('images/GrayCity.jpg', 0,
0, 2, 3);
```

The completed project source code is hosted on GitHub.

4. Results

For results of the colorization tests, see Figures 8 and 9 in the Appendix.

5. Evaluation

As a whole, our framework colorized images moderately well. We tested our framework on the Caltech 101 image dataset. The most common problem appeared to be a lack of respecting boundaries when colorizing. Since we are essentially mapping luminance to colors, this is to be expected. Colorizers that explicitly use segmentation perform better in this regard. However, the odd color bleeding that such colorizers are susceptible to is not present in our results.

As an objective metric for colorization quality, we used our own method for measuring color difference. We tested on images whose color was already known and compared the end colorization with the original. We took the difference in the IQ of the YIQ color space, divided this by the maximum possible difference, and scaled the result by some multiplier. The measurements generally indicated the following (*with multiplier = 5*):

- **100%** image colors identical
- **95%** slightly off-color
- **80%** some sections miscolored
- **50%** significant deviation in color
- **25% or lower** catastrophic failure (e.g. *inverse color*)

Unfortunately this metric assumes that the original color of the image is the only "correct" coloring, which is an imperfect measurement. Therefore we asked 10 participants in a poll to rate the colorizations in terms of whether each was valid and/or aesthetically pleasing (Figure 5). Most colorizations appeared to be "valid", but participants seemed to hold a much higher bar for "aesthetically pleasing". Usually the threshold for a "valid" coloring corresponded with around 70% accuracy, and the threshold for an aesthetically pleasing coloring was around 90% accuracy by our metric.

We found that the end colorization was heavily impacted by the failure of any one of the phases. In the first phase, an incorrect classification would lead to an unrepresentative exemplar with often undesirable color palettes. As the number of categories increased, misclassifications increased and colorization quality declined (Figure 4). In the second phase, inferior exemplars (*e.g. 2-D figures, drawings, poor camera angles, etc.*) were sometimes chosen over more photo-realistic ones, resulting in unrealistic colorizations (see Figure 9a). In the third phase, image analogies would often fail to due major differences in lighting and luminance values. In the fourth phase, perhaps one of three colorizations would fail and result in extremely vibrant, but *wrong*, colors; unfortunately, the vibrance combination would favor such a colorization (see Figure 9d).

Another major disadvantage to this approach is the runtime. While increasing the number of levels improved results (Figure 6), it heavily impacted runtime. Figure 7

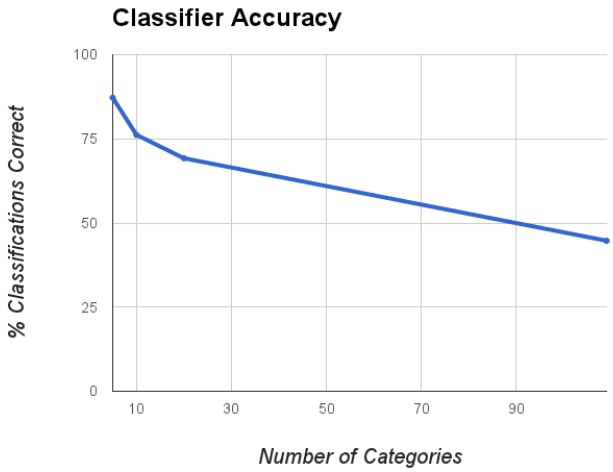


Figure 4: The effect of number of categories on classifier accuracy. As the number of categories increases, the number of misclassifications increase. An "all-purpose" classifier would likely perform poorly.

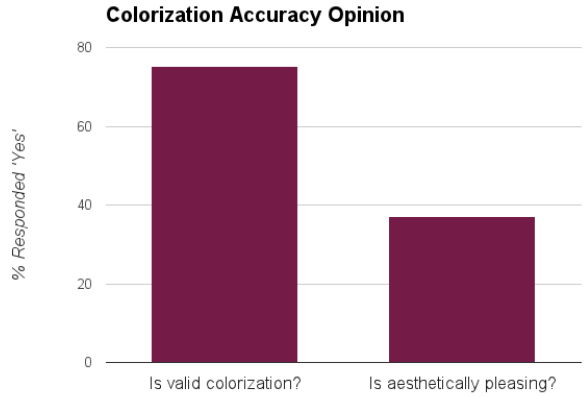


Figure 5: Polled responses on colorization results. A majority of participants voted that the results were a valid coloring; however most did not find it aesthetically pleasing.

shows the average runtime for the best results with levels = 5 was over 8 minutes, which is significantly longer than many other colorizers.

6. Discussion

Here we discuss the successes and failures of the framework as well as possible future development.

6.1. Analysis

Aside from misclassifications, the image analogies phase is where the most errors are introduced. The analogies are very sensitive to image structure, background clutter, lighting, contrast, and color vibrance. Most of the time the analogy fails to capture the 'boundaries' of certain features and ends up coloring most of the image in the same manner. This results in a sepia-tone or a tinted image (see Figures 9e and 9f). This is boring largely unacceptable.

Due to that fact, our colorizer had a difficult time with faces in particular. We found that many photographs of 'faces' often highlighted the face and had huge contrasts with other elements of the image; the background was either blown-out or the same luminance as the face. As a result our colorizer would use the pink skin-tone to color any elements of the input image that weren't very dark or very bright (see Figure 9g).

Since image analogies works in neighborhoods in scanline order, "bands" of color are often influenced horizontally across the image. Thus, gradients fail to colorize well (see Figure 9d).

Picking the exemplars was the second most common place where things went awry. For example, the Caltech 101 dataset included some images that you certainly would not want to use as a model for color, such as in Figure 9a .

6.2. Future Development

Better image analogy features. Image analogies is not inherently suited for colorization across all varieties of images. The fact that we are working in grayscale severely limits the improvements you can make on the algorithm. The original paper suggests that you can get a boost in quality by changing the amount and types of features you use, such as RGB channels; indeed this improves performance of other image analogy applications, but obviously this is not possible for colorization. In the future we may look into features other than luminance that may be available in grayscale color space.

Better image dataset. It would be interesting to see how this framework would perform with a stronger classifier and more rich dataset. Ideally, you would want a dataset of images with a full palette of vibrant colors enclosed in clearly defined sections. Each should be representative of their category.

Automatic alignment. Sometimes photos were rotated or taken at an odd angle. This would throw off the classifier and usually resulted in a misclassification. Automatic alignment of input images would improve this area.

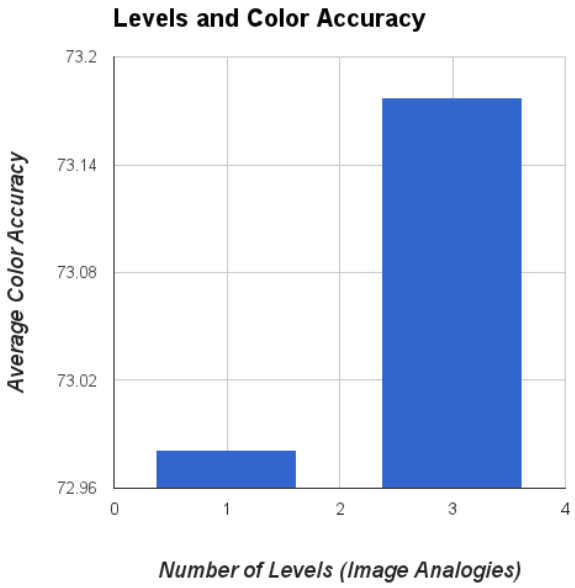


Figure 6: Number of levels of resolution used in the Gaussian pyramid for image analogies and the colorization accuracy. As the number of levels was increased, the average accuracy increased.

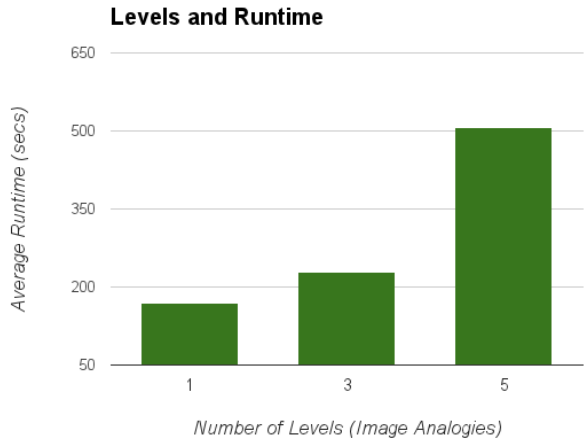


Figure 7: Number of levels of resolution and the runtime of the entire colorization algorithm. The runtime is heavily dependent on the number of levels used. As the number of levels was increased, the average runtime increased.

Optimize speed. The performance of the algorithm is heavily dependent on the image analogies stage, which is logarithmic to the number of pixels in the training pair and linear to the size of the input image. This represents a major bottleneck in the process. While the ANN library certainly gives a speed boost for feature vector matching, other search methods and heuristics may offer a more acceptable trade-off.

Integrating segmentation. The analogies most commonly fail because the colors do not understand the overall features in an image. It would be interesting to see if you could introduce segmentation. You might segment similar areas from each image and then perform a sort of image analogies within each segment. Certainly this would produce valid colorizations more often.

Intelligent colorization combination. Combining two colorizations can produce muddy results, even when using the vibrance combination method. Often two aesthetically pleasing colorizations are combined to produce a final colorization that is worse than either of the two inputs by themselves. It would be better to segment the colorizations and evaluate the validity of each section, take only the colors of the most valid section on a case-by-case basis.

References

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

[2] Sunil Arya, David M. Mount. *Approximate Nearest Neighbor Queries in Fixed Dimensions*. In *Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 271-280, 1993.

[3] David M. Mount, Sunil Arya. *ANN: A Library for Approximate Nearest Neighbor Searching*. @ONLINE. August 2006. URL: `http://www.cs.umd.edu/~mount/ANN/`

[4] Shai Bagon. *Matlab class for ANN*. @ONLINE. Feb 2009. URL: `http://www.wisdom.weizmann.ac.il/~bagon/matlab.html`

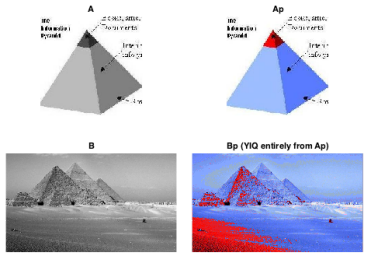
[5] Reinhard, E. Ashikhmin, M., Gooch B. and Shirley, P., 2001. *Color Transfer between Images*, IEEE Computer Graphics and Applications, September/October 2001, 34-40.

[6] Welsh, T., Ashikhmin, M., and Mueller, K. 2002. *Transferring color to greyscale images*. ACM Trans. Graph. 21, 277280.

[7] Chia, A. Y.-S., Zhuo, S., Gupta, R. K., Tai, Y.-W., Cho, S.-Y., Tan, P., and Lin, S. 2011. *Semantic colorization with internet images*. ACM Transactions on Graphics 30, 6, 156.

[8] A. Torralba, R. Fergus, W. T. Freeman. IEEE. *Transactions on Pattern Analysis and Machine Intelligence*, vol.30 (11), pp. 1958-1970, 2008.

Figure 8: Successful colorizations using our framework.

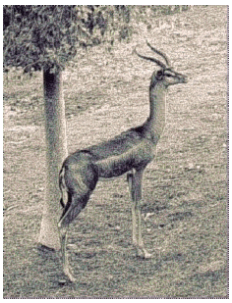


(a) Picking the right exemplar is important.

(b) Incorrect section colors

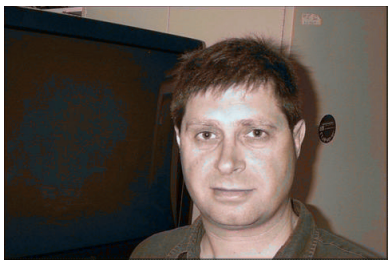
(c) Incorrect section colors

(d) ”Bands” across gradients



(e) Sepia-toned result

(f) Sepia-toned result



(g) Poor face results due to lighting

Figure 9: Failed colorizations from our framework.