

# Image Colorization Using Scribble

Kimberly Chen





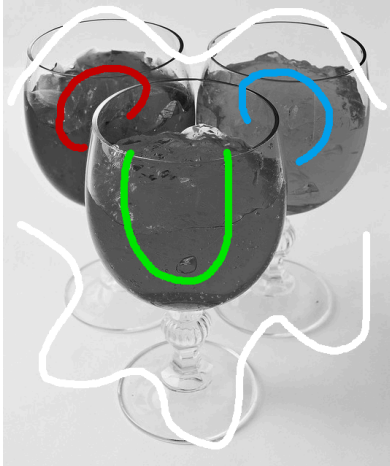

Anson Thai

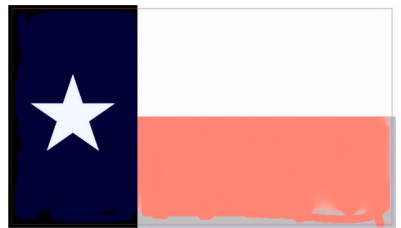
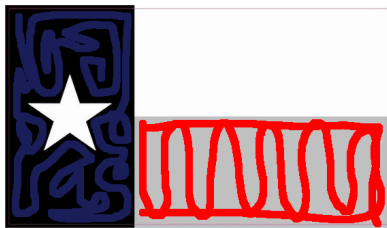
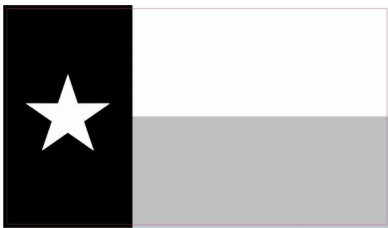
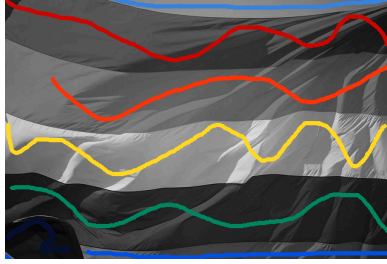
CSCE 448 Computational Photography

Spring 2025

Our project implements an automatic image colorization algorithm based on user-provided scribbles, following the method from Levin et al. (SIGGRAPH 2004). A grayscale image and its corresponding colored scribbles are used to propagate color across the image using an edge-preserving smoothness constraint.

## Results:

Gray Scale Image	Scribble Image	Results
		
		



## Discussion:

At its core, the algorithm assumes that similar grayscale values should have similar colors. We first normalize the grayscale image and construct a Laplacian matrix based on the intensity variance within a local window. The scribble pixels drawn by the user on the image are embedded into a sparse linear system as hard constraints. Then system solves the U and V chrominance channels in the YUV color space separately. Once solved, the obtained chrominance values are combined with the luminance channel (Y) in the original grayscale image to generate the final funny colored output image.

Before starting the colorization process, we first load a grayscale image and the corresponding color scribble image. The grayscale image contains only luminance information (uniform RGB values), while the scribble image contains the “color cues” provided by the user. In order to identify which pixels contain color information, we compare the scribble image with the grayscale image converted to a three-channel format. Any pixel that does not have equal RGB values is identified as a "scribble pixel". This process is critical because these pixels will serve as the fundamental constraints for color propagation. Next, we construct a sparse Laplacian matrix to model the similarity between neighboring pixels based on intensity values. The matrix assumes that pixels with similar grayscale values should also have similar colors. Thus, we define a local window around each pixel (e.g., radius  $d=1$ , i.e., a  $3\times 3$  neighborhood) in which the variance of intensities and the weights between pixels are calculated. The weights are calculated based on a quasi-Gaussian function that decays rapidly as the square of the intensity difference between pixels increases, ensuring that colors are more easily propagated in areas with similar shadows or textures.

To avoid division by zero in areas with almost constant intensity, we add a small regularization term ( $REG\_EPS = 1 \times 10^{-6}$ ) when calculating the variance. For scribble pixels, we directly set their color values as known quantities on the right side of the equation when constructing the linear system, which acts as a hard constraint. For other pixels, smoothness constraints are added based on the similarity between their neighborhood pixels to construct a complete sparse matrix. Finally, we use `spsolve` to solve the two chromaticity channels (U and V) separately. The resulting chromaticity map is combined with the original grayscale map (i.e., the Y channel) to reconstruct a YUV color image, which is then converted to BGR format for display and storage. This method ultimately enables smooth and coherent propagation of colors from sparse user scribbles while effectively maintaining the integrity of image edges and structures by virtue of brightness similarity.

## References

Levin, Anat, Dani Lischinski, and Yair Weiss. "Colorization Using Optimization." ACM Transactions on Graphics (TOG), vol. 23, no. 3, 2004, pp. 689–694.  
<https://webee.technion.ac.il/people/anat.levin/papers/colorization-siggraph04.pdf>.