A Framework for Using Custom Features to Colorize Grayscale Images

Undergraduate Honors Research Thesis

Presented in Partial Fulfillment of the Requirements for the Degree
B.S. Computer Science and Engineering with Honors Research
Distinction in the College of Engineering at The Ohio State University

By

Daniel J. Marchese

Undergraduate Program in Computer Science and Engineering

The Ohio State University

2016

Committee:

Dr. James W. Davis, Advisor

Dr. Raphael Wenger

Abstract

We propose a new framework for automatic colorization of grayscale images by using the composition of features from multiple color images to solve for the most likely coloring with machine learning. The intention of this process is to alleviate the amount of manual labor required to colorize a photograph. Given a target grayscale image and several context color images, our methodology will colorize the target image using the data from the color context images. The algorithm calculates custom user-defined features for each of the context images, and then uses these features to generate a local loss function for the colors at each pixel of the grayscale image. These loss functions are then used to generate a Markov Random Field, which is solved for the most likely coloring using graph cuts. Simple features have been tested such as the luminance or variance of 5x5 pixel neighborhoods, and more complicated features have also been tested involving a combination of luminance values and texture statistics. In general, the we have found that features carrying more texture-based information tend to result in better final colorizations. More complex and descriptive features could be tested, but at the cost of algorithm performance.

To my family, who always pushes me to be my absolute best

Acknowledgments

I started this whole journey off being scared of the very notion of Artificial Intelligence. I can state with certainty that one of the largest reasons I've come to embrace the field of Computer Vision is because of Dr. Davis's innate ability to describe not only *how* to approach a problem, but *why* we choose to approach the problems in the way we do.

I would additionally like to thank Dr. Wenger for his continued help through the process of undergraduate honors research, as well as for his willingness to sit on my defense committee.

I would also like to thank Ryan Niemocienski for reminding me on a regular basis that everything is always easier than I thought. You served quite often as my "fresh pair of eyes" after I would spend weeks trying to solve a problem; I don't know what I would have done without this.

This would not be a proper acknowledgements page if it did not recognize the integral role that my parents have played in my education. For my father, who often (unknowingly) served as my rubber duck in debugging, and my mother who always finds a way to extract a life lesson out of everything, I am deeply grateful.

I would finally like to acknowledge how ridiculously out of balance my last 3 years in college would have been without the friendship of David Siegal. He taught me to channel my insane tendencies instead of ignore them.

Vita

May 2011	William Mason High School
September 2011 - present	The Ohio State University
Summer 2013	Software Engineering Intern, Thermopylae Sciences and Technology
Summer 2014 and 2015	Software Engineering Intern, Apple Inc.

Fields of Study

Major Field: Computer Science and Engineering

Studies in Computer Vision and Machine Learning: Dr. James Davis

Table of Contents

List of Figures

Chapter 1: Introduction

Automatic image colorization is the act of hypothesizing the color data for a black-and-white image without the need for user input or intervention. Clearly, there is not enough data within a single grayscale image to accurately recover the color data at each pixel. Due to the need for prior knowledge, and the lack of sufficient implicit data within the grayscale image itself, this problem is ill-posed and thus becomes a procedure of manufacturing accurate context from some reference as opposed to extracting it from the image itself (Figure 1.1). Put simply, image colorization can be formulated as a learning problem.

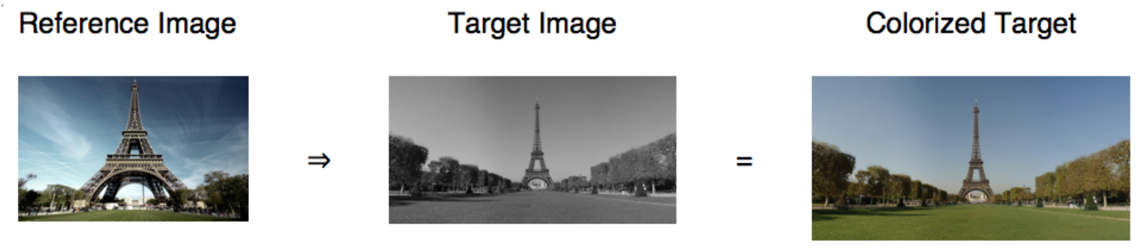


Figure 1.1: Image colorization via color transfer

There is currently much interest in the colorization of old photographs that were originally taken in black-and-white. From the perspective of professionals, there are entire companies that are dedicated to the re-touching and colorization of old photographs and movies. The process by which this is done, especially in the case of movies, requires large amounts of skilled manual labor. From the perspective of hobbyists, entire internet communities have formed around users requesting for skilled individuals to colorize their old photographs (generally of distant relatives). The hobbyists who perform these colorizations generally bring their own time and expertise, often from a professional setting, to these communities.

The goal of colorization is often to help preserve history, and even bring certain aspects of it back to life.

Currently, the best known approach for colorizing a black-and-white photograph is a tedious manual process performed in Adobe PhotoShop. It consists of manually segmenting the image in question, and layering colors on top of the image segments so that the pixel intensity values remain the same with the added color. Often, a proper colorization can take hours to complete for a single photograph, with the process time increasing linearly for the number of frames in the colorization of a movie. Even though the context that is recovered in a manual colorization often comes from a personal and human experience, there are still several things that can be adapted in an automation of the manual procedure.

When a manual colorization is performed, the person performing the colorization is drawing from their own personal experience to extrapolate the various colors in an image. For example, if someone is colorizing a scene containing a playground in a park, they are most likely recalling the colors they associated with the last park scene they witnessed. They will remember that the grass was green, the swingset was a gray-shaded metal, the swings were painted a bright color, *etc.* Overall, the colorizer is picking the most likely coloring of the various parts of the playground based on their memory of similar scenes. From the standpoint of automation, this is analogous to using a color photograph of a similar scene to the grayscale image as context (analogous to human memory) for the colorization.

Most colorization algorithms take this approach of using a color image as context. There is generally a procedure for calculating the likelihood of colors at each pixel in the grayscale image, followed by some methodology for ultimately selecting each pixel's color. The proposed algorithm in this paper also takes this approach.

The other analogous automation methodology comes from the preservation of intensity values. As stated earlier, the recovery of color in a grayscale image necessitates the calculation of three different values for each pixel: Red, Green, and Blue. The inherent problem with the RGB color definition is that one cannot change the color of a pixel independently of its brightness. In the manual process, this separation of channels is preserved by overlaying values from the HSV H, and S channels on top of the V values which are pulled from the original grayscale image. Most of the current colorization algorithms take a similar approach by choosing to perform the colorization in a color space which separates the intensity and color values so that they may be changed independently of one another. The proposed algorithm in this paper also takes this approach.

The issue with current colorization algorithms stems from the idea of using multiple contexts to colorize a single image. When a person manually colorizes an image, they are likely not considering a single scene which they only remember from one perspective. They are drawing from several different contexts. This is especially the case when colorizing a scene containing a unique combination of objects. Using the analogy of a playground in a park, let's assume that we additionally have children playing in the park who are each wearing a unique article of clothing. We can no longer rely on a single image of the playground scene for context.

The techniques used in our research are built-in part to address this issue of scene composition. Instead of treating each item in the scene as a unique object to colorize, the problem is formulated in such a way that the automation techniques address the contexts composing a single scene from the very beginning. It is the aim of this thesis to build and analyze several different techniques for performing image colorization using a customizable framework, as well as objectively measure their feasability.

1.1 Terminology

Throughout this thesis, we will refer to the images being used, and other components of the framework, in several different ways. The following are our naming conventions.

Target Image The grayscale image that is being colorized.

Source Image A color image being used as context for a colorization.

Reference Image Also used to describe a color image being used as context.

Context A collective reference to all of the source images.

2-vector The combination feature vector of luminance and variance statistics. (So-called because it is two-dimensional).

1.2 Contributions and Significance

This thesis introduces several novel approaches to the various aspects of automatic image colorization. Specifically, we present:

- A framework that is designed to be highly customizable (and pluggable in some cases). Users of our framework are not constrained to the design decisions used in the original implementation. Each of the four major steps of the algorithm presents an opportunity for customization so that factors outside of those presented in this thesis may be implemented and tested.

- A novel approach to the generation of a color palette to be used in a colorization. This approach is designed to leverage the Euclidean distance property of the $L\alpha\beta$ color space in order to preserve color variance while still capturing the overall color mood of a scene. Although our method is built to be used in further automated parts of the algorithm, the discretization algorithm on its own has a potential manual use for providing a color palette to a PhotoShop user.

- A novel approach to the handling of a scene compositionin in a colorization. This approach fundamentally increases the space of grayscale images that can be reasonably colorized due to the removal of the need for a fully composed color-equivalent context. Instead of looking *across* the space of source images, our approach looks *through* the space in order to avoid diluting the meaning of a feature match between the source and target images.

1.3 Organization

This thesis is organized as follows. We begin with a review of existing literature and an evaluation of current algorithmic colorization techniques in Chapter 2. Next, Chapter 3 provides a detailed view of the implementation of the framework, as well as the thinking behind many of the design decisions. Chapter 4 provides an overview and analysis of the results. The final chapter provides a wrap-up and summary of our framework, a listing of possible future work, and discussion of the significance and contributions of this research.

Chapter 2: Previous Works

There are already several existing methodologies for the colorization of grayscale images. This section will provide an overview of these algorithms and methodologies. It will also take a closer look at the methodologies that were used to inspire the algorithm being proposed in this thesis.

2.1 Review of Methodologies

In order to move past the need for user input to provide context to a colorization, several semi-automatic methods have been proposed to learn the necessary context. These methods involve drawing the context from a similar image (or collection of images), and then "transferring" their color data using the learned context. In general, existing colorization algorithms fall into one of two categories: direct feature matching from reference image to target image, or color probability (or cost) estimation followed by a global coherency algorithm to perform the final color selection.

2.1.1 Direct Feature-Matching Algorithms

The first set of algorithms reviewed are built using a feature-matching approach. In a colorization by feature matching, the target image is colorized by transferring colors directly from the source. The color being transferred from the source is often selected by using some type of matching statistic that is representative of specific pixel regions in the image.

The first, and simplest feature-matching technique is given in [1]. A simple matching technique is presented which transfers the color from the single best match in the source image. To save on computational costs, the algorithm only considers a subset of the pixels in the source image along a jittered grid. The quality of a match is built on the weighted distance with a normalized luminance value, as well as the standard deviation of luminance values in the 5x5 pixel neighborhoods.

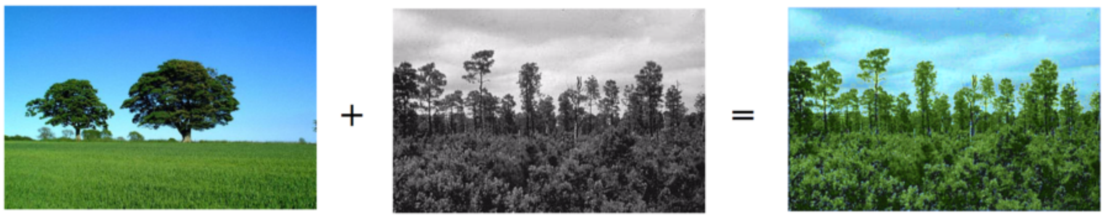


Figure 2.1: Welsh et al colorization algorithm

The next technique, proposed in [2], follows as a more in-depth approach to colorization built along the same lines as [1]. Instead of considering statistics for 5x5 pixel neighborhoods, this methodology instead employs SLIC superpixels as the grouping methodology. The statistics are then computed for each pixel in a superpixel and averaged. Features used to build the feature vector include pixel intensities, intensity standard deviation, Gabor features, and SURF descriptors.

Feature-matching algorithms are arguably the simplest type of colorization algorithm, as they only place importance on the local coherency of a color decision. Even with this simplicity, however, these methods can be rather effective given a good

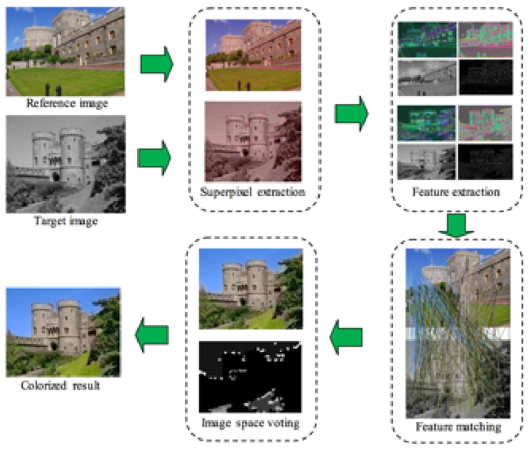


Figure 2.2: Gupta et al colorization algorithm

choice of source image. Other methods have been proposed which aim to address the lack of global-coherency gained through simple feature-matching. (In fact, [2] employed a "voting" technique to implement global coherency at the end of their algorithm).

2.1.2 Global Coherence Algorithms

Global coherence algorithms can be viewed as an extension of feature-matching. The difference lies in how the algorithms handle the uncertainty associated with the local confidence, and the methodologies are often built to take advantage of the range in costs associated with the color choices at a given pixel.

The global coherence technique with the most overlap in the feature-matching domain is the algorithm proposed in [3]. Instead of building features from the source image itself, [3] builds a source image "epitome", which as the name implies, is designed to epitomize the important information from the context. After calculating the *epitome*, the algorithm then calculates a dissimilarity between patches in the target image and patches in the *epitome*; this dissimilarity measure is then used as the local cost function in a Markov Random Field (MRF) with a smoothness function based on the selected color labels and relative pixel locations in the images.

One common technique for global coherency is the use of "scribbles". Scribbles are an optimization technique introduced by [4] in which a user provides color *scribbles* on top of the grayscale image which are then expanded out to the rest of the image to complete the colorization. In order to automate this technique, algorithms have been proposed which generate "micro-scribbles" that are then used as the scribbles in [4].

One such micro-scribble technique is that which is proposed in [5]. Their algorithm uses three different matching techniques (simple nearest-neighbor, image-space voting, and feature-space voting) to generate the final color confidence values at each pixel. These final confidence values are thresholded, and any colors with a confidence over the given threshold are used as the micro-scribbles in the final colorization. [6] approaches the problem similarly, except for their use of multiple source images as context.

The final global-coherence algorithm investigated in this thesis is proposed in [7]. Their methodology contains much crossover in the domain of feature-matching, however their major contribution occurs in their formulation of the problem. Instead of throwing away low overall confidence, they choose to exploit the color uncertainty by plugging the costs into an MRF, which is designed to make a decision on the uncertainty based on the surrounding context. As an alternative to finding the best

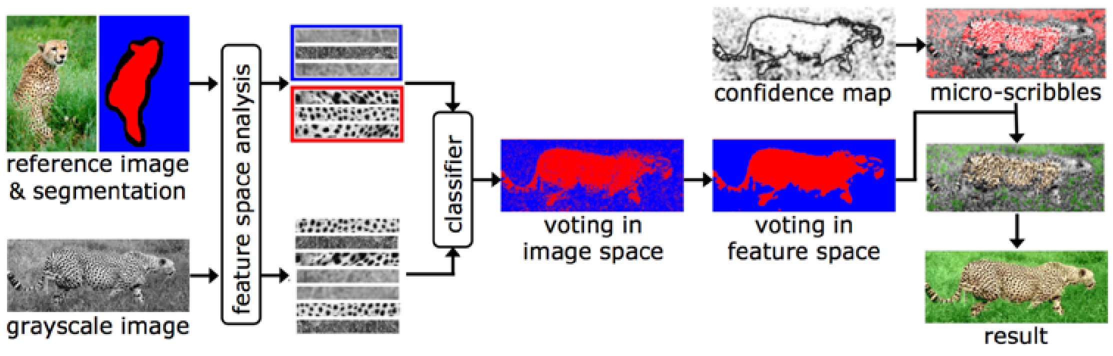


Figure 2.3: Irony et al colorization algorithm

match, the algorithm finds the best several matches, and uses the qualities of these matches to formulate the color confidences at each pixel.

2.2 Analysis of Methodologies

Both the feature-matching and global coherence approaches come with their advantages and disadvantages. Additionally, design decisions in each of the mentioned methodologies carry their own advantages and disadvantages.

The clearest advantage of the feature-matching methods is that they are algorithmically simpler to implement, and tend to perform much faster than their global-coherence counterparts. Feature-matching can be thought of as a special case of the global coherence algorithms, wherein the highest confidence is selected at every pixel as opposed to being used with other matches in a cost function. Since there is no last step for “solving” the colorization, implementations can generally consist of an implementation of a 1-NN classifier with vectors being the defined features, and colors being the ultimate classes.

The simplicity of direct feature-matching is gained at the price of algorithmic robustness. In general, these algorithms tend to fall short on their overall color selections since there is no check on their global coherence. If, for example, a red pixel was placed in the middle of the sky among many blue pixels, there is no way of correcting for this. Even though the red could be very high confidence statistically, it is a poor perceptual choice.

As mentioned, the global coherence algorithms aim to correct for this perceptual correctness issue. Not only do these algorithms make sure that color choices make local sense (by preferencing lower-cost labellings), but they also make sure that regional and global color choices also make sense (by preferencing similar labels in similar regions). In general, the results presented in the papers for global coherence algorithms tend to be more perceptually correct than the results from direct feature-matching algorithms.

Global coherence algorithms tend to perform much slower. Not only is a feature-matching step being run, it is also followed up with some sort of cost minimization techinque which often does not scale well with image size. Additionally, the feature-matching step is not as simple since the cost function must account for all possible color choices at each pixel in the target image, not just the cost function for the best choice at each.

2.3 Takeaways

In this chapter, we have discussed colorization algorithms with both desirable, and undesirable traits. In order to formulate a new colorization algorithm, we must choose which desirable traits to put together in a way that minimizes the undesirable side-effects.

Simply from the results seen in the aforementioned papers, it is clear that the better choice of algorithm class is global-coherence. The goal of this thesis is not to provide a super-high performance algorithm for colorization, but rather a framework for customization (as long as the runtime is bearable). We choose to implement the global coherence using an MRF as there are known efficient algorithms for solving an MRF using graph cuts as described in [8].

Throughout this all, we would also like to maintain some algorithmic simplicity in our approach. In order to do this while sticking with a global coherence algorithm, we decide to use simple features for our images with few dimensions. Additionally, we choose to take some example from [6] due to their use of *multiple* source images. This will need to be modified some as their approach uses multiple source images in order to *reinforce* the scene rather than *compose* it.

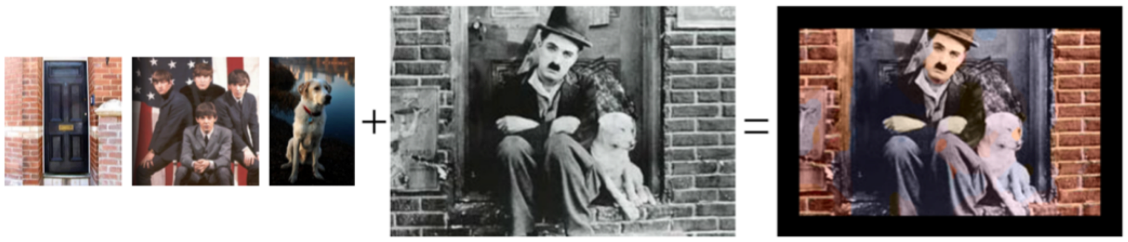


Figure 2.4: Attempted context composition in Charpiat et al

Chapter 3: Framework Overview and Implementation

In this chapter, we provide a detailed look at the way in which the colorization framework is designed. It will begin with some necessary background knowledge on color theory and image transformations, provide an overview of the framework, and then go into detail on each of the points described in the overview.

3.1 Background

Before going into detail about the various methods that were employed to implement the overall colorization framework, there are a few important pieces of background knowledge necessary for laying a base of understanding for the various methodologies.

3.1.1 The $L\alpha\beta$ Color Space

During the course of image colorization, it becomes necessary to work in a color space that separates the *luminance* (intensity) of a pixel from its *chrominance* (color). By separating these channels, it becomes possible to change the color of a pixel without changing how bright it is at the same time. Using such a space also implicitly decreases the amount of data that the algorithm needs to recover.

In our algorithm, we choose to leverage the $L\alpha\beta$ color space. The space is separated into a luminance channel L, and chrominance channels α and β. The α channel of the color space represents the redness/greenness of a pixel, and the β channel represents the yellowness/blueness of a pixel. The two channels are independent, which means that they may be changed without affecting one another (α and β values are mutually exclusive). Additionally, when treated as coordinates in a cartesian space (as a vector: $< \alpha, \beta >$), the Euclidean distance between two colors is proportional to the perceived distance of those colors for a human. This Euclidean distance property will become useful in the final colorization stage, as well as in the analyses of the various results in Chpater 4.

3.1.2 *SLIC* Superpixels

There are several approaches for reducing the computational complexity of per-pixel algorithms such as the one proposed in this thesis. In general, computational complexity can be kept low without sacrificing overall effectiveness by grouping similar pixels into single *superpixels*.

One such method for creating these superpixels is Simple Linear Iterative Clusting (*SLIC*), proposed in [9]. The *SLIC* methodology works by grouping pixels using a weighted distance of absolute image position and color distance (a combination of luminance changes and chrominance changes from the $L\alpha\beta$ space). The algorithm starts by using several pre-computed cluster centers, and then proceeds as K-means would by grouping pixels with their "closest" cluster-center and re-calculating the centers until convergence. Although superpixel contiguity is not guaranteed, it can be enforced should the user desire it.

Figure 3.1 shows the an example output of the *SLIC* superpixel algorithm. Note how similar pixels within proximity to one another are grouped together (note, it is possible to have non-contiguous superpixels). It is important to mention that in our approach, several more superpixels are calculated than shown in figure 3.1.

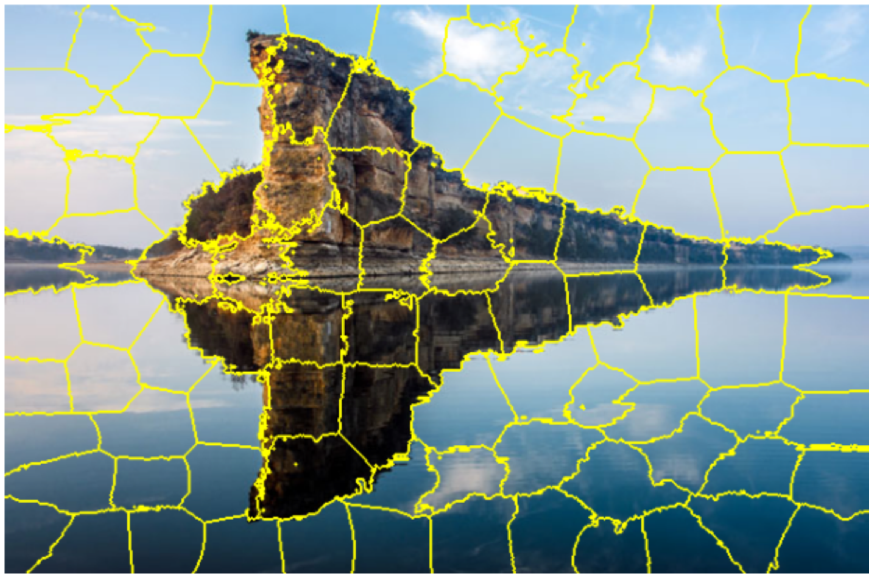


Figure 3.1: An image segmented using *SLIC* superpixels

3.2 Overview

The approach proposed by this thesis is presented in four major stages. In all of these stages, there are several opportunities for customization and tuning. This section is intended to provide a high-level overview of the overall process being proposed. A more detailed description of each of these steps is provided later in the chapter. Figure 3.2 provides a graphical view of our algorithm.

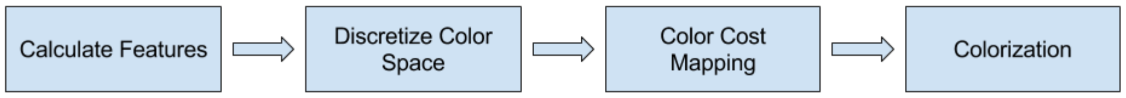


Figure 3.2: Framework Overview

3.2.1 Feature Calculation

The proposed colorization framework was designed with customization in mind, this is especially the case for the calculation of image features. In general, there were two methods for grouping pixels in the calculation of features: SLIC superpixels and 5x5 pixel neighborhoods. On these groupings, features are calculated such as median intensity and intensity variance. The framework is not limited to only the features tested in this thesis, however. The user/implementor may choose to define their own features and pixel regions. The only critical requirement for a feature is that it is agnostic of the color values in the pixel region since these color values will be unknown for the regions in the grayscale image. Features can be as simple as a singular intensity value, and as complicated as SIFT or SURF descriptors. It should however be noted that more complex features will lead to slower performance in later stages of the algorithm.

3.2.2 Discretization of the Color Space

In order to properly select colors for the final image, a subset of the colors in the context must be chosen. There are two key requirements for a useable color palette: it must be able to effectively represent all of the items in the composed scene, and it must also be able to represent subtle variations in color as necessary. In this thesis, the color space is discretized into bins by splitting the α and β channels of the $L\alpha\beta$ color-space versions of the source images. This discretized palette will be used as the basis for colors to pick from in the latter stages of the framework.

3.2.3 Color Cost Mapping

Before picking the final colors in the target image, it is first useful to build a listing of possible colors and their likelihoods at each pixel. This is arguably the most important part of the algorithm as it is what will ultimately inform the choice of color in the final step. The algorithm proposed in this paper formulates a cost function for the selection of one of the discretized colors at each pixel depending on the dimensionality of the features calculated in the first phase. This cost function is built on top of the number and quality of matches between features in the target image, and features in the source images. The system rewards both a large number of matches and high quality matches. The cost of a color selection is built to reflect the overall confidence of a genuine match, which means that this cost function is not to be confused with a probability density function. A probability density function would always have the confidence values for each pixel summing up to one, our cost function will not reflect actual probabilities.

3.2.4 Final Colorization

There are two goals for a proper overall colorization of a grayscale image: local coherency, and global coherency. For local coherency, it is important that the final colorization honors the loss function that was calculated in the previous step, and does not deviate too far from the lowest-cost choices. For global coherency, it is important that the final colorization maintains realistic results for large color regions.

For example: if the sky is being colorized in an image, most of the pixels are going to have a very low-cost for the color blue; however, there may be a few misplaced red pixels in the middle of the sky whose high confidence value must be ignored. In order to achieve these two goals, the colorization is formulated as an energy minimization problem in two parts.

- Local coherency is enforced by placing higher energy with color predictions that come at a higher cost according to the cost function calculated in the previous step.
- Global coherency is enforced by placing a higher energy with adjacent color predictions that are perceptually far apart in the $L\alpha\beta$ color-space.

A multiplier λ is placed on the global coherency which can be raised or lowered depending on a preference to the global or local coherence statistics.

3.3 Feature Calculation

The calculation of image features is one of the most consequential steps in our proposed framework. These features are what will ultimately determine matches in section 3.5. In general, the proposed framework was implemented on three different types of features: *average luminance*, *luminance variance*, and the *2-Vector* (Figure **??**). Because these statistics are used for matching, they are calculated for the target image, as well as all of the reference images in the context.

In our framework, features are calculated over groupings of pixels and are represented by some type of summary statistic (e.g. mean or median), or a statement of differences over the pixel region (e.g. variance of luminance values). In general, the overall complexity of the algorithm is directly proportional to the complexity of the features being used, so our tests stick with simple features that are easy to calculate. Each type of feature has two important parts: the representation of the statistic and the pixel region over which it is calculated. In the following sections, an overview of the various statistics that were used is provided.

3.3.1 Average Luminance

The luminance feature is by far the simplest feature to calculate over an image. The goal is to calculate the mean luminance value of every pixel region in the image, so the only necessary design decision is to determine the type of pixel region to use for calculation. In the case of luminance, superpixels were chosen as the calculation region since they provide several advantages over simple pixel neighborhoods:

- In the case of the target image, the luminance values are the only hint available as to the proper segmentation of the image. There is also a reasonable expectation that pixels with similar luminance values over a contiguous region will have the same or similar chrominance values.

- In the case of the source images, superpixels do not separate the correlation between chrominance and luminance in contiguous regions. A simple 5x5 pixel neighborhood destroys this correlation as pixels are not grouped by any similarity criteria. This information is important to retain as it can provide a better mapping between the luminance statistic in the target image and the ultimate color that is chosen.

- When the framework ultimately comes to the matching of pixel regions from source to target, it is important that the statistic used for grouping the pixels both take into account the same criteria for contiguity. *SLIC* superpixels consider the luminance of a pixel as one of the criteria for grouping pixels (and is in fact the only criteria for black-and-white images outside of spatial considerations).

For the target image and all of the source images in the context, the luminance values from the L channel of the $L\alpha\beta$ color space are used for the overall calculation of features in each superpixel.

3.3.2 Luminance Variance

The aim of the variance statistic is to carry more texture information with it to the matching section of the framework. The statistic is built with the goal of serving as a more descriptive alternative to pure luminance values while still maintaining the single-dimensionality of the feature vector as well as the ease of feature calculation. For the target and source images, the values used for variance calculations are extracted from the L channel of the $L\alpha\beta$ color space version of the images. For the target image, we calculate the variance at every pixel (using overlapping 5x5 neighborhoods as described in the next paragraph). For the source images, we calculate the variance at every fifth pixel along both the rows and columns, creating a 5x5 grid across the image.

For the variance statistic, we no longer use superpixels, but rather switch to the use of 5x5 pixel neighborhoods. This mostly follows from the side-effects of the *SLIC* algorithm. *SLIC* is built using a modified K-means methodology; this is not a concern for luminance values where the pixel grouping methodology should be built with the similarity of contiguous regions in mind so that an averaging of luminance values does not deviate too far from any of the pixel values in the superpixel region. K-means is designed to minimize the total mean squared error of data points to their assigned cluster center, or simply to minimize the total variance of the clustering. Minimizing the total variance of our pixel clustering would render the variance statistic ineffective; 5x5 pixel neighborhoods are agnostic to the values surrounding a chosen pixel, which preserves the meaning of the luminance variance as a texture representation.

3.3.3 2-Vector

The final feature that was used to test the framework was a 2-dimensional feature vector. Both Luminance and Variance carry benefits and drawbacks of being the single-dimension feature.

- Luminance features are very easy to calculate, and any image that is worthwhile to colorize is guaranteed to have some variance over the domain of luminance values. However, luminance values vary with lighting, and thus matching is not always a straightforward procedure.

- Variance features provide a lighting-invariant description of a pixel neighborhood, and are a good descriptor of texture. However, the shortcoming is that lighting-invariance can cause problems for incorrectly matching similar textures that appear in different lighting scenarios.

As is evident, both methods contain a major shortcoming that is overcome by the other. Variance statistics overcome the lighting-sensitivity of luminance features, and luminance statistics overcome the lighting-invariance of variance features. So it is only natural to combine these two statistics into a single feature vector.

The first dimension of this feature vector is the direct luminance value of the pixel in question. The second dimension is the variance of the luminance values of the 5x5 neighborhood. Once both dimensions are calculated, the first dimension is normalized to the range of values in the second dimension so that both dimensions of the vector exist in the same range of values for each vector.

3.3.4 Customization Opportunities

Although three specific methodologies for calculating features were explored in the proposed framework, these features are not hard-coded. It is possible to program the framework in such a way to calculate any type of feature over any type of pixel neighborhood. More detail will be discussed in the future work section, however it should be noted that much more complex features were considered for testing. The features can be programmed as SIFT or SURF descriptors, and possibly even more rich information. The only tax on complex features is a decrease in overall algorithm performance in the cost mapping phase (described in section 3.5).

3.4 Discretization of the Color Space

In order to begin the process of colorization, it is important to build a palette of colors to choose from when overlaying the eventual colors on the target image. From a theoretical standpoint, the α and β channels of the $L\alpha\beta$ color space are continuous over their entire domain; when properly represented in memory, an image in $L\alpha\beta$ will use floating point numbers to represent each color channel. Although the infinite possibilities could be useful for a highly detailed approach to colorization, this is not practical from both a human and algorithmic view of the problem. To build a usable palette of colors, it is necessary to break the color space into discrete bins, *i.e.* to *discretize* the space.

As mentioned earlier in the chapter, the selected color palette must meet two major requirements.

1. The color palette must be capable of representing the overall color "mood" of the scene being colorized.

2. The color palette must be capable of capturing small and subtle color variations that happen often.

When treating α and β as the axes of a Euclidean space, the palette can be represented as a set of rectangular bins with the edges of the rectangle being parallel to the α or β axes. The approach to discretization is formulated in a way that meets the two goals of the overall color palette. A proper colorization will encapsulate all of the points in the space to a bin, and will have smaller bins in areas where there is a high density of points in the color space. These goals follow directly from the Euclidean distance property of the $L\alpha\beta$ color space. Colors that are perceptually close will also appear close in the Euclidean representation.

The discretization procedure is as follows:

1. Set $channel = \alpha$

2. Convert the image or images into their color-space representation. Figure 3.3 illustrates this representation.

3. For each bin, split into two bins along *channel* at the median point along the corresponding axis. Figure 3.4 illustrates this step over the first fiew iterations.

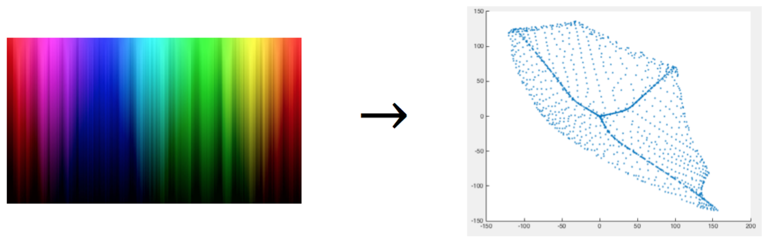


Figure 3.3: Conversion of an image into its Euclidean color-space representation

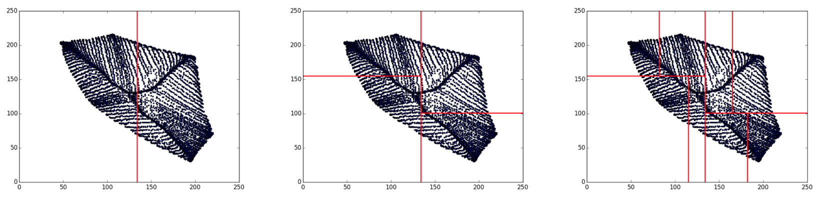


Figure 3.4: First three cuts on the rainbow distribution

4. Switch *channel* between α and β.

5. Go back to step 3 if the number of bins is not as large as desired.

6. Calculate the representative color for each bin.

When the discretization is run on a large number of bins, the result will look much like Figure 3.5. Note how the discretization has met our two goals. In the parts of the space with a higher density of points, the bins are smaller and more concentrated, and areas with lower densities are still encapsulated, but in larger bins. This approach takes some example from the KD-Tree data structure, notably by splitting along alternating dimensions.

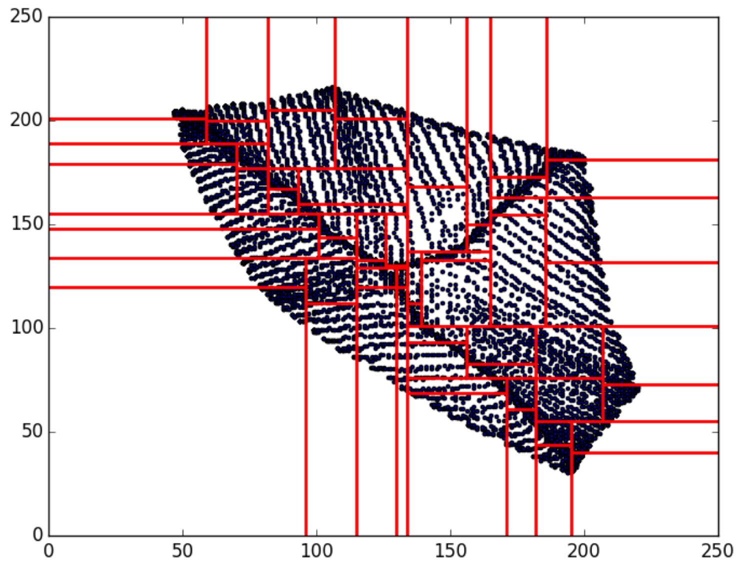


Figure 3.5: Final discretization on the rainbow distribution

The representative color for each bin is calculated as the median value of all points in the bin. So the representative α is the median α value of all points in the bin, and the representative β value is the median β value of all points in the bin (ultimately, the results do not change significantly if mean is used instead). For the later steps of the algorithm, each bin will be given a label so that the colorization can be treated as a pixel-wise classification problem.

3.5 Color Cost Mapping

The color cost mapping step in the framework generates the cost function that will ultimately be used as input to the final colorization. The goal of this process is to generate a per-pixel cost function that expresses the cost of associating a specific color with a given pixel.

The general approach to our cost function is based around the matching of features from each image in the context to the features in the target. Fundamentally, we create a search space over some subset of the pixel neighborhoods in each reference image. Within this subset, our cost function will be some mathematical statement about the characteristics of the matches in our search space that also exist in the color bin in question. The formulation of this statistical statement is ultimately determined by the dimensionality of the feature vectors calculated in the first step of the algorithm.

3.5.1 Scalar Matching

In the case of a scalar, we can avoid several of the complexities that come with the use of multi-dimensional vectors. Our biggest advantage comes with the way in which the search space for matches can be limited. Instead of running over the entire feature space to find our best matches, we can "bin" our values into a histogram. For some pixel p from the target image, our cost function becomes a summary of the source image pixels in the histogram bin that p falls into.

The cost function is formulated as follows:

1. Generate a histogram H_i for all of the pixel regions in each of the source images. For the average luminance feature, each bin is fixed to a width of 5 (*i.e.* [0-5), [5-10)...*etc.*); for the luminance variance statistic, the histogram is built with 50 bins of equal depth (as opposed to equal width which resulted in a heavy performance penalty).

2. For each pixel in the target image, build the match score.

$$S(c|p) = \sum_{i \in \mathbf{I}} \frac{\left|\left\{q : q \in H_{i,b(p)} \wedge c = b_c(q)\right\}\right|}{\left|H_{i,b(p)}\right|} \tag{3.1}$$

Where $S(c|p)$ is the score for the given color bin c at the given pixel p in the target image; $\mathbf{I}$ is the set of all source images; $H_{i,b(p)}$ is the set of all items in the bin $b(p)$ from the histogram of source image i; and $b_c(q)$ is the color bin of the pixel q from the source image.

3. Normalize the histogram to sum to 1 for each pixel in the target image.

$$N(c|p) = \frac{S(c|p)}{\sum_{d \in \mathbf{C}} S(d|p)} \tag{3.2}$$

   Where $\mathbf{C}$ is the set of all color bins.

4. The cost function is formulated as

$$C(c|p) = 1 - N(c|p) \tag{3.3}$$

It is important to note that even though the values are normalized as such for a single source image, the scoring function should still not be confused with a proper probability density function (pdf). When the scores are summed across each of the source images, we lose the representation of a probability.

3.5.2 Vector Matching

In the case of multi-dimensional vectors, we lose our capability to properly "bin" values along a single dimension. With this being the case, we instead need to more judiciously choose the matches with which to work, and we need to take better account of how "good" a match is. Instead of using a histogram as we did for scalar features, we now use k-nearest neighbors to limit the set of matches our cost function is calculated over.

In order to properly formulate the cost function, the algorithm must take into account two things:

- A large number of matches of a certain color should indicate a higher likelihood of the given color being the final label.
- Matches with large distances between the matched feature vectors should be given less consideration.

We must also make sure that our search space is large enough to provide a rich set of selections, yet small enough to avoid high time-complexity in this stage of the framework. We use K-nearest neighbors to build our search space; the cost function is calculated as follows:

1. For each pixel in the target image, calculate the match score for each color bin based on each source image.

$$S(c|p) = \sum_{i \in \mathbf{I}} \left(\begin{cases} \frac{\left|M_{i,b(c)}\right|^2}{\sum_m^{M_{i,b(c)}} \|\mathbf{v}(m) - \mathbf{v}(p)\|}, & \text{if } \left|M_{i,b(c)}\right| > 0 \\ 0, & \text{otherwise} \end{cases} \right) \tag{3.4}$$

   Where $S(c|p)$ is the score for the given color bin c at the given pixel p in the target image; $\mathbf{I}$ is the set of all source images; $b(c)$ is the indicator function for the bin to which color c belongs; $M_{i,b(c)}$ is the set of matches from source image i from the K-nearest neighbors to pixel p that exist in the same color bin as color c; $\mathbf{v}(p)$ is the feature vector from pixel p.

   The rationale behind the scoring piece of the calculation is to "reward" matches that happen often, and at shorter distances. In this case, it is the number of matches for the given bin, divided by the average distances for a match in that bin. If no matches are found, the score defaults to 0 for the given bin at the pixel in question.

2. Normalize all of the values in the match scores with respect to the highest overall score.

$$N(c|p) = \frac{S(c|p)}{\max\limits_{c,p} S(c|p)} \tag{3.5}$$

   Note the difference in formulation from the single-dimensional case.

3. The cost function is formulated identically to equation 3.3 with respect to $N(c|p)$.

3.5.3 Scene Composition

It is also important to note that the formulation of the scoring function in equation 3.4 and equation 3.1 is what allows us to compose a scene. Instead of calculating the match score over the entire space of images at once, the score is calculated per image and then summed together. In this way, images that provide for poor matches no longer dilute the matches for each pixel, but rather contribute less to the overall matching score (through larger distances between feature vectors). Instead of calculating the scores *over* all images in the context, the scores are calculated *through* the source images.

3.5.4 Customization Opportunities

Although the framework was only tested with the particular cost function described in this section, it was built so that any cost-function calculation can be used. These cost functions can possibly be changed to better reflect the type of features being used. For example, one may want to use a different formulation for vector distances (*e.g.* cosine distance); or the costs could be scaled differently to further separate higher costs from lower costs. The only restriction is that the cost formulation cannot be dependent on other pixels in the target image; this raises ambiguity with respect to the order in which the costs are calculated.

3.6 Final Colorization

With our hands on the cost function, it is now time to resolve the function into the final colorized image. Unfortunately, this is not as simple as selecting the lowest cost color at each pixel. With the imprecisions of the calculations in the color cost mapping phase, we are not able to blindly rely on the cost function for our selections. We do, however, use it to guide our final color selections.

As mentioned earlier, there are two important aspects to a proper colorization:

Local Coherence The eventual color selections must comport with the calculated cost function to a certain degree.

Global Coherence The color selections across pixel regions should reflect some contiguity (*i.e.* the final colorization should be able to correct for mistaken cost calculations).

As they are stated, these goals can be formulated into an energy function. We can enforce local coherence by summing up the dissimilarity of color selections with their associated cost from the cost function. The global coherence can be enforced by penalizing pixel labellings that vary over small regions (and vary largely according to the $L\alpha\beta$ Euclidean distance property). Our energy function is as follows:

$$E(L) = \sum_{p \in T} C(L_p|p) + \lambda \sum_{q \sim p} \frac{\|L_p - L_q\|_{L\alpha\beta}}{\max\limits_{b_i, b_j \in B} \|b_i - b_j\|_{L\alpha\beta}} \tag{3.6}$$

Where $E(L)$ is the energy associated with the color labelling L of the target image; p is a pixel in target image T; L_p is the chosen color for pixel p in the labelling L; B is the set of all color bins; λ is the smoothness constant; $p \sim q$ is the generator for each 4-neighborhood pair of pixels; and $\|L_p - L_q\|_{L\alpha\beta}$ is the $L\alpha\beta$ color distance between L_p and L_q.

The ideal goal is to find the labelling that results in the minimum energy for equation 3.6. Finding the absolute minimum is an *np-hard* problem, which poses a significant performance concern for data on the scale of an image with 20 possible labellings at each pixel. In order to retain our performance, we can formulate the energy function as a Markov Random Field (MRF). This field can be solved efficiently with graph cuts to a known proximity of the global minimum [8].

Once the best labelling is determined, the actual colorization of the target image becomes a matter of transferring the α and β values corresponding with each label to the labelled pixels.

Chapter 4: Results and Analysis

With a complete understanding of the framework and its underlying algorithms, we now explore the results. This chapter starts by introducing *how* we will be evaluating the framework, as well as *why* this model is an acceptable measure of the framework's performance. We will then provide a breakdown for the results of our algorithm as they vary by changing things at the various customization points of the framework. We also provide a brief overview of algorithm performance. We then wrap up with an analysis of the presented results.

Our results are broken up into sections based on the different customizable pieces of the framework. In section 4.2 we explore the results of changing between the three feature types described in chapter 3. In section 4.3, we look at the ways in which changing the size of the context changes the results of the colorization. Section 4.4 describes the effect of changing the smoothness constant in the MRF formulation (described in chapter 3.6). It is first important, however, to understand how and why we are evaluating and analyzing the results the way we are.

4.1 Evaluation Methods

When it comes to properly evaluating the effectiveness of a colorization, there are two types of "correctness" that we can rely on: 1) Perceptual Correctness, and 2)

Quantitative Correctness. This section explains these two concepts, and additionally provides some insight into the idea of *self-colorization.*

4.1.1 Perceptual Correctness

The perceptual correctness of a colorization is a human-centered view of the algorithm's effectiveness. In this case, we are trying to determine how close a colorization is to what a human might reasonably expect to see based on their own personal experience with the world and the colors in it. Naturally, this is not something that is easily quantified as it will most certainly change depending on the human evaluator. With all of this considered, it is still necessary to evaluate our algorithm with a qualitative eye since colorization is such a perceptual process. In order to formalize such a qualitative review, we will stick to a few questions for our analysis:

- In general, do the selected color regions appear as one would expect them to? *i.e.* the sky is blue, the grass is green, concrete appears gray, *etc.*
- Is the colorization *crisp*? *i.e.* Do color regions start and stop where one would reasonably expect them to?
- Do all of the colors make sense in the scene? *i.e.* Are we seeing any colors that simply don't belong in the picture?

These questions are by no means comprehensive. They represent, however, several features of a colorization that are too hard to quantify but are important aspects of a "correct" coloring.

4.1.2 Quantitative Correctness

For the sake of comparison, we must also have a quantitative measure of a colorization's succes. For this measure, we will need to take into account how perceptually "far" the colorized image is from its ground truth. We formulate this as follows:

$$D(L) = \frac{\sum\limits_{p \in T, q \in GT} \|L_p - C(q)\|_{L\alpha\beta}}{|T|} \tag{4.1}$$

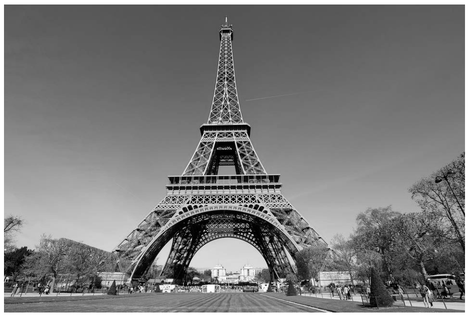
Where $D(L)$ is the average per-pixel distance from the ground truth; T is the colorized target image; and GT is the ground truth colorization of the target image.

This formulation as an average per-pixel distance is preferrable to a standard misclassification rate in that it takes into account the "closeness" of an incorrect labelling.

4.1.3 Self Colorization

With the number of pieces in the algorithm that can be customized, there are a very large number of framework configurations that can be used. In order to isolate as many variables as possible, we employ a simple technique. Except in the case of using multiple reference images, our tests are run by performing a *self-colorization.* A *self-colorization* is done by colorizing an image with its own ground truth. This technique allows us to avoid the issues associated with the selection of a "good" context; and isolates whichever variable is being tested.

For this chapter, we will be performing all of our self-colorizations on the picture of the Eiffel Tower from figure 4.1.


(a) Grayscale image of the Eiffel Tower

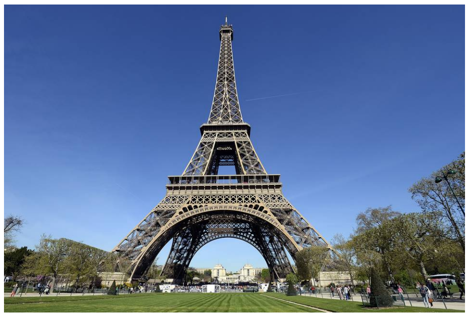

(b) Color image of the Eiffel Tower

Figure 4.1: The source and target images being used in self-colorizations

4.2 Feature Type

As described in chapter 3, there are three different types of features that were used to test the framework: 1) Average Luminance, 2) Luminance Variance, and 3) 2-vector. This section will present the results of a self colorization on a picture of the Eiffel Tower using the three different features. Across all of the features, 20 color bins are used, and λ values are chosen which provide the best perceptual results for each feature (using identical lambdas results in a bad comparison since each cost function is calculated differently).

4.2.1 Average Luminance

The average luminance feature is the first feature tested with self-colorization. For a more detailed explanation of this feature, please refer to section 3.3.1.

When running a self colorization on our image of the Eiffel Tower with $\lambda = 0.0$ and 20 color bins, we get the result shown in figure 4.2.

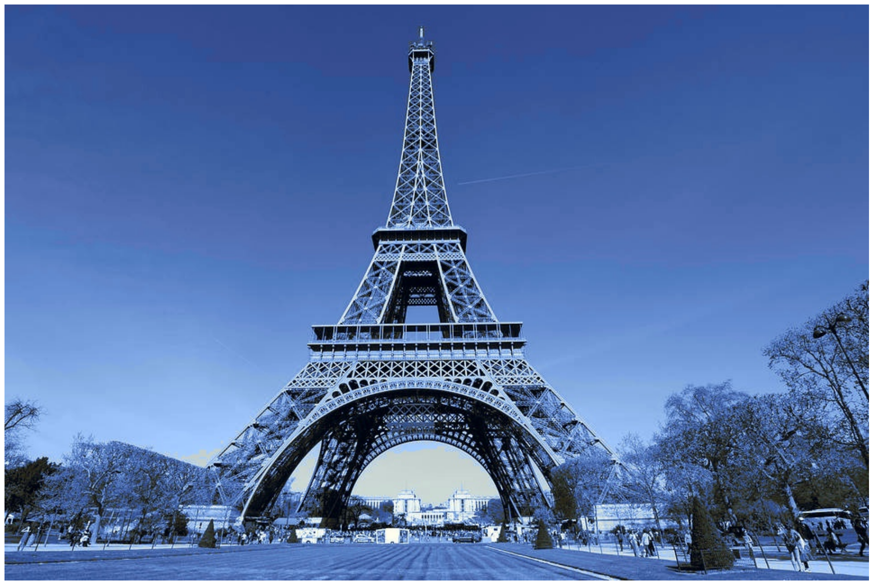


Figure 4.2: Self-Colorization of Eiffel Tower using average luminance feature

For this colorization, the average per-pixel color error was 13.165.

4.2.2 Luminance Variance

The luminance variance feature is the second feature tested with self-coloriation. For a more detailed explanation of this feature, please refer to section 3.3.2.

When running a self-colorization on our image of the Eiffel Tower with $\lambda = 1.5$ and 20 color bins from the discretization, we get the result shown in figure 4.3.

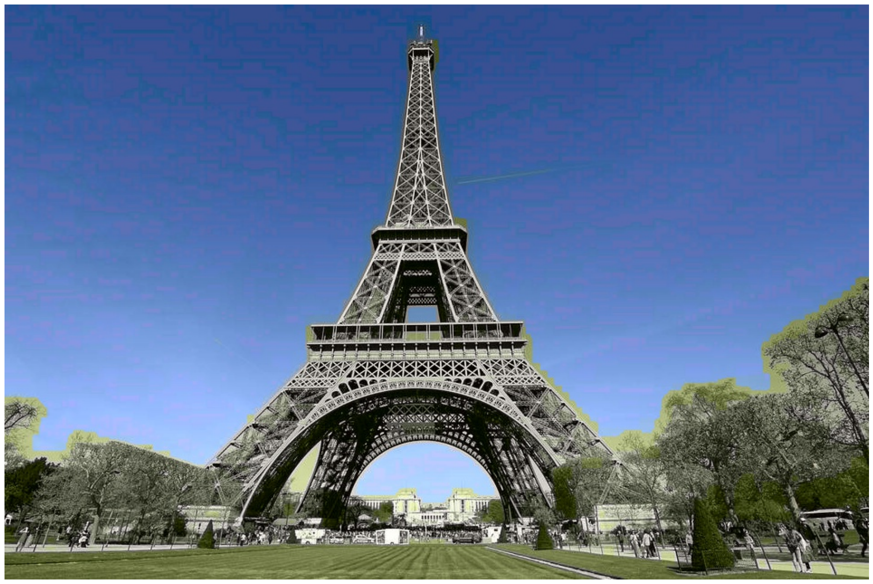


Figure 4.3: Self-Colorization of Eiffel Tower using luminance variance feature

For this colorization, the average per-pixel color error was 6.936.

4.2.3 2-Vector

The luminance variance feature is the final feature tested with self-coloriation, and is our only multi-dimensional feature. For a more detailed explanation of this feature, please refer to section 3.3.3.

When running a self-colorization on our image of the Eiffel Tower with $\lambda = 0.025$ and 20 color bins from the discretization, we get the result shown in figure 4.4.

For this colorization, the average per-pixel color error was 7.995.

4.2.4 Analysis

If there is anything that is most important to glean from the feature tests, it is the very large effect that feature choice has on the final image colorization. In the specific

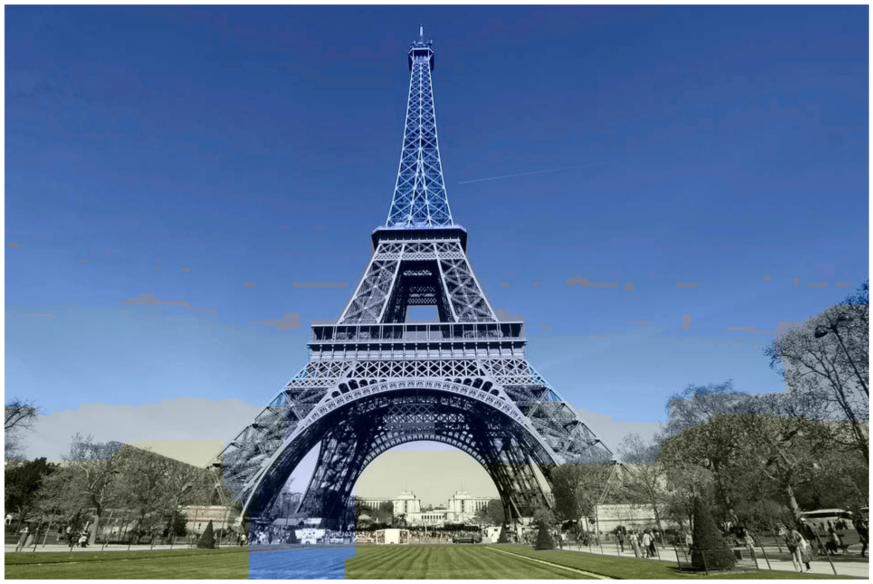


Figure 4.4: Self-Colorization of Eiffel Tower using the 2-vector feature

case of our Eiffel Tower image, changing features nearly doubled our colorization effectiveness. We will analyze the results of each of the features individually in this section, and also provide some insight into why we believe the results turned out the way they did.

What is blatantly clear about the luminance feature is that it is a fairly ineffective standalone feature for the purpose of colorization. In our case, the matching score from the color cost mapping was washed out by the sheer number and variety of pixels in the sky portions of the image. The only colors that eventually transferred were the ones found in the sky, and a couple instances of green from the grass. This “washing out” effect is likely due to the numerous ways in which the entire luminance “mood” of a picture can vary. It is also a likely side-effect of the lack in descriptiveness of a grayscale value for its backing color: thinking in the RGB color space, the number of RGB combinations that can yield an intensity of 127 (for example) is enormous;

a mapping from grayscale to color is likely to vary within a single color image on its own. Needless to say, the quantitative performance of this feature is poor, and from the standpoint of perceptual correctness, this colorization only passes a single one of our "questions".

The luminance variance feature nearly doubled the quantitative performance of the average luminance feature. From a human perspective, this colorization is clearly more plausible than that of the average luminance statistic. When analyzing the perceptual correctness, we very clearly pass our first and third criteria, the only failed test is the crisp colorization. Quantitatively, the average color error is less than 7, which is the difference between a dark blue, and a slightly less dark blue for example. This improvement is likely the result of the ability of luminance variance to convey texture information. Unlike the case with pixel brightness, the texture of a pixel region is much less likely to represent many different colors throughout a single image. The result being that matches during the color cost mapping phase of the algorithm are much more likely to carry over the "representative" color of the accompanying texture.

The performance of the 2-vector is much more in-line with that of the luminance variance feature. It is interesting however that a more descriptive statistic would be outperformed by a single-dimensional feature. In our perceptual tests, our third criteria still passes, however criterias one and two are not met (note the patch of blue in the grass, gray patches in the sky, *etc.*; it should be noted however that the "crispness" of the colorization shows promise in some areas of the image). Additionally, the quantitative score is slightly worse than that of the luminance variance colorization. Although it cannot be stated with certainty, it is very possible that this is in part due to the innefectiveness of the average luminance feature. Half of the matching score is determined by a statistic that we now know is not good at informing color decisions.

Of the three statistics discussed in this thesis, test results (including ones not mentioned herein) confirm that luminance variance tends to provide the best colorizations both quantitatively and perceptually. This is not to say that more complex features could perform even better; in fact many of the papers discussed in chapter 2 show better overall performance using highly complex features that are combinations of SIFT or SURF descriptors and other texture information. However, using simple luminance does not appear to benefit the colorization, and even potentially harms it in some cases. Given the success of the luminance variance statistic, we will be using it for all of our remaining benchmarks.

4.3 Context Size

As was mentioned in chapter 3.5.3, the formulation of the cost function is designed to account for the composition of a scene with both functions 3.1 and 3.4. In order to test this effectiveness, we will run a colorization with an increasing number of source images and determine the result of the increased context size.

For this specific set of results, we will be using the cliff image in figure 4.5, and subsets of varying size from the images in figure 4.6.

Five different experiments were run to find the efficacy of increasing context size to improve colorization. The image in figure 4.5 was colorized using an increasing number of selections from the images in figure 4.6. For all of the colorizations, the number of color bins was fixed at 20, and λ was kept constant at 0.5.

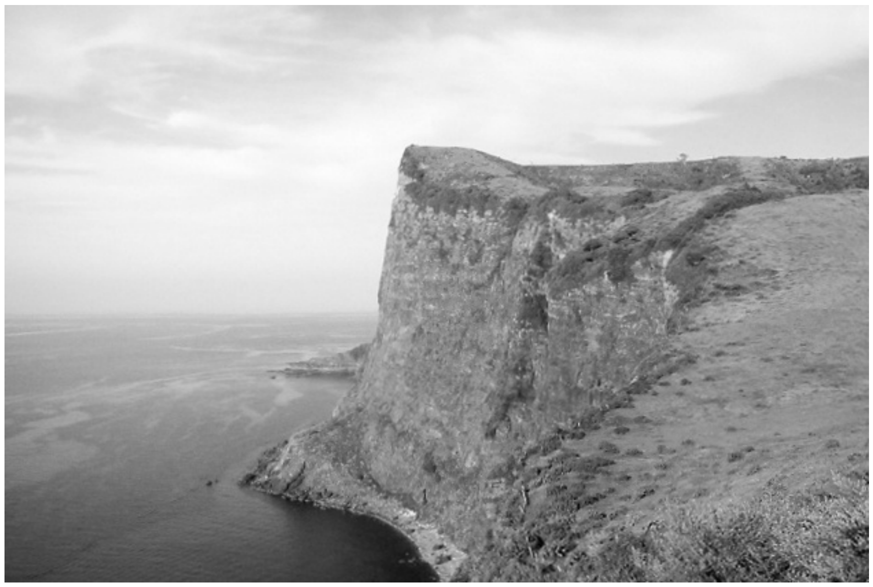


Figure 4.5: Base image for testing colorization with various context sizes

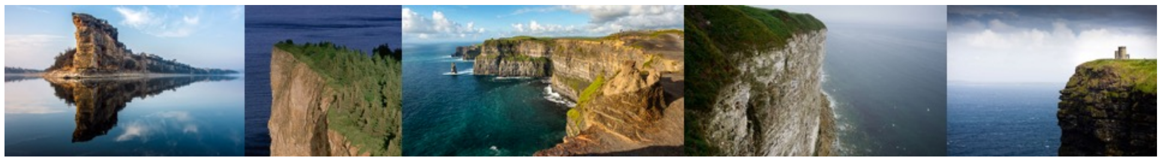


Figure 4.6: Context selection for colorization experiments with context size

Figure 4.7 shows the cliff as it was colorized in order from smallest to largest context. Figure 4.8 provides a graphical representation of the quantitative correctness for the colorization compared to the ground truth image. For reference, figure 4.9 shows the ground truth colorization of the cliff image.

4.3.1 Analysis

One of the main goals for our formulation of the colorization process was to provide for a way to properly use multiple source images in a colorization. In general, the

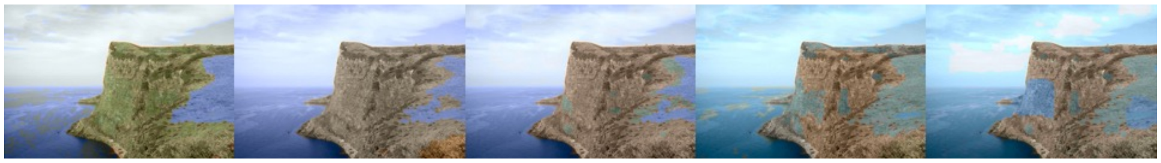


Figure 4.7: Cliff colorization results

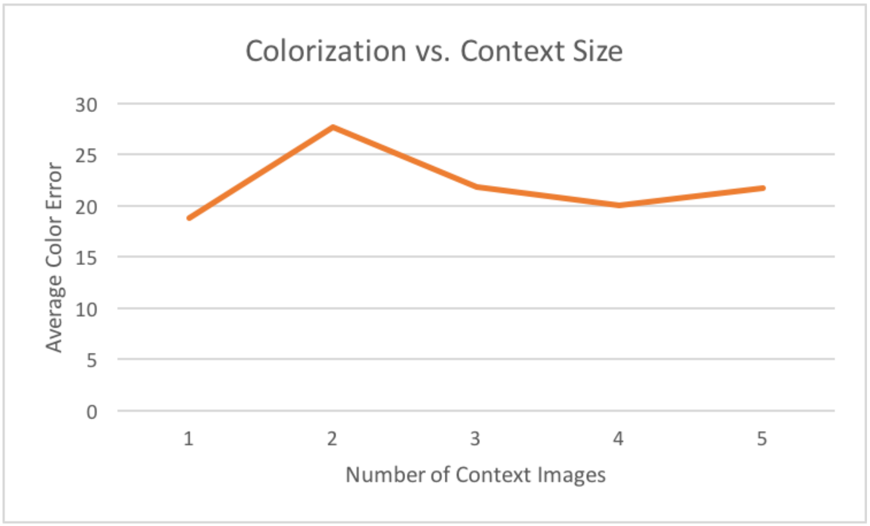


Figure 4.8: Quantitative results of changing context size for colorization

procedure is meant to formulate match scores *through* the context images as opposed to *across* them. From the above results, it appears that this approach has not been entirely successful. There does not appear to be any overall positive or negative quantitative trend associated with an increase in the number of source images for a colorization. In fact, the best colorization score in our case was associated with a single-image context.

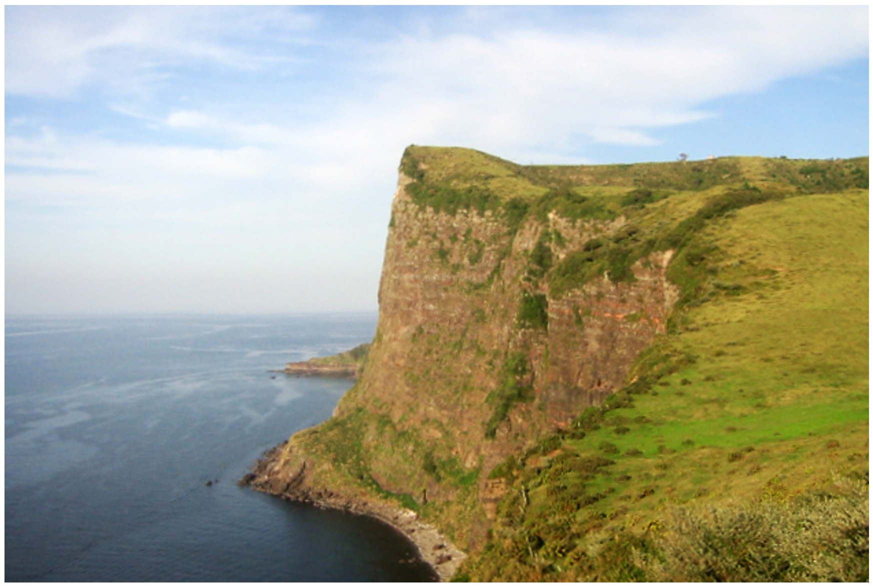


Figure 4.9: Ground truth colorization of the cliff image

What these results show us is that the process of image colorization by example is heavily dependent on the human selection of source images. Even for a single-image context, the colorization of our image in this section would be fairly different depending on *which* of the five images in figure 4.6 is selected as the context.

From the standpoint of perceptual correctness, we start to see one of the shortcomings of the luminance variance feature: it is not descriptive enough for a proper colorization with large uniform regions. This is especially evident with the portions of land in the image that are colored blue because their texture is similar to that of the water.

4.4 Smoothness Constant

The final colorization stage of the algorithm (described in section 3.6) is where we make the actual color selections for the target image. Although the calculation of the local and global coherency values are fixed, we can still modify the smoothness constant λ to preference local vs. global coherency. In this section, we re-perform the self-colorization of the Eiffel Tower image with the luminance variance statistic using 20 color bins; λ is varied to find the resulting quantitative error.

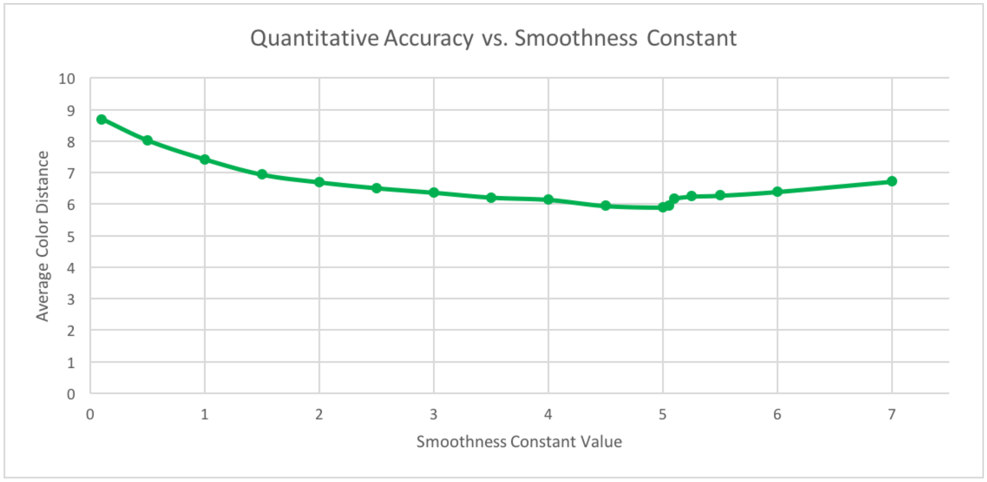


Figure 4.10: Quantitative results of changing the smoothness constant

Figure 4.10 presents a graphical view of the results from varying the smoothness constant λ. λ was varied between 0.1 and 7, and the quantitative correctness was calculated at various intervals. The error reaches a minimum at $\lambda = 0.5$. Figure 4.11 shows a sampling of the results from the colorization tests used to generate the chart in figure 4.10.

4.4.1 Analysis

Unlike our previous two tests, the smoothness constant has a much more subtle effect on the colorization of an image. As mentioned previously, λ changes the weight of the global coherence portion of the Markov Random Field equation; smaller values

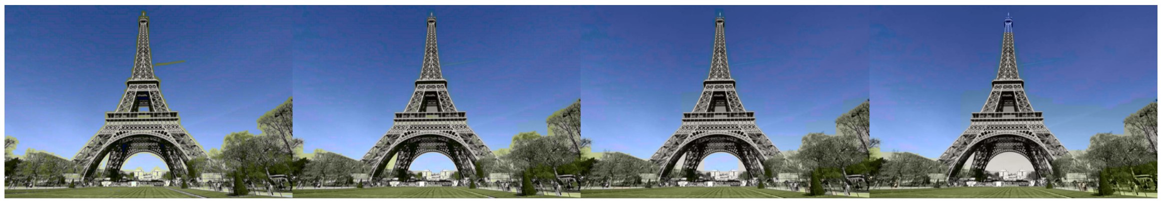


Figure 4.11: Sampled results from smoothness constant tests

will place a larger preference on the cost function calculated during the color cost mapping, and larger values will place more preference on the "smoothness" term. As can be seen in the results from this section, there are trade-offs associated with selections of both big and small λ values.

Using the quantitative error makes the trade-off between local and global coherence very clear. There is a price for relying too heavily on the cost function, as wekk as a price for not trusting it enough. The higher error rates for small λ values can be explained in the context of feature-matching algorithms described in the previous-works section (in fact setting $\lambda = 0$ is much like feature matching). One of the issues discussed with direct feature-matching was the tendency to blindly follow the cost function; which means any imprecise matching will result in an incorrect color selection. As we increase λ and move away from the cost function, it follows naturally that the quantitative error would decrease. It also follows naturally that there is a point at which ignoring the cost function can lead to problems.

Generally speaking, the perceptual correctness maps fairly well to the quantitative correctness. With heavy preference on the local coherence, we see many "spots" of incorrect color in the image resulting from bad matches during the color cost mapping phase. As λ is increased, these low-confidence spots tend to be overtaken by the broader color consensus in the pixel region. However, after a certain point, these regions will begin to overtake pixels beyond their boundary, leading to color “bleeding”.

4.5 Performance

Although our algorithm was not designed with performance as our first goal, some steps were taken throughout the design process to ensure that the algorithm runtime was reasonable for use and testing. This section provides a brief description of the performance of our algorithm in various scenarios. All results were obtained on a Mid-2014 MacBook Pro with a 2.5 GHz Intel Quad-Core i7 and 16 GB of RAM.

Generally speaking, there are two major factors that affect the performance of our algorithm: context size, and image feature type. In order to make a fair comparison, all tests on changing context size were done using the variance feature, whose color cost mapping phase only utilizes one core on the processor. Comparing across feature types is not fair due to the way in which color cost mapping was implemented (with parallel processing utilizing all cores on the CPU) for multi-dimensional features. Suffice to say, scalar features perform much faster in color cost mapping than multi-dimensional features, hence the need for parallelization in 2-vector color cost mapping.

The cliff images of section 4.3 were used to perform timing tests against context size using the luminance variance statistic. Figure 4.12 shows the runtime as it changes with the size of the colorization context. It should be noted that runtime increases at a roughly linear rate with respect to the number of source images in the context.

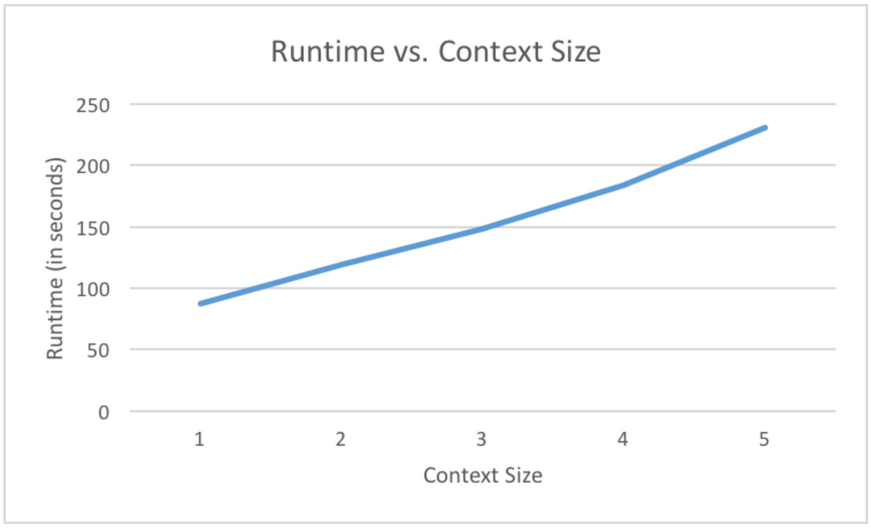


Figure 4.12: Runtime changes with respect to the size of the context

4.6 Final Analysis

Although some of the results of self-colorizations present promising results, our framework still has a long way to go before it can colorize images as well as the claimed results of the current state-of-the-art algorithms. As shown in this section, the selection of feature, context size, and global vs. local coherence preference can dramatically change the effectiveness of a colorization.

From our results, we can see that a new formulation of the color cost mapping may be necessary to handle the use of more than one source image to enhance a colorization. The results further indicate that our mapping algorithm falls short in a way that could cause problems for a scene composition: the ultimate goal of a large source image set. These issues with context sets can also be attributed to the feature selection.

Arguably, the most encouraging result is the effectiveness of the simple variance statistic in colorization of fairly complex scenes such as architecture. We attribute this success to the descriptiveness of variance for texture, which tends to carry more unique color information with it. However, with large context sets, we started to see some of the issues associated with using only one piece of texture information on a single un-changing scale. The need for scale invariance is what makes the popular choice of SIFT and SURF descriptors clear. Without scale invariance, large-scale textures can be mistaken for small ones and the same problem is encountered from small to large. Ultimately, the success or lack thereof for a colorization is determined by the choice of image features, the selection of source image(s), the formulation of the color cost function, and the resolution of the energy function to the final colorized image; all of these features are customizable within our framework.

Chapter 5: Conclusion

5.1 Summary

Grayscale image colorization is a problem with many wide-ranging applications, from restoring color to historical photographs, to colorizing entire old movies. This thesis presents a new framework for marshalling the image colorization process through several customizable steps. Each of these steps is chosen from a previous work with the intention of bringing the best part of different existing colorization algorithms together. Through the testing of several different configurations of the framework, we have found that none of our tested configurations outperform existing algorithms in perceptual quality, however there is promise for tuning the framework in a way that will make the algorithm perform better.

5.2 Contributions and Significance

This thesis introduces several novel approaches to the various aspects of automatic image colorization. Specifically, we present:

- A framework that is designed to be highly customizable (and pluggable in some cases). Users of our framework are not constrained to the design decisions used in the original implementation. Each of the four major steps of the algorithm presents an opportunity for customization so that factors outside of those presented in this thesis may be implemented and tested.

- A novel approach to the generation of a color palette to be used in a colorization. This approach is designed to leverage the Euclidean distance property of the $L\alpha\beta$ color space in order to preserve color variance while still capturing the overall color mood of a scene. Although our method is built to be used in further automated parts of the algorithm, the discretization algorithm on its own has a potential manual use for providing a color palette to a PhotoShop user.

- A novel approach to the handling of a scene compositionin in a colorization. This approach fundamentally increases the space of grayscale images that can be reasonably colorized due to the removal of the need for a fully composed color-equivalent context. Instead of looking *across* the space of source images, our approach looks *through* the space in order to avoid diluting the meaning of a feature match between the source and target images.

5.3 Future Work

Even though many of our goals were accomplished with the current implementation of the framework, there are still several items that would be addressed given more time and resources.

- Tune the framework to better compete with the state-of-the-art colorization techniques:

Even though the current state of the framework does not compete with current colorization techniques, we believe that given enough time, a new configuration could yield significantly better results.

- Better formulate the color cost mapping:

  Although the current cost function serves its purpose, the low-cost values tend to appear at a very high rate which can prevent the final colorization from being able to correct for single-pixel anomalies.

- Better formulate the Markov Random Field (MRF):

  The current version of the MRF has two major short-comings:

  1. The library being used in our code only supports the use of the 4-pixel neighborhood in the global coherence term. Ideally, this would be an 8-pixel neighborhood to collect as much surrounding information as possible.
  2. The current MRF formulation provides no forgiveness in the global coherence term to pixel regions where we expect to see a variation in color. This could possibly be represented by using some type of image gradient or similar statistic to detect regions of expected variation.

  Addressing these two issues could potentially improve the overall quality of a final colorization.

Bibliography

[1] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Trans. Graph.*, vol. 21, pp. 277–280, July 2002.

[2] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong, "Image colorization using similar images," in *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pp. 369–378, 2012.

[3] Y. Yang, X. Chu, T. T. Ng, A. Y. S. Chia, J. Yang, H. Jin, and T. S. Huang, "Epitomic image colorization," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 2470–2474, May 2014.

[4] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, pp. 689–694, Aug. 2004.

[5] R. Irony, D. Cohen-Or, and D. Lischinski, "Colorization by example," in *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, pp. 201–210, 2005.

[6] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng, "Intrinsic colorization," *ACM Trans. Graph.*, vol. 27, pp. 152:1–152:9, Dec. 2008.

[7] Charpiat, Guillaume and Hofmann, Matthiasand Schölkopf, Bernhard, *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III*, ch. Automatic Image Colorization Via Multimodal Predictions, pp. 126–139. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[8] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1222–1239, Nov 2001.

[9] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels," tech. rep., 2010.