Template Matching in Image Colorization

Caris Moses
MIT CSAIL
carism@mit.edu

Abstract

Image colorization is the process of adding color, or chrominance, to a gray-scale image. User-guided approaches require a user to provide desired color information to influence the output of the colorization process. Data-driven approaches train on large sets of gray-scale and corresponding colorized images. They tend to result in grayish images and need to be retrained to handle images with new object types. They also do not allow the user to input desired color information. Therefore, to allow the user to influence the resulting colorized image and to produce vibrant and lifelike images, this paper focuses on the user-guided approach. We build on user-guided colorization work by Levin et al. *in which the user draws color patches on top of regions of the gray-scale image with similar intensity values (this typically corresponds to a specific object). One drawback of this approach is that in regions with intricate and highly repetitive patterns, hand specifying color information can be very tedious and time intensive. This paper discuses the development of a template matching algorithm to automatically provide color information to patterned regions of gray-scale images given one template of the pattern. An important step of the algorithm involves searching the space of affine transformations. The algorithm is faster than a brute force approach to template matching and results in realistic colorized images.*

1. Introduction

Image colorization is the process of converting a gray-scale image into one with color. The problem is ill-posed, meaning that there is no way to recover the true original color of a given gray-scale image. However, many methods have been developed to produce colorful images which are realistic and believable to the human eye. Data-drive approached use large amounts of naturally colored images and their gray-scale component to learn a colorization model, then apply that to future gray-scale images with unknown color values. User-guided methods involve a user giving some input to the colorization process. For example, we will discuss at an approach in which the user supplies "scribbles" of color over the gray-scale image. Those colors are then propagated throughout the image to colorize the remaining pixels.

Data-driven approaches are useful in that, once the model is trained, no user input is required to produce a colorized image. However, these trained networks tend to overfit to their training data. For example, if a network is trained on images of places and a user would like to colorize an image of a person, additional training will most likely need to take place. In addition, if a user would like to colorize an image of which they *do* have some ground truth knowledge, it is impossible (as far as I know) to give a network this data and have it effect the outcome colorized image.

User-guided approaches are able to overcome both of these disadvantages of data-driven approaches. The user is able to input ground truth knowledge, as well as colorize images of any type. However, a pitfall of this approach is that it can be time consuming and tedious for users to seed these colorization algorithms with enough information to produce realistic results. This issue is especially true when images have large regions of patterns where the user needs to supply ground truth color information for each portion of the patter. Figure 1 gives an example of a gray-scale image which would be very difficult to provide color information to by hand. If any part of the plaid shirt is missed, the resulting colorized image will not be colored in those regions.

Template matching is the process of finding regions in an image that correspond to a given template regardless of any kind of transformation it may have undergone. In this paper I will discuss a user-guided approach to image colorization, and several approaches I took to automate the user input process in patterned regions through template matching.

2. Related Work

2.1. Data-driven Methods

Iizuka *et al*. [3] trained a convolutional neural network (CNN) which was able to learn per-pixel color values, as well as global and local features of an input image. They note that, "the main limitation of the method lies in the fact

Figure 1. A gray-scale image with a heavily patterned region that would be very time intensive to color by hand using the current user-guided approach developed by Levin *et al*.

that it is data driven and thus will only be able to colorize images that share common properties with those in the training set."

Zhang *et al*. [7] also uses a CNN, however they include semantic information in the training process to improve colorization. They note that a main issue with colorization networks is that the output is an average of many possible colors, resulting in a grayish and desaturated appearance. They improve this by treating colorization as a color classification problem, and are able to produce "visually compelling results."

Figure 2 (right) gives the output of the CNN developed by Iizuka *et al*. on the input image (left). As you can see it was able to successfully colorize the woman's face, however the remainder of the picture does not vary far from the original gray-scale image. This is most likely due to the network begin trained on faces, but not on couches, dresses, polka dot patterns, etc... In addition, if the user wanted the couch to be a specific color, there is no way for them to specify this.

2.2. User-guided Methods

Another method of image colorization involves a user supplying a gray-scale image along with a colorized refer-

Figure 2. The output (right) of the CNN developed by [3] on the input gray-scale image (left).

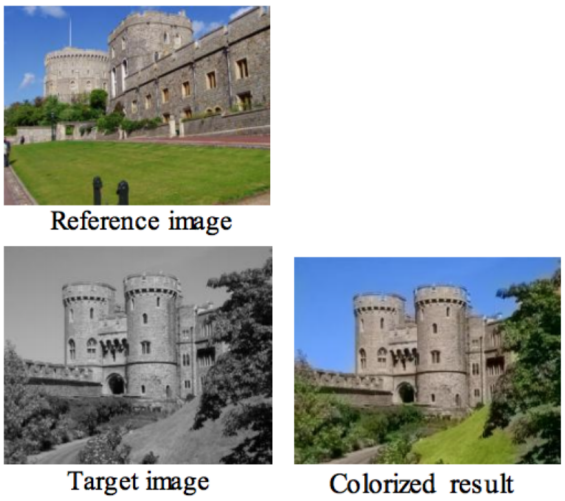

Figure 3. A visual of image colorization using a reference image. As you can see features common to both the reference and target image are colorized as such in the colorized result (Image courtesy of [1]).

ence image [1]. In this method the two supplied images must have similar semantic information. The algorithm works by finding correspondences between the two images and color matching these corresponding super pixels. The authors note that this method gives better results than colorization techniques that work on a per-pixel basis. Figure 3 gives an example of using this method.

Levin *et al*. [6] also takes a user-guided approach to the colorization problem. A user "scribbles" color patches over a gray scale image, and an optimization is performed which propagates the color throughout the image. The optimization uses the fact that regions with similar intensity values should also have similar color values. Figure 4 gives an example of what to supply the optimization problem (left) and the resulting colorized image (right). This method produces very realistic results, but certain images can be very user

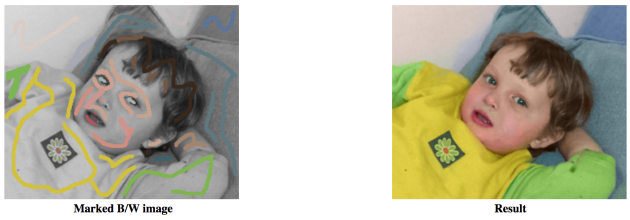

Figure 4. A visual of image colorization using the user-guided approach developed by [6]. The left image is provided by the user, and the right image is the output of the colorization optimization problem (Image courtesy of [6]).

demanding.

A common issue with the colorization optimization problem posed by Levin *et al*. is color bleeding at the edges of objects. This is due to the fact that the chrominance values vary linearly with the input images' intensity values. Huang *et al*. [2] improved upon Levin *et al*.'s work regarding this issue by incorporating an edge detector into the colorization process. First, the user supplies color "scribbles" as seen in Figure 4. Then, object edges are detected in this image. The provided colors and edge information is used to color the edges of the objects in the image. Finally, this image with the original user's "scribbles" as well as the colored edges are put into the colorization optimization algorithm to colorize the rest of the image.

2.3. Template Matching

This work focuses on the problem of template matching. Given a gray-scale template, how can we find image regions matching the template regardless of translational, rotational, scale, and affine transformations?

The simplest and most obvious approach is to perform a brute force search of the transformation space and look for matches. Kim *et al*. [4] proposed the Ciratefi algorithm to improve upon the speed of a brute force algorithm. They claim that their algorithm produces the same results as a brute force approach and executes 400 times faster. The Ciratefi algorithm uses several cascading filters to eliminate non matching pixels over three iterations. The first filter calculates "probable scale factors" and eliminates unlikely pixels based on this criteria. The second filter calculates "probably rotation angles" and eliminates unlikely pixels based on this criteria. Finally, a third filter performs the brute force template matching algorithm on the remaining pixels.

While Kim *et al*.'s approach to template matching is quick and effective, translation, rotation, and scale differences do not cover all of the possible transformations one might encounter in a natural image. Pixels corresponding to a template may be under an affine transformation. Korman *et al*. [5] developed the FasT-Match algorithm to handle these situations. Their approach to template matching under affine transformations works by sampling the affine transformation space. It takes advantage of the fact that the space of affine transformations can be bounded under the assumption that images are smooth, which implies that the measure of similarity between a template and an image will vary smoothly over an image. They take a branch-and-bound approach to sampling the space of affine transformations.

3. Approach

This work on template matching is a combination of approaches taken by Kim *et al*. and Korman *et al*. A user supplies a template, or one section of a pattern, from the gray-scale image. They also supply a colored template (the same template with the desired color "scribbled" over it). An initial pass of a filter is used to eliminate unlikely candidates for template matches. Then, we sample the space of affine transformations to find template matches for the remaining candidates. These transformations are applied to the colored template and overlaid with the original gray-scale image. Finally this image is run through Levin *et al*.'s colorization algorithm to produce a colorized image.

Throughout this paper the similarity function will correspond to the following normalized SAD (sum of absolute differences) function:

$$SAD(T,R) = \frac{\sum_{i=1}^{n} |T(i) - R(i)|}{n} \quad (1)$$

where T is the gray-scale template, R is the region of the gray-scale image currently being tested, and n is the number of pixels in both T an d R. These images are typically converted into a column vector before math operations are performed. The template matching algorithm takes in a gray-scale image which is used to extract the template as well as to find matches to the template. The template matching algorithm is also supplied a user-colored image in which all regions are "scribbled" aside from the patterned region. However, one section of the patterned region *must* be colored (colored template) as it is used to automatically color the rest of the user-colored image after the template matching algorithm. The output of the template matching algorithm will be referred to as the auto-colored image. This is put into the colorization process, and the output of that will be referred to as the colorized image.

First, I took an exhaustive search approach to template matching. I first counted pixels as matching if their similarity to the template was lower than a threshold value (Section 3.1). Then, I improved upon these results by only counting pixels corresponding to the minima of the similarity function (Section 3.2). Finally, I implemented a search of the affine transformation space at the minima of this similarity function (Section 3.3) to find more accurate and realistic

template matches.

3.1. Brute Force Approach with a Threshold (BF-Thresh)

In the brute force approach, the user supplies a range of scale and rotation values to transform the template with. They also supply the desired step size in the translational, rotational, and scale space with which to perform the search. The measure of similarity between the template, T, and the region of the gray-scale, R, is calculated using the SAD function.

At each pixel the minimum over the scale and rotational space is stored. Finally, a user-guided threshold is used to determined what pixels correspond to a match. These matching pixels are overlaid with the correct rotated and scaled colored template to produce the auto-colored image.

3.2. Brute Force Approach with Local Minima (BF-Min)

The approach given in Section 3.1 uses a simple threshold value to determine matching pixels. This results in auto-colored images where many colored templates are placed near regions of matching pixels. This is not the desired auto-colored image. We aim to find a single region that matches the given template, not overlapping groups of regions that are all similar to the template.

Therefor, we use the minima (as well as a threshold) of the SAD function to determine matches rather than a simple cutoff threshold alone. This greatly improved the results.

3.3. Affine Transformation Space Search using Sampling (Affine-Search)

The issue with the approach given in Section 3.2 is that some of the matching templates still were not being detected. This is due to the fact that simply rescaling and rotating the template does not find instances where the region matching the template has undergone an affine transformation.

In addition, the brute force approaches given in Sections 3.1-3.2 do not scale well. By adding in another transformation dimension (affine space) this performance will only worsen.

Therefore, in the final approach two stages are used to speed up the template matching process. First, a course estimate of the SAD function is calculated by passing the untranformed template over the entire gray-scale image with a translational step size one 1. Then, as in Section 3.2, the first set of candidate matches are the minima of the SAD function that are also under a user-specified threshold. Then, a local search at each of these candidates is performed. The local search calculates the SAD function for a predefined set of local homogeneous transformations. If the minimum of these SAD functions is lower than a second user-specified threshold value, then it is stored as a match and the colorized template is transformed to match this found minima.

The BF-Thresh and BF-Min approaches have a runtime of $\mathcal{O}(NRS)$ where N is the number of pixels in the gray-scale image, and R and S are the number of rotation and scale values the search is trying. As you can see this quickly explodes as the user tests more transformations of the image. The Affine-Search approach is able to find template matches in $\mathcal{O}(N)$. The only way that performance would be worse than this would be if $HM > N$. If this is ever the case it is most likely due to user error. H the number of homogeneous transformations used in the search and M is the number of matches found after the first pass of the Affine-Search algorithm. For reference, in our example in Section 4, $N = 706560, H = 10$, and $M = 76$. Clearly $706560 >> 760$ which shows what a great improvement the Affine-Search algorithm is over the BF approached.

4. Experimental Results

All algorithms were tested on the image in Figure 5. This image shows the color image that the user supplies. The program then asks the user to draw a rectangle around the colored template and the region of interest (both shown in green in Figure 5).

4.1. BF-Thresh Results

The results from using BF-Thresh for three different threshold values are given in Figure 6. As you can see, this approach is very sensitive to the threshold value. As the threshold gets larger we get the benefit of more valid matches being detected and incorporated into our final auto-colored image, however there is the negative effect of more matches being found overall. This leads to large patches of red outside of the white polka dots. When the auto-colored image places the colored template outside of the polka dot it results in color-bleeding in the final colorized image.

4.2. BF-Min Results

The results from using BF-Min for three difference threshold values are given in Figure 7. This technique of finding the minima of the similarity function instead of just including all values below the threshold greatly improves the resulting scribbled image. As you can see the resulting auto-colored image is much less sensitive to the threshold. In addition, as the threshold increases more valid matches are added without the preexisting matches growing too large and spreading beyond the boundary of the polka dot. As you can see there is less color-bleeding with this method than with the BF-Thresh method.

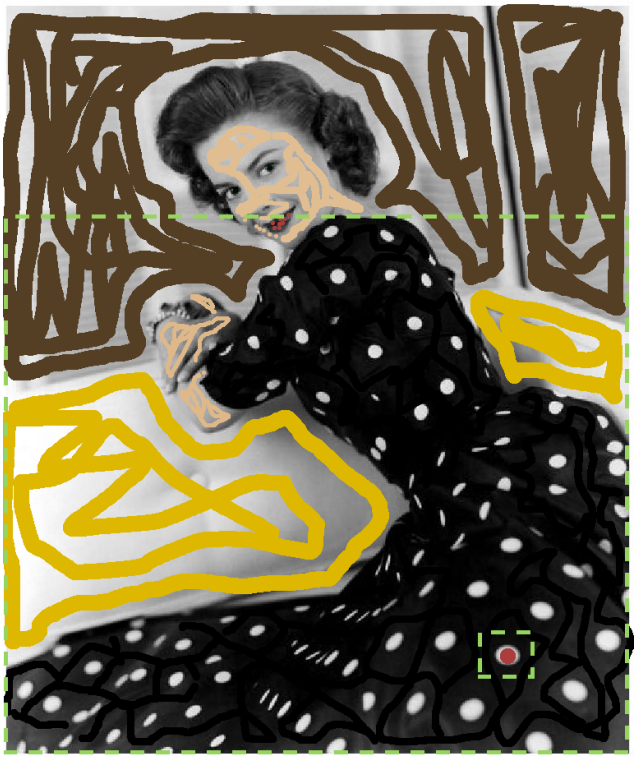

Figure 5. The input to the template matching algorithm. The user is prompted to outline the colored template and the region of interest with rectangles.

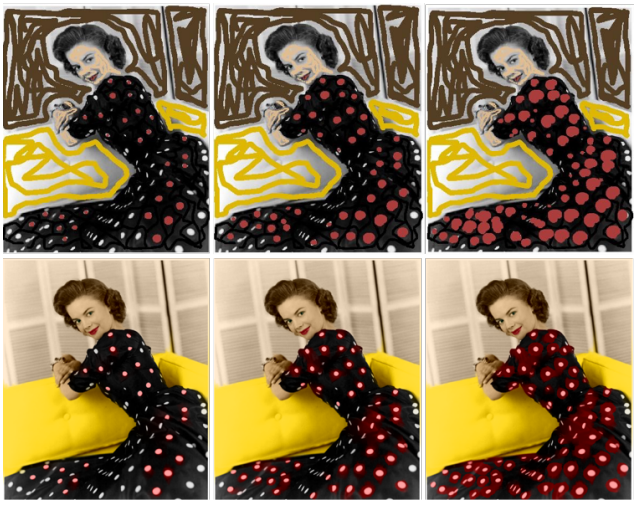

Figure 6. Resulting auto-colored (top) and colorized (bottom) images using the BF-Thresh algorithm. For left to right the threshold values on the similarity function used are 0.1, 0.15, and 0.2.

4.3. Affine-Search Results

Figure 8 shows the results for the Affine-Search algorithm. As you can see the results are a big improvement

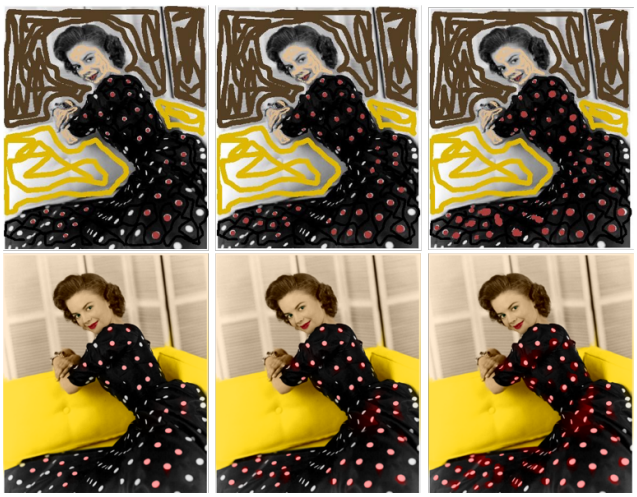

Figure 7. Resulting auto-colored (top) and colorized (bottom) images using the BF-Min algorithm. For left to right the threshold values on the similarity function used are 0.1, 0.15, and 0.2. The minima of the similarity function are restricted to being less than these threshold values.

Figure 8. Resulting auto-colored (left) and colorized (right) images using the Affine-Search algorithm. The threshold used to determine initial template matching candidates was 0.25. The threshold used to determine a final affine transformed matching template was 0.15.

over the brute force approaches. It is able to detect more of the polka dots which are nearly perpendicular to the camera.

Figure 9 shows the result of the colorization optimization on an image colored by hand. Table 1 gives the time it takes for the algorithms (or user) to perform the task of coloring in all of the polka dots. As you can see the affine transformation technique is vastly faster than the brute force approaches as well as the coloring by-hand approach. The Affine-Search technique is not perfect, but I am confident that improvements would still result in an algorithm much faster than a by-hand approach to image coloring. I will

Figure 9. A gray-scale image colored by hand (left) and the resulting colorized image (right).

discuss possible improvements in Section 5.

Method	Time (min)
BF-Thresh	60
BF-Min	60
Affine Search	1
By hand	5

Table 1. Times to color in the image given in Figure 5 using different approaches.

5. Conclusion

The final results of the Affine-Search algorithm are promising, but still lacking. For one, there is still a bit of color-bleeding due to the fact that the colored templates are not being perfectly aligned to the underlying gray-scale image. In this approach a discrete set of homogeneous transformations were tried on the underlying image. A more intelligent search of this transformation space (perhaps similar to Korman *et al.*'s approach) could be used to ensure that the final colored template is correctly overlaid with the image. In the final image many of the polka dots which were near perpendicular to the camera were missed. Again, a better search of the affine transformation space would improve this. Another improvement would be to incorporate a more sophisticated similarity function. The SAD function is a very simple calculation of the similarity between a template and the underlying image. However a new function could be used that takes into account contrast and brightness differences between the template and the underlying image.

In this work I was able to show promising preliminary results in template matching for image colorization. This process enables users to quickly bring gray-scale images to life with vibrant colors of their choosing.

My code along with the colorization code from Levin *et al.* can be found at `https://github.mit.edu/carism/colorization.git`.

References

[1] 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*, pages 369–378. ACM, 2012.

[2] Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu. An adaptive edge detection based colorization algorithm and its applications. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 351–354. ACM, 2005.

[3] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016.

[4] H. Y. Kim and S. A. De Araújo. Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast. In *Pacific-Rim Symposium on Image and Video Technology*, pages 100–113. Springer, 2007.

[5] S. Korman, D. Reichman, G. Tsur, and S. Avidan. Fast-match: Fast affine template matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2331–2338, 2013.

[6] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM transactions on graphics (tog)*, volume 23, pages 689–694. ACM, 2004.

[7] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.