Colorizing Black and White Film Photography:

Comparing the Prokudin-Gorskii Method and Neural Networks

Isabel MacGinnitie

CS144/MATH164 Scientific Computing

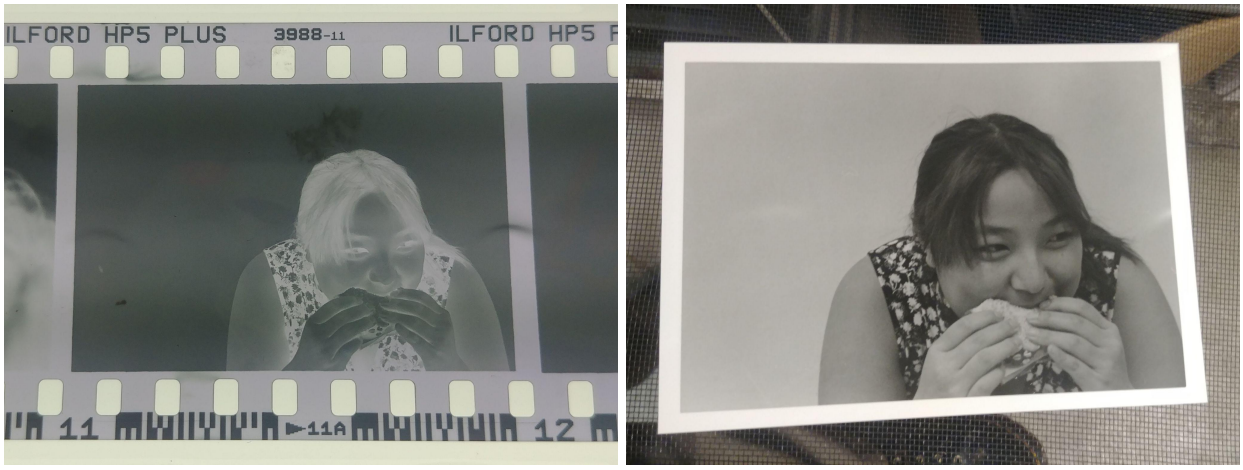
As a joint project with MS160 Computational Photography II

May 7, 2021

Table of Contents:

1. Introduction

In the age of social media and amazing digital cameras on even budget smartphones, it's no surprise analog film is no longer the layperson's photography medium of choice. However, there exists significant communities of amateur and hobbyist film enthusiasts, with Reddit's r/analog, a subreddit dedicated to film photography, having 1.3 million members, and Instagram's #filmphotography tag having 28.6 million posts. The vast majority of the images found in these communities are color film, but black and white film is still extremely popular. It often serves as a first exposure to the analog process due to its relatively simple chemistry, including at 5C introductory film photography courses. This allows significantly easier development and printing, which are the processes of turning film exposed to light into usable negatives and creating physical photographic prints, respectively. Therefore, there are realistic circumstances for an aspiring film photographer to be restricted to black and white film.



Left: a developed black and white film negative. *Right:* the corresponding silver halide print.

Consider two cases where we are restricted to the black and white process, but want a colored result: first, that we only have black and white film but want to take color photographs, and second, that we have some black and white negatives already that we want to make into color photographs.

In this project, I will present an approach for each of these cases: first, I will present a modernized, computational adaption of the Prokudin-Gorskii, or three-color, method—a technique to *create* color photos with black and white film; second, I will detail a way to colorize *pre-existing* black and white photos by training a neural network.

2. The Prokudin-Gorskii or Three-Color Method

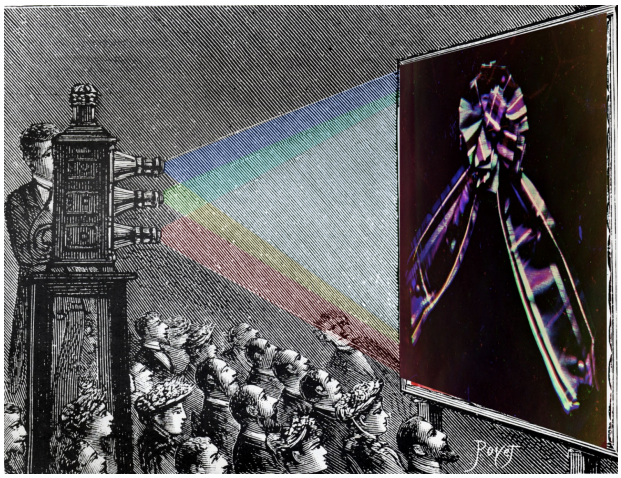
2.1. Background

Sergei Prokudin-Gorskii (1863-1944) was a Russian chemist and photographer who developed a novel technique to take color photographs with a black and white process before the invention of conventional color film (“Russia”). In the early 20th century, he spent a decade traveling through the Russian Empire and photographing its peoples and environments, shooting 1,900 negatives on glass plates (“Russia”).

In this technique, the photographer takes three exposures of the same scene with a translucent red, green, or blue filter over the camera lens, one for each color (“Color Photography Method”). The filter absorbs the light of the opposite wavelength, so light with the complimentary color will not reach the film, and thus not cause it to darken. For example, on the red-filtered negative, the red parts of the image would be darker on the negative than the green parts.

Negative and positives. *Left:* red-filtered. *Center:* green-filtered. *Right:* blue-filtered.

In the more classical technique, the photographer now prints an inverse of these three negatives, called positives, onto transparent glass. With a special triple-bulb “magic lantern” projector, they project red, green, and blue light into the corresponding positive, such that the three resulting images are aligned on a white background. This creates a singular color image because of the additive nature of light.



An illustration of the magic lantern technique (from https://www.google.com/url?q=https://www.skipcr.cz/akce-a-projekty/dokumenty/akm-2011/Hubicka.pdf&sa=D&source=editors&ust=1620427056868000&usg=AOvVaw0qyK0O_KhWDOkcOwbMfJaB).

In 1948, the Library of Congress purchased this Prokudin-Gorskii collection, and in the 1990s, archivists made significant strides in digitizing the negatives and combining the scans into a color image. However, the negatives are not perfectly identical, so manually aligning them is very time consuming. Thus, in the early 2000s, computer programmers created a computational, automatic process to align and combine the three negatives in a digital color image ("Russia"). Now, the Library of Congress has made high-quality black and white scans and their combined color images available on their website for public use.

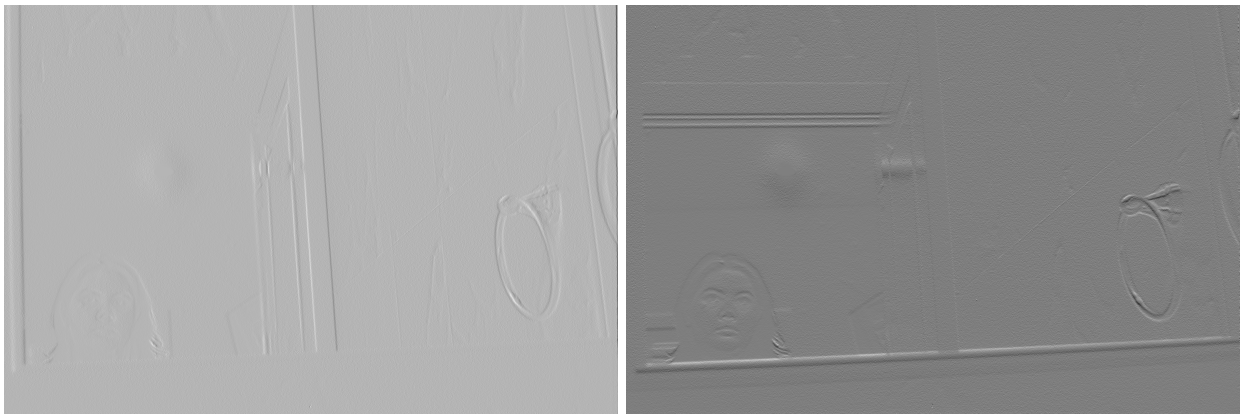
2.2. My Implementation

In the first part of this project, I both followed the Prokudin-Gorskii method to take sequences of three exposures using black and white film, then algorithmically aligned and combined the scanned images into color photographs.

To take the images, I used a Canon Rebel T2 camera with a Canon EF 38-76mm f/4.5-5.6 lens and plastic red, green, and blue filters with 36 exposure Kodak Tri-X 400 film. For each of the 9 scenes, I took one image each using the red, green, and blue filters on my film camera, for use in the alignment algorithms detailed below. I also took a color image on my phone camera for comparison and an unfiltered image on my film camera for use in the neural network described later in this report. To standardize the images, I used a tripod to help stabilize and held the zoom and aperture of the lens constant, so only the exposure

time varied between shots. I intentionally set up these 9 scenes to be similar to Prokudin-Gorskii's collection, with a focus on people, greenery, and the sky. I then sent in the film to get professionally developed and scanned. Once I received the scans, I used two algorithms developed in the Prokudin-Gorskii lab of Computational Photography II to align and combine them.

The first method was largely given in class by Prof Doug Goodwin and uses an image pyramid and a Sobel filter to make alignments. A Sobel filter identifies the areas of an image with the most difference between pixels, which we call their "edges." First, we find these edges of the image with the Sobel filter, and then zoom out 20 times (such that our image is 20 times smaller than the original). We now try to align the edges for each of the red and blue images with their green counterparts by trying all possible moves in the x and y direction ±15 pixels. We keep whichever move minimizes the distance between the red or blue edges and the green edges, and then zoom in by a factor of two. This process of zooming out and then successively zooming in is the aforementioned image pyramid technique. We repeat this process until we are back to the original image size, and stack the now-aligned three images as a 3D array.



Left: Sobel filtered positive with vertical edges. *Right:* Sobel filtered positive with horizontal edges.

The second method (which I wrote) relies on facial recognition to align the three positives by aligning the faces in the photo. It uses dlib's face predictor to get the positions of 68 features (left eye, nose, etc) on each face in the images, and then calculate the distance between each feature on the red or blue positives and the corresponding feature on the green positive. It averages all these distances to get an x-y transformation that aligns all these features, and hopefully the rest of the image. We finally do the transformation and stack the images as a 3D array.

In both methods, each layer of the resulting 3D array is the 2D array of pixel intensity for one of the three color channels, red, green, and blue. In other words, before we stacked them, each layer was a grayscale image corresponding to the reds, blues, or greens of the scene. In stacking them, we assign each layer to the red, blue, or green channel, so for example, each value in the array representing the red-filtered positive now tells us how strongly red the pixel in the same spot will be in the color image. Now, we can save these aligned and stacked arrays as colored JPG images.

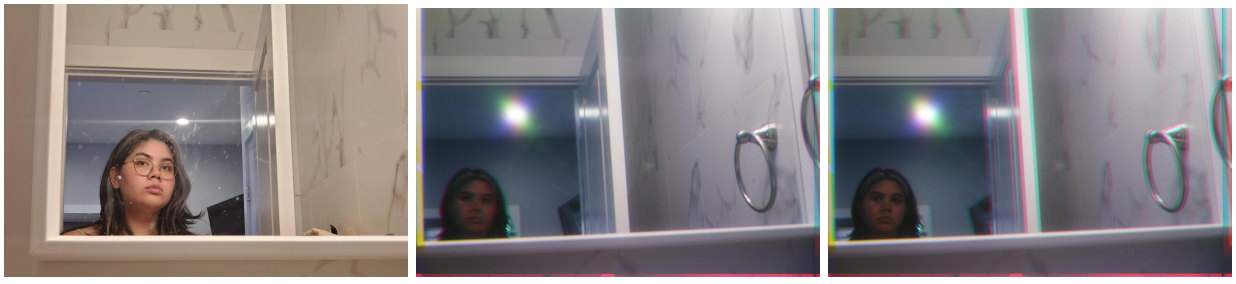
2.3. Results

Most shots had at least one alignment that looked very reasonable, most frequently the pyramid method. Although I made sure all the images had faces in them, in 4 of the 9 scenes the dlib face detector did not recognize the face in some or all the positives. Thus, the facial alignment method could not run, and its output remained unaligned for those images.

Where both methods were successful, which method was better is a very subjective choice. The people and faces were what generally had the most movement, so we have to decide if we value a more aligned face or background.

The aligned images are also somewhat cooler toned than the digital images. However, due to the automatic photo processing phones do, I think the digital images are slightly more warm and saturated than real-life. In the selected images below, we can see the varying levels of success for the different scenes.

Selected images. *Left to right:* phone image, pyramid alignment, face alignment.



Above: results of the example negatives.

Above: aligned images are significantly cooler than phone images.

Above: face alignment and pyramid can give significantly different results.

Above: Some faces were not detected by the face predictor, so face alignment did nothing.

2.3. Discussion

Overall, the Prokudin-Gorskii method has relatively easy to reproduce, aesthetically pleasing, and generally realistic results when trying to create color photos with black and white film.

Personally, I like the coloring of the aligned images more than the warm, saturated digital photos, particularly its combined effect with the colors in misaligned sections. The overall coloring also reminds me of the dreamy, retro look of expired or budget color film, such as Kodak Gold 200 or Fujifilm Superia X-Tra 400.

Conventional color film. *Left:* Kodak VR 1000, expired 1986, shot 2019. *Right*: Fujifilm Superia X-Tra 400.

Film and computers have come a long way since the original Prokudin-Gorskii negatives and later computational alignments, so it is now reasonable for even beginners to try this method. Although I had a nicer camera and experience with film, a less experienced photographer could likely use any film camera (such as an inexpensive point-and-shoot or disposable) as long as it can be held stable between shots and get comparable results. Similarly, a less experienced programmer could easily use pre-existing code to align their images, or even use Photoshop or similar software to do so manually. Therefore, the accessibility to film and coding beginners is a clear advantage of this method.

A notable weakness of this method is how much film it uses, and therefore how expensive it is. A roll of Kodak Tri-X 400 is currently $7.99 for a roll of 36 exposures, so even assuming access to inexpensive home film development and scanning, the net cost is about $0.67 per image. In comparison, a roll of budget color film is about $5 for 36 exposures, and the cheapest professional development is around $12 a roll, so just using color film to get color images is only $0.47 per image.

Most significantly, this method only works to *create* color images, so is not relevant in any other case, such as wanting to colorize pre-existing or historical black and white images.

3. Neural Networks

3.1. Background

The conventional process to colorize black and white photos involves time-consuming hand-coloring, either with pigments on paper or pixels in Photoshop. Before the invention and popularization of color film in the mid-20th century, hand-colored images were the only way to achieve a color photograph, and the accuracy of their coloring depended solely on the colorist's artistic vision (Weitz). Modern colorists digitally add layers of tints to black and white scans, often after significant research to determine the historically accurate colors ("The problem"). However, it can still be difficult to objectively tell what is the "right" color for historical photographs and other images without objective color references to compare to, e.g. in machine learning lingo, lacking a ground truth.

With the explosion machine learning research and applications in the last few decades, the idea of using artificial intelligence to automatically colorize black and white images has correspondingly grown in prominence. Currently, two major approaches are to use either a convolutional neural network (CNN) or generative adversarial network (GAN) to colorize the images (Wallner). In this project, I followed Emil Wallner's "How to colorize black & white photos with just 100 lines of neural network code" CNN tutorial and adapted his implementation to fit my needs—a neural network optimized for Prokudin-Gorskii style images that ran exclusively on Google Colab Pro.

This tutorial is based on the 2017 "Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2" project by Federico Baldassarre, Diego Gonzalez Morín, and Lucas Rodés-Guirao from at KTH Royal Institute of Technology. In it, Baldasarre and team propose combining a pre-trained image classification model with a convolutional neural network to predict the coloring of inputed black and white images. The pre-trained classifier identifies high-level features in each image, and the CNN predicts what color each of those features should be. As seen in the later results, the classifier and CNN excelled at identifying plants and other greenery and coloring them green. Like other neural networks, these predictions are based on the data the model encountered in training. In high-level terms, this training happens by inputting large amounts of inputs into the model, which compares the predicted colors to the actual colors and subsequently adjusts the prediction model to minimize the difference. Because of this process, models generally have better predictions for inputs when they are trained on data similar to them.

3.2. My Process

In this part of this project, I had the two-fold goal of adapting Wallner's model to work using Google Drive and Colab and choosing training datasets to optimize coloring my 9 film photos and original Prokudin-Gorskii images, which contained mainly people, greenery, and the sky.

Wallner's tutorial gives three iterations of the colorizer: an alpha version with no classifier that trains on a single image, a beta version with no classifier that runs on multiple images, and a full version incorporating a pre-trained classifier Inception ResNet v2 with the CNN. These versions were already in Jupyter Notebook form but were built to run on machine learning platform FloydHub with datasets stored in their servers.

For the alpha and beta versions, I added code to load training images from a specified Drive folder and save testing images and models to another. I also created supplementary notebooks to resize training and testing images to 256x256 images, as well as load a previously trained alpha or beta model (the latter functionality was directly added to the full version's main notebook). I then used the beta version to test some of the datasets I intended to use on the final implementation by running small batches of the set's images on selected test images.

The full version of the colorizer required significantly more adaptation. Wallner's code relies on deprecated versions of TensorFlow and Keras and is unfortunately written in a way not easily adapted to the significantly updated TensorFlow 2. However with research, I serendipitously found the Kaggle project Image Coloring Using AutoEncoders, also based on Wallner's tutorial, which specifically requires TensorFlow 1.14.0 and Keras 2.2.4. Although not a long term solution, I used these depreciated libraries to allow the model to work in Colab.

Colab also only has enough RAM to load about 4,000 256x256 images before crashing, so I also added code to allow easy loading and saving of models. Thus, training can be done sequentially on smaller batches of the total training data. Since I wanted to compile multiple datasets, I also wrote code that allows the user to run these smaller batches in random but non-repeating order. Admittedly, this addition was only after I spent two days training the model only to realize that running each dataset separately significantly biased the model to the last dataset it trained on.

I had initially intended to screen datasets on the beta version and then test them on the full version. However, for timing reasons, I only tested two of the three datasets used in the final version on the beta version before moving to the full version. I ultimately used the following three datasets: Wallner's Unsplash, Nvidia's StyleGAN, and Stanford DAGS Lab's Stanford Background Dataset.

The Unsplash dataset contains approximately 9,500 training images and 500 testing images from the website Unsplash, a source for freely-usable, professionally taken photos, in a generally artistic style of mostly portraiture. I used both the training set and the testing set to train and test my final model, respectively.

The StyleGAN dataset is a set of 100,000 GAN generated closeups of faces which were relatively diverse in race, ethnicity, age, and gender, at least in comparison to other face databases. I used 5,000 images to train and 1,000 images to test my final model.

The Stanford Background Dataset contains around 700 images of outdoor scenes with something in the foreground. I used this entire dataset to train my final model.

Left to right: Example images from StyleGAN, Unsplash, and Stanford Background Dataset training sets.

I had hoped to use the comprehensive ImageNet database, as it was also used in the training for the image classifier. Unfortunately, their website was being updated at the time, so it was impossible to choose specific subsets of the data. I was only able to access their randomized testing dataset, which contained images so random that I ultimately chose not to use it after experimenting on the full version.

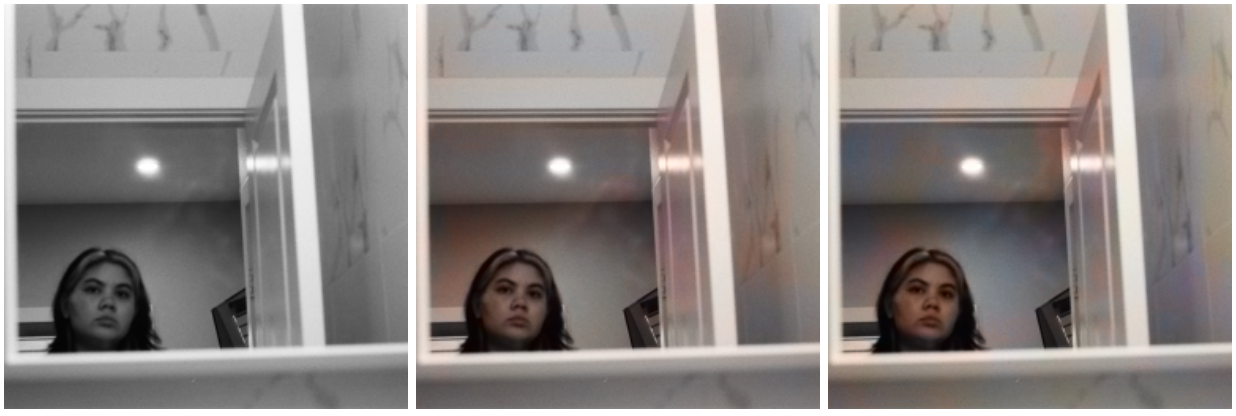
I also compiled a comprehensive set of images to test my model on. It consisted of the 9 scans of my Prokudin-Gorskii-style images, 199 scans from the original Prokudin-Gorskii dataset, the Unsplash dataset's approximately 500 testing images, the StyleGAN dataset's 1,000 faces, and 21 miscellaneous scanned black and white film images, including both cropped faces from the 9 earlier scans and images from 5C campuses.

256x256 images from the test set. *Left to right:* My film, an original Prokudin-Gorskii negative, Harvey Mudd Campus in Spring 2020, shot on Ilford HP5 Plus 400.

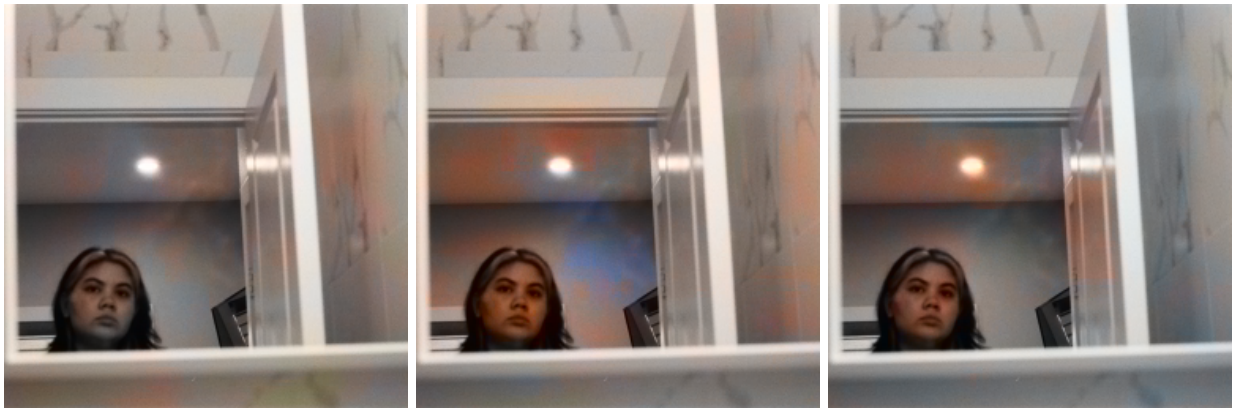
3.3. The Final Implementation

I trained my full final model on 15,007 images, divided into five sets of 3,000 (with the batch #5 being 3,007). For each set, I trained the model for 50 epochs, which each ran on batches of 30 images for 100 steps. After each set, I ran the model on the testing images and spot-checked that the colorization was improving and no problems had occurred. Each set took approximately 3 hours to train and test, so the total runtime was around 15 hours.

The visible improvements between sets.



Row 1, left to right: original image, set #1, set #2.



Row 2, left to right: set #3, set #4, set #5 (final).

3.4. Results

Overall, the neural network colored images approximately correctly, but with a very light hand. Most of the “bad” images lacked any color rather than being colored oddly,and even the very successfully colored images are lightly shaded. Most noticeably, the full model excelled at identifying and coloring plants green and the sky blue, almost universally. It also tended to color roofs and buildings red or orange but had more trouble identifying them. The model also struggled to colorize people and faces, but when it did, they came out lighter toned and orangey, but well within the realm of actual skin colors.

Selected images.

Above: trees and plants colorized green. *Left and right*: Prokudin-Gorskii negatives.

Above: skies colorized blue. *Left:* miscellaneous film. *Right*: Prokudin-Gorskii negative.

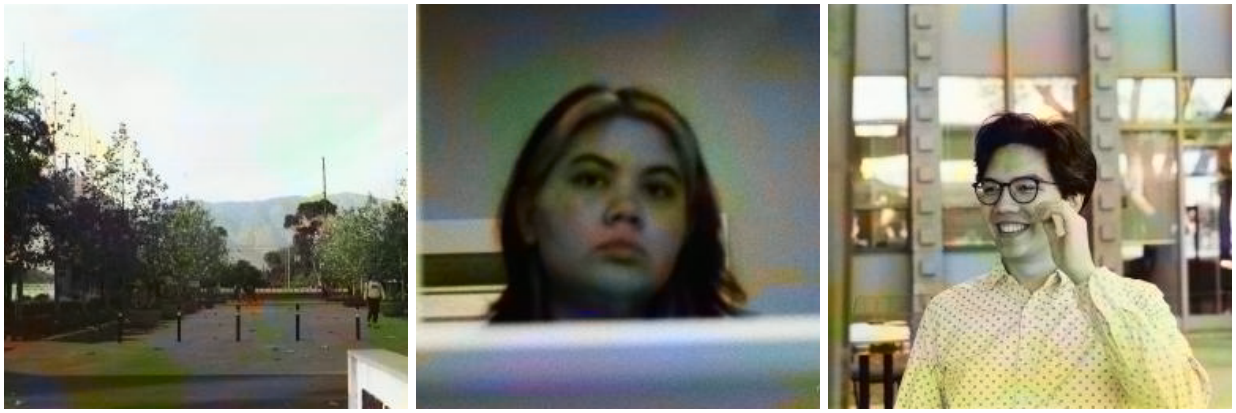
Above: face and people sometimes colorized warmly. *Left and right*: miscellaneous film.

3.5. Further experimentation

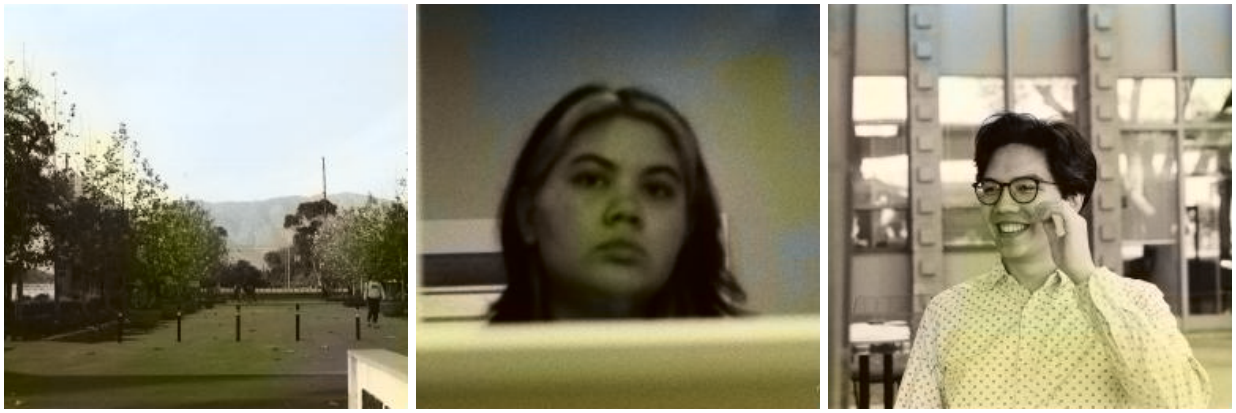
I also ran four models using only the color versions of film, the 199 professionally colorized Prokudin-Gorskii images I used for testing and the 9 digital photos corresponding to each scene in my film set. I trained both the beta and full version of the colorizer with each of the two sets, then ran original black and white negatives as the test set. The results of both the Prokudin-Gorskii trained models and the digital photo trained beta model were very

interesting and diverse. However, the digital photo trained full model either made predictions out of the color space or entirely colorless predictions, likely because of the miniscule 9 photo training set.

Selected results.

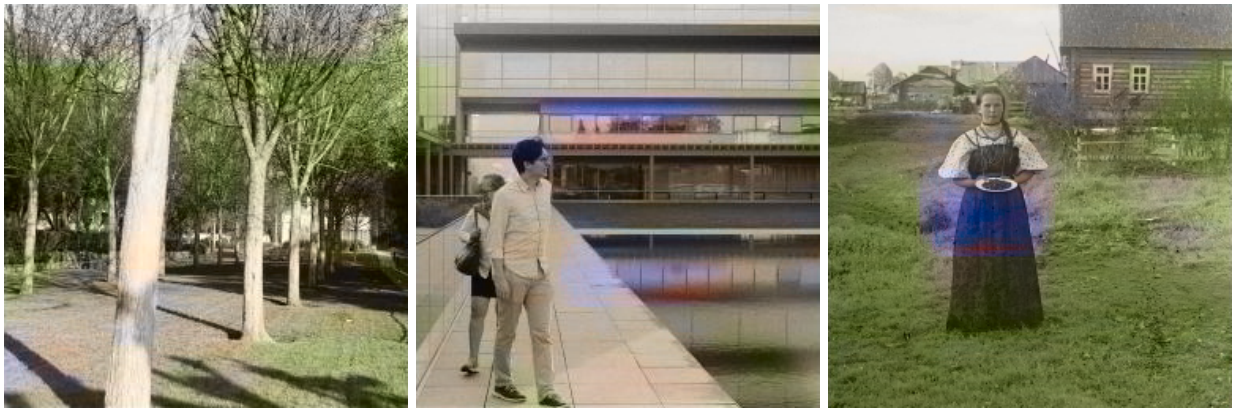


Above: Prokudin-Gorskii trained beta version, tested on miscellaneous film.

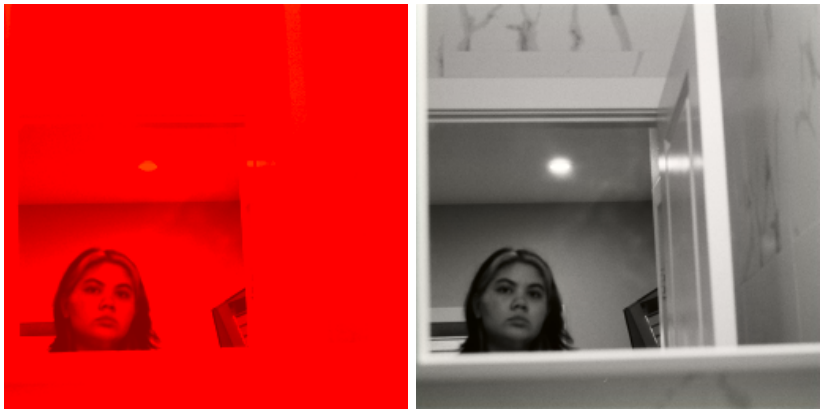


Above: Prokudin-Gorskii trained full version, tested on miscellaneous film.

Above: Digital photo trained beta version, tested on similar film images.



Above: Digital photo trained beta version, tested on unrelated film images.



Above: Digital photo trained full version, tested on similar film images.

3.6. Discussion

Overall, this model colorized black and white images with varying levels of success, though with notable strong points. However, it had significant weaknesses in how technically complicated, time consuming, and resource intensive the model is.

This model is realistically impossible to run without access to a GPU and takes around 15 hours to run, even with the GPU in Google Colab Pro. Therefore, it is not particularly accessible to those without these resources or to those less experienced with coding in general. It is also somewhat specialized to Prokudin-Gorskii-style images, and although not pictured in this report, had mixed success on the other testing images.

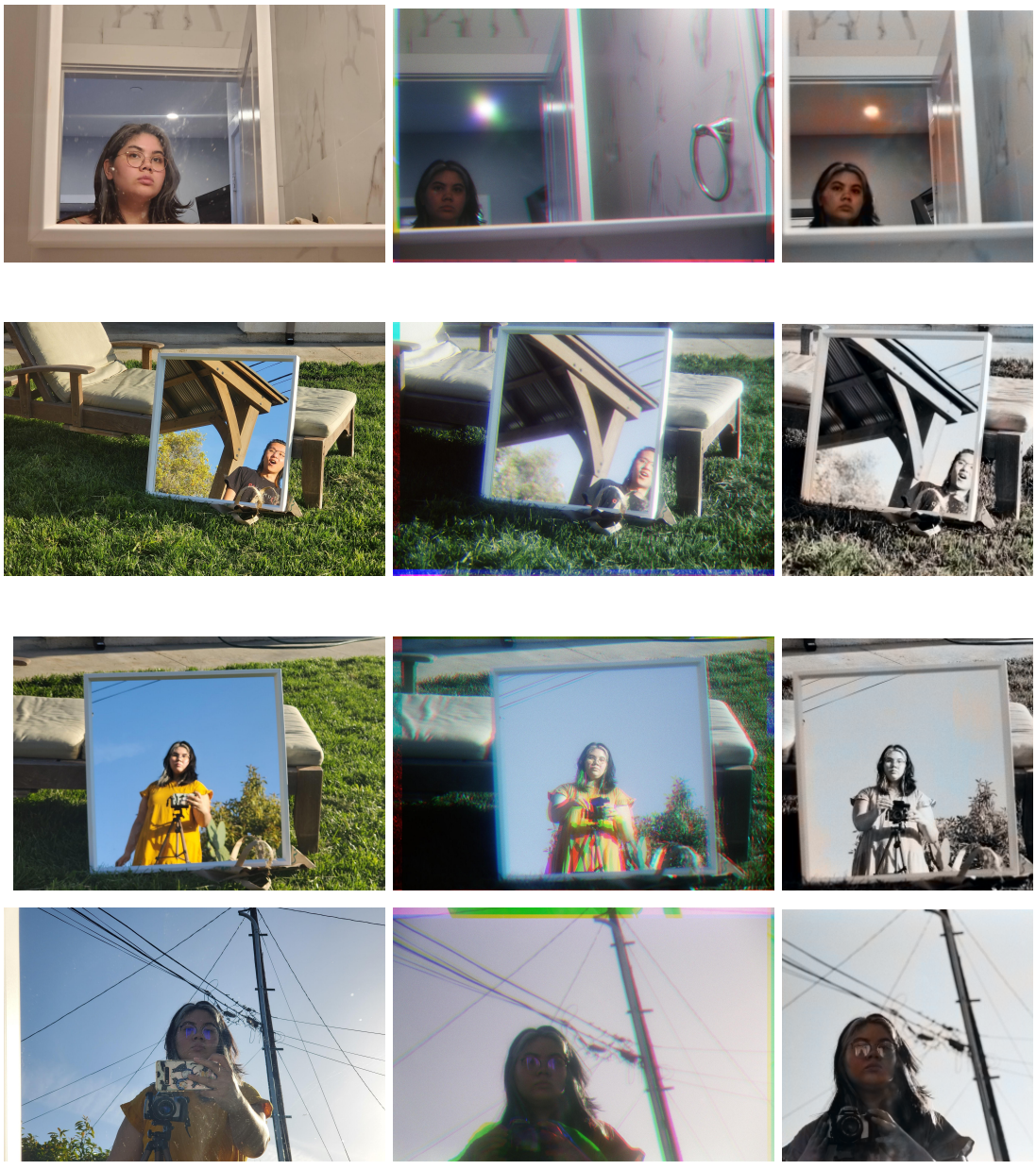
In terms of more subjective measures, I personally quite like the general look of the successfully colorized images, especially those containing trees, the sky, or people. However, because the images had to be square and low resolution to be colorized, I am not sure how usable these images would be for non-academic purposes.

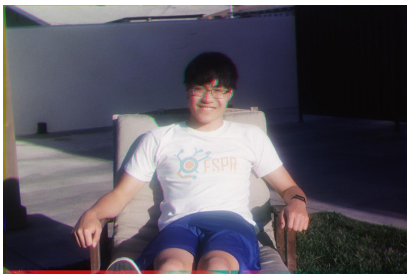
In general, this specific model was not universally successful or usable, but still begins to solve the problem of wanting to colorize pre-existing black and white film.

4. Side by Side Comparisons

4.1. My Film

Left to Right: phone image, aligned image, colorized image.














4.2. PG negatives

Left to right: professionally aligned image, black and white image, full model colorized image.









5. Conclusion

The success of this project is ultimately subjective, as how “good” or “bad” each result is depends on our own perception of the image. In general, I was very satisfied with both the Prokudin-Gorskii method to create color images from black and white film, as well as using a neural network to colorize grayscale images. The two methods are very different in both application and implementation, so realistically have little overlap. Thus, I do not find it necessary or even possible to suggest one is objectively superior to the other.

The learning experience of this project, complex in both magnitude and content, was also incredibly valuable for me as an aspiring film photographer and computer science student. Overall, I am completely satisfied with both what I learned and the actual results of the project. I would highly recommend experimenting with these methods to other budding photographers and students alike.

Sources

A. Bhole. “Image Coloring Using Autoencoders.” 13 Nov. 2019, www.kaggle.com/abhishekbhole/image-coloring-using-autoencoders.

A. Weitz. “The Basics of Hand-Coloring Black-and-White Prints.”*B&H Photo.* 2020, https://www.bhphotovideo.com/explora/photography/tips-and-solutions/the-basics-of-hand-coloring-black-and-white-prints.

ADotSapiens. “The problem with AI colorization.” *Reddit.* 15 April 2021, https://www.reddit.com/r/Colorization/comments/mqn103/the_problem_with_ai_colorization/.

“Color Photography Method.” *Library of Congress*. https://www.loc.gov/collections/prokudin-gorskii/articles-and-essays/color-photography-method/.

E. Wallner. “Coloring Black and White Images with Neural Networks.” *GitHub.* https://github.com/emilwallner/Coloring-greyscale-images.

E. Wallner. “How to colorize black & white photos with just 100 lines of neural network code.” https://emilwallner.medium.com/colorize-b-w-photos-with-a-100-line-neural-network-53d9b4449f8d.

F. Baldassare, D. Morín, L. Rodés-Guirao. “🐨 deep koalarization.” https://github.com/baldassarreFe/deep-koalarization.

“Russia in Color Photographs, 1905-1914.” *Library of Congress*. https://www.loc.gov/collections/prokudin-gorskii/articles-and-essays/russia-in-color-photographs-1905-1914/.

S. Gould, R. Fulton, D. Koller. Decomposing a Scene into Geometric and Semantically Consistent Regions. *Proceedings International Conference on Computer Vision (ICCV)*, 2009. [pdf]

T. Karras, S. Laine, T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. https://arxiv.org/abs/1812.04948.

Wikipedia contributors. "Hand-colouring of photographs." *Wikipedia, The Free Encyclopedia.* Wikipedia, The Free Encyclopedia, 7 Feb. 2021. Web. 8 May. 2021.

Acknowledgements:

I would like to thank Prof Darryl Yong for his continued support and understanding throughout this complex and stressful project and semester, as well as Prof Doug Goodwin for introducing me to this amazing intersection of photography and computer science, as well as his support. I would also like to thank Harvey Mudd Math Department for paying for my Google Colab Pro subscription and film development. Finally, I would like to thank the subjects of my Prokudin-Gorskii-style images: Andy Liu and Eugene Gao, as well as those in my miscellaneous film scans: Amy Xiao and various members of my family.

Appendix:

A. Code

Also available in the accompanying code zip file.
Film alignment- “film stuff.ipynb”, contains both algorithms to align sets of 3 Prokudin-Gorskii filtered images
Beta version- “beta_version.ipynb”, contains my adaption of Wallner’s beta model
Full version- “full_version.ipynb”, contains my adaption of Wallner’s full model

Resizing images- “resize images.ipynb”, contains code to resize images to 256x256 and 128x128 dimension images for using in the neural network
Downloading/resizing PG images- “save and resize PG images.ipynb”, contains code to download Prokudin-Gorskii images from the Library of Congress website, then resize them to 256x256 and 128x128 dimension images for using in the neural network
Loading a beta model- “loading alpha beta.ipynb”, contains code to load an existing beta model and run more test images.

B. Datasets

Unsplash- source, ready to use images
Nvidia's StyleGAN- source, ready to use images
Stanford DAGS Lab's Stanford Background Dataset- source, ready to use images
My Raw Film- source, ready to use negatives, digital images
Miscellaneous Film- source, ready to use images
Prokudin-Gorskii Images- source, ready to use images

C. Full results for major iterations

Below are links to the Google Drive folders containing the colorized test sets for each of the major iterations of my colorizer.

- Aligned Film
- Final Full Version
    - Set #1
    - Set #2
    - Set #3
    - Set #4
    - Set #5 (final)
- Non-Randomized Full Version
    - Set #1 - 0-2k unsplash
    - Set #2 - 0-4k unsplash
    - Set #3 - 0-6k unsplash
    - Set #4 - unsplash + Nvidia 000000 + Stanford Background Dataset
    - Set #5A - unsplash + Nvidia 000000 + Stanford Background Dataset + 0-4k ImageNet
    - Set #5B - unsplash + Nvidia 000000 + Stanford Background Dataset + 0-1k ImageNet
    - Set #5C unsplash + Nvidia 000000 + Stanford Background Dataset + Nvidia 002000
- Experiments:
    - Beta PG
    - Full PG
    - Beta color “film”
    - Full color “film” (turns red)
    - Full color “film” (does nothing)