How Can a Grayscale Image be Pseudocolored based on a Similar True Color Image?

Zoya Shafique
Final Project Report
Digital Image Processing

Abstract

Pseudocoloring is widely used to not only enhance the visual appeal of images but also to highlight important details. As such, it plays an important part in data visualization in many fields. In this paper, we propose a method for pseudocoloring images based on the generation of a color map from a true color image with similar objects present as the grayscale image we would like to color. We first use k-means clustering to segment regions of the true color image based on their color and then create a color map for each region. Then, each map is applied to the corresponding region of the gray scale image using edge detection. For more accurate segmentation, we introduce a feature vector containing texture, spatial, and intensity information. Our method was successful in transferring colors from a specified true color image to a grayscale image. We also discuss an attempt at a region growing algorithm for segmentation.

Introduction

A technique for mapping gray level intensities to red, green, and blue values either through a table or function, pseudocoloring is an important aspect of data visualization in many fields [1]. For instance, grayscale images taken by the Hubble Space telescope are pseudocolored to represent distributions of different gases and compounds [2]. Current methods for pseudocoloring involve applying color maps [3], which are arrays of colors indexed to correspond to specific intensity levels, or color matching between two images [4]. Pseudocoloring techniques in the frequency domain have also been discussed [5, 6].

Although various methods exist for image coloration based on existing images [7], they involve complex matching algorithms. In this paper, we propose a method for image coloration based on an existing, similar image that relies on segmentation. Our algorithm does not color match, but relies on manual user input to segment a true color image to create color maps and then apply them to a grayscale image. We create color maps from true color images by selecting values indicative of the range of colors in each red, green, and blue (RGB) channel of the true color image and then interpolate to create a color map based on the image. We then employ k-means clustering to segment our image and create color maps for each cluster.

1. Initial Qualitative Analysis of Pre/post processing techniques on Pseudocolored Images

Pseudocoloring techniques map gray level intensities to specific values of red, green, and blue to create color images. In this section, we explore histogram equalization and various methods for creating grayscale images and their impact on pseudocolored images.

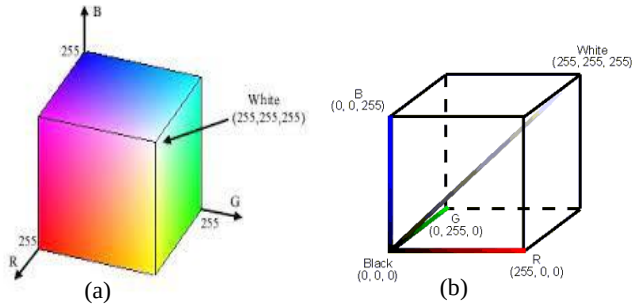
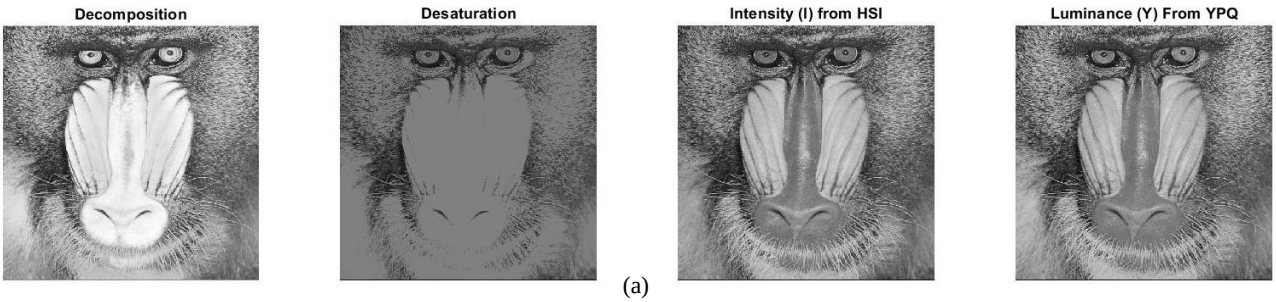


Figure 1: (a) The RGB color space [8]. Different colors on the cube represent combinations of red, green, and blue. The white vertex is diagonally across from the black vertex and marks the points where R, G and B are all equal. (b) [9] The line from the black vertex (0, 0, 0) to the white vertex (255, 255, 255) in the RGB color space represents the grayscale values. In desaturation, the RGB values are forced to move to this line.

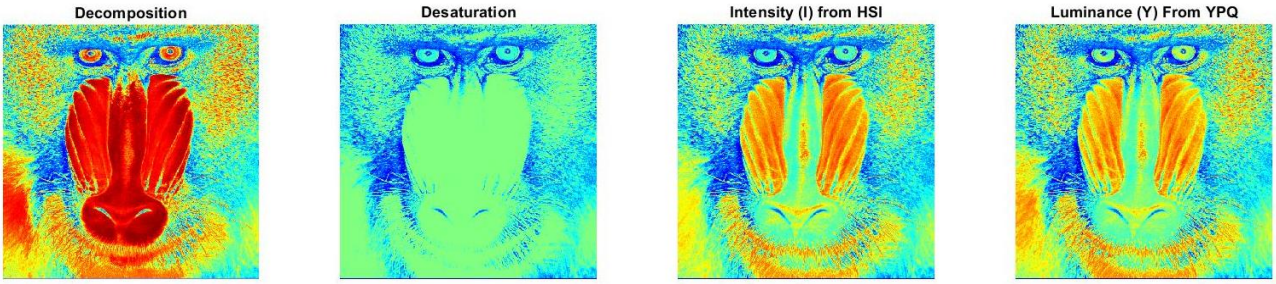
1.1 Qualitative Comparison of True Color to Grayscale Conversion Methods

We compared four grayscale conversion methods: desaturation, decomposition, intensity in an HSI color space, and luminance.

Desaturation [10]: Desaturation works in the hue, saturation, intensity (HSI) color space by forcing the saturation to zero. The equivalent method in the RGB



(a)



(b)

Figure 2: (a) The true color baboon image was converted to grayscale using: decomposition, desaturation, intensity from the HSI color space, and luminance. Decomposition preserves many of the details of the original image and presents a clear and well contrasted image. Using desaturation to convert the true color image into grayscale resulted in the loss of many details, such as the ridges on the baboon's nose. Using the intensity from the HSI color space and the luminance from the YPQ color space resulted in similar images. (b) Pseudocolored images. Although the grayscale decomposition image was appealing and preserved most of the details of the original image, applying a color map to the decomposed image results in an image in which some of the nose ridges and whiskers are difficult to see. The intensity and luminance grayscale images had similar results once the colors map was applied.

color space moves the point from the RGB cube surface to the grayscale line as shown in Fig. 1. This is done by averaging the maximum and minimum RGB values as follows:

$$Gray = 0.5(Max(R,G,B) + Min(R,G,B)). \quad (1)$$

Decomposition [10]*:* Decomposition takes each pixel and forces it either to its maximum or minimum value without regard to the color channel. In other words, each pixel is replaced by either the maximum or minimum value across all three color channels.

Intensity [11]: The HSI color space represents the hue, saturation, and intensity of an image and is more useful for describing colors than the RGB color space. The hue, H, is defined as the pureness of a color (i.e. pure orange), the saturation, S, is the amount a pure color has been diluted by white light, and lastly, the intensity, I, is the gray level of the image. For the purposes of converting a true color image into grayscale, we focus on the intensity. For normalized, R, G, and B values the intensity is given as:

$$I = \frac{1}{3}(R + G + B). \quad (2)$$

Luminance: The YPQ color space is defined in [12] as analogous to the HSI color space. Both define colors in a manner similar to human visual processing. For instance, describing a flower in terms of its RGB triplet i.e. [0 50 125] is obscure. However, describing the color in terms of its lightness, and shades of color is more accessible and easy to understand. The YPQ color space thus defines the luminance, or lightness (Y), the yellow-blue levels (P), and the red-green levels (Q) of an image. The conversion between the RGB and YPQ spaces is given as:

$$\begin{bmatrix} Y_i \\ P_i \\ Q_i \end{bmatrix} = \begin{bmatrix} 0.2989 & 0.5870 & 0.1140 \\ 0.5000 & 0.5000 & -1.000 \\ 1.0000 & -1.0000 & 0.0000 \end{bmatrix} \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}. \quad (3)$$

The results of our conversions are given in Fig. 2(a). After converting true color images into grayscale images through the aforementioned methods, we applied the in-built jet color map to compare the effect of the different methods on the pseudocolored image as shown in Fig. 2(b).

1.2 Effect of Histogram Equalization on Pseudo-Colored Images

We also compared the effect of histogram equalization on pseudocolored images by comparing pseudocolored images with no contrast enhancement, pseudocolored images that were colored after contrast enhancement and pseudocolored images that were equalized after coloring. To equalize the histogram of the colored images, we

enhanced the contrast of each true color channel. Our results are shown in Fig 3.

Figure 3: From left to right: the winter color map applied to an image with no histogram equalization, the winter color map applied to a histogram equalized image, the winter color map applied to an image which was equalized after the application of a color map. The pseudocolored image without any equalization presents no artifacts and preserves all of the information in the original grayscale image.

Fig. 3 shows the winter color map applied to an image. Without any histogram equalization (the image all the way to the left of Fig 3), the image is free of any artifacts and the details (the boat, birds, and plants in the right corner) are clearly visible. When the grayscale image is equalized before the application of the color map, the resulting pseudocolored image has a pixelated sky and some details are lost (the boat, birds, and landscape in the back are all one color causing the details to blend into each other where they overlap). The image on the right of Fig. 3 depicts a pseudocolored image equalized after the application of the color map. This image is equalized by equalizing each, R, G, and B channel separately and then recombining them. Equalizing the histogram of the pseudocolored image also produces an image that is pixelated and in which some details are hard to make out.

The other color maps we tested on various images (see supplemental material) showed similar results when their histograms were equalized before and after pseudocoloring. It is possible that the pixelation occurs because, in the original image, the sky is a smooth and uniform region. Equalizing the histogram results in a contrast enhancement, altering the intensity values of pixels in the sky and thus leading to the pixelation seen in the middle and right images in Fig. 3.

2. Creating Color Maps from a True Color Image

After concluding that the grayscale conversion method and histogram equalization produced subjective results depending on the image, we decided to use the MATLab inbuilt grayscale conversion method (rgb2gray) and forego contrast enhancement for our pseudocoloring algorithm. Next, we created a color map from an input reference image and used it to colorize grayscale images.

2.1 The Algorithm

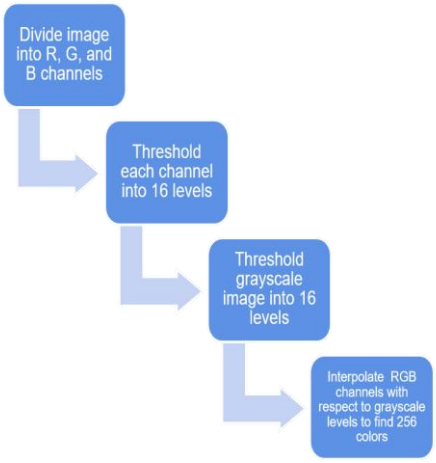


Figure 4: Our algorithm for creating color maps based on an input image. Sampling 16 points from each channel of the reference image and interpolating to 255 colors for each r, g, and b channel results in a color map based on the colors found in the image.

Fig. 4 depicts the algorithm for creating color maps based on a true color input image. We first separated the true color image into its R. G, and B channels and thresholded each channel to 16 values. In other words, we divided the intensities into 16 values that represented the entire range of intensities from 0 to 255. After thresholding each channel, we thresholded the grayscale image that we wanted to color into 16 levels as well. With the thresholds set, we interpolated the R, G, and B, channels from 16 values to 256 values with respect to the grayscale image intensities. The channels were interpolated with respect to the image we wished to color instead of the grayscale version of the input true color image because the corresponding intensities for the generated color map would directly correspond to the intensities in the image we wished to color. In general, our algorithm worked for a variety of colored images. In some cases, however, we had to adjust the levels at which we thresholded the image.

2.2 Results

One of our generated color maps is shown in Fig. 5. Although we were able to successfully generate an accurate color map from the input image, we found that for images with multiple colors rather than hues of one color, our method did not generate accurate color maps. Instead of obtaining a color map which defined the range of colors present in the image, the generated color maps were all grayscale as shown in Fig. 6.

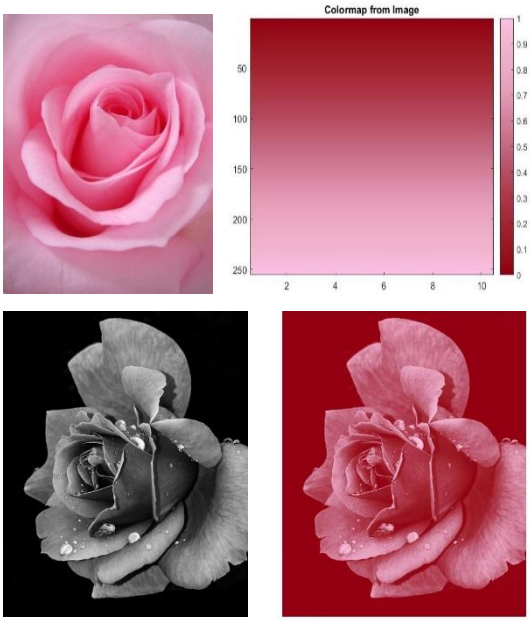


Figure 5: Using our method, we were able to successfully generate an accurate color map (top right) based on the true color image of a pink rose. The color map was effective in coloring a grayscale rose.

After unsuccessfully attempting to resolve the issue, we realized that our algorithm lacked a matching component. In other words, our method did not take into consideration that certain regions were a certain color (i.e. that the sky was blue and the grass was green) but rather processed an input image in its entirety. Considering the image as a whole resulted in a gray scale image as the distribution of colors across the channels was uniform. Thus, thresholding and interpolating the entire image resulted in overlapping values which summed to gray color levels.

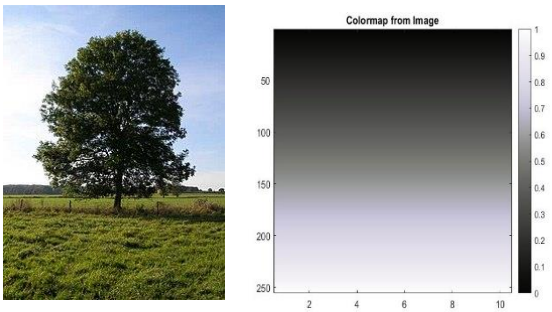


Figure 6: The result of generating a color map from a multicolored image. When we used a multicolor image to generate a color map, the produced color map was in grayscale. This occurred as our algorithm had no way of differentiating between regions of different colors and when all of the colors where considered

3. Segmenting Images for Coloring Using K-Means Clustering, Edge Detection, and Connectivity

To tackle the problem of matching, we turned to image segmentation. Using a true color image similar to the image we wished to color, we converted the true color image into the l*a*b* color space and segmented the resulting image based on color using k-means clustering. After segmentation, each region was turned into a binary mask and used to extract regions of color from the original true color image. With each region separated, a color map was generated for each region and then manually applied to corresponding regions of the grayscale image. Regions of the grayscale image were generally divided into the background and foreground, which were determined using edge detection. Our method is shown in Fig. 7.

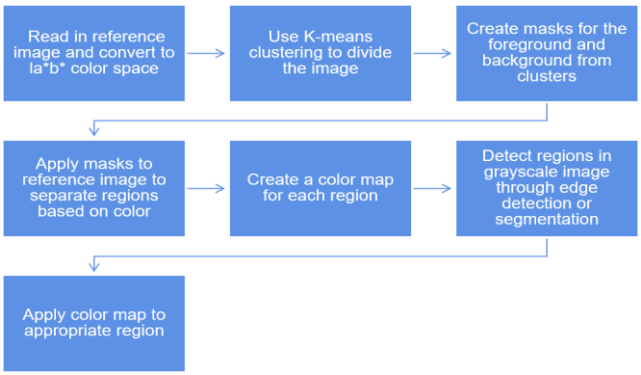


Figure 7: Our algorithm for generating multiple color maps based on different objects in a true color image. Using k-means clustering, we divide the reference image into different regions based on color and then convert each cluster to a binary mask. This mask is eroded depending on the amount of error in the cluster (i.e. if the cluster was assigned parts of the image that were a different color than the desired color) and applied to the original reference image to obtain accurate regions for each color in the true color image. Color maps were then generated for each region and applied to corresponding regions of the grayscale image using segmentation and edge detection.

3.2 L*a*b* Color Space

The CIELAB color space, also referred to as L*a*b* represents a perceptually uniform color space, or one in which changes in the value of a color are proportional to the distance between the two colors in the color space [11]. Based on the color-opponent theory, which states that a color cannot be red and green or blue and yellow at the same time as one opponent color suppresses the other, the L*a*b* color space defines the lightness, L*, of a color, the red-green chromaticity, a*, and the blue-yellow chromaticity, b* [14].

We used the L*a*b* color space for our segmentation method due to the color space's sensitivity to slight changes in color. Our goal was to separate regions of

either similar colors or the same color, making the L*a*b* color space ideal for processing.

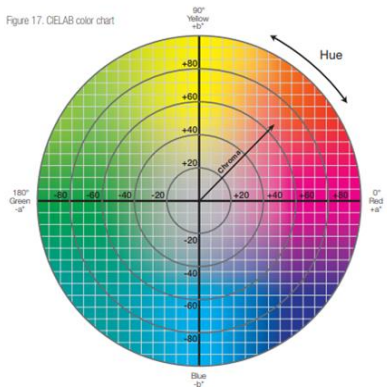


Figure 8 [13]: The l*a*b* color space represents the lightness (l), amount of red or green (a*), and amount of yellow or blue (b*) in a color. This color space is sensitive to slight changes in color, making it an ideal color space for clustering.

3.3 K-means Clustering

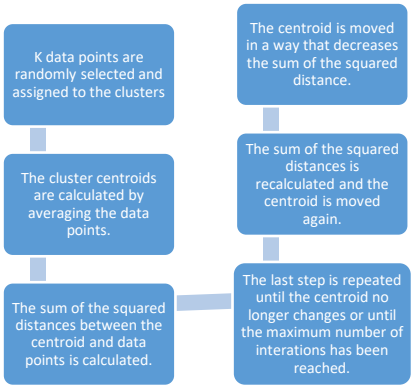


Figure 9: The algorithm for k-means clustering. This method assumes that the number of clusters, k, is already known and iteratively calculates the clusters by minimizing the sum of the squared distance between all of the data points in the cluster and the centroid [17].

K-means clustering is a way of solving the clustering problem, which consists figuring out how to group data points from a given dataset in a meaningful way [15, 16]. In other words, the clustering problem asks how data points can be grouped together so that data with similar characteristics are placed in the same group, or cluster. One answer to this problem is k-means clustering, a method which assumes that the number of clusters, k, is already known. In k-mean clustering, a set of data points is randomly assigned to a cluster with a calculated centroid. The algorithm calculates the sum of the squared distance between the data points and the centroid until the sum is minimized. By minimizing the sum of the squared distance between the data points and the centroid, k-means clustering ensures that the resulting clusters consist of points with similar characteristics. The general algorithm for k-means clustering is presented in Fig. 9.

We chose k-means clustering as the way to segment our image as we wanted to segment our image based on color. Clustering allowed us to group together all of the pixels of similar or the same color and thus proved effective for segmentation.

After segmenting our true color images, we created binary masks for each region and applied them to the original image to extract the true color region. Then, color maps were created for each region using the method described in Section 2.1. The results of our method are given in the next section.

Figure 10: We successfully generated accurate color maps for the purple rose and leaves shown in the true color image. Then, through edge detection, we detected the rose in the grayscale image shown and applied the foreground color map. The leaves color map was applied to the background. Although the background color is not vibrant or entirely visible, a difference in the background between the grayscale image and the pseudocolored image can be seen.

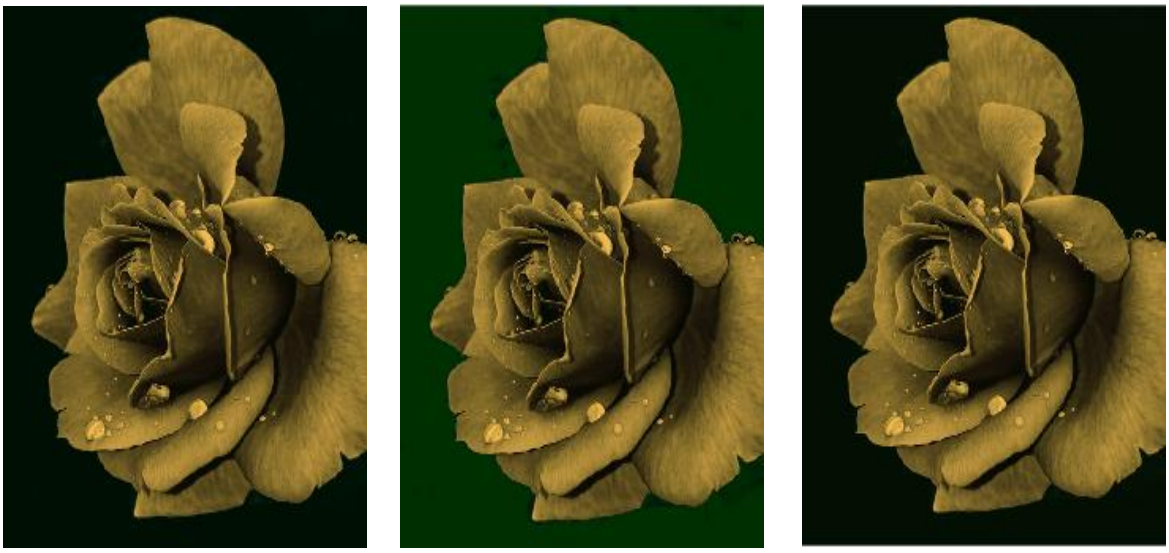


Figure 11: Pseudocolored images generated from the same reference image but with varying erosion strengths for the background. The erosion was set to 1, 5, and 10, respectively.

3.4 Results

Using a purple rose with a leafy green background as our true color input image, we colored a grayscale picture of a rose as shown in Fig. 10 by segmenting the true color image into a foreground (the purple rose) and a background (the leafs).

Although we were able to successfully generate color maps from the true color image, applying the leaf color map to the background of the grayscale image did not produce the level of color we had been hoping for. To determine how to produce a vibrantly colored background, we experimented with various parameters in our algorithm. We found that the amount of erosion applied to our binary masks greatly impacted the final pseudocolored imaged. Fig. 11 shows pseudocolored images with the structuring element for erosion set to 1, 5, and 10 respectively.

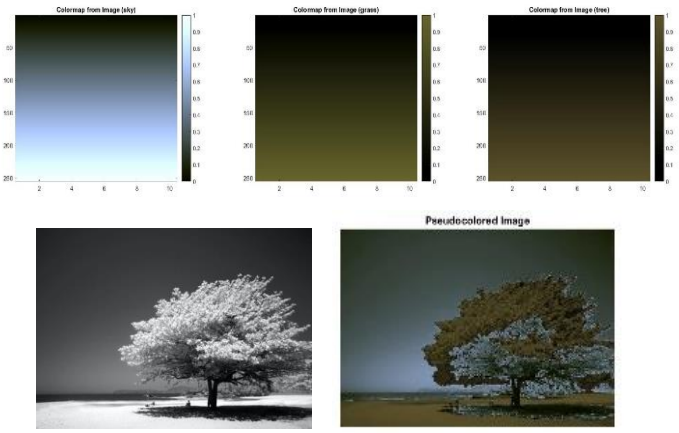


Figure 12: color maps generated from the true color image in Fig. 6. Although we were able to successfully generate accurate color maps for the different regions of the reference image, our edge detection algorithm did not accurately detect the tree in the grayscale image. Furthermore, the edge detection algorithm did not differentiate between the grass and the tree.

The image in the middle has a much more vibrant background than either image on its side, indicating that the erosion of the binary mask – which we perform to clean up any residual part of the image that does not fit in with the rest of the cluster – greatly impacts the quality of the final pseudocolored image. We also found that the level of erosion required to achieve the desired effect varied from image to image and that there was no optimal value across various images.

We used our segmentation algorithm to generate color maps for the true color image shown in Fig. 6. We created three separate color maps: one for the sky, one for the grass, and one for the tree as shown in Fig. 12. After applying the respective color maps to each region, we produced the final pseudocolored image shown. As can be seen from Fig. 12, the final pseudocolored image has lost many details of the tree in the original grayscale image. Furthermore, our edge detection algorithm did not accurately detect the tree and was not able to separate the grass from the tree.

To try to improve the quality of the pseudocolored image, we attempted to segment the tree and grass instead of using edge detection to find the regions to color. As shown in Fig. 13, our segmentation algorithm did not perform significantly better than our edge detection method in terms of detecting the tree. We were, however, able to successfully separate the grass from the tree and color the two separately.

Figure 14: Pseudocolored images with mask erosion strengths of 1, 2, and 5, respectively. As the erosion increased, the intensity of the pseudocolored image decreased. We concluded that the choice of erosion is subjective ad that for each region, the erosion should be treated differently

We experimented with the erosion of our binary maps in an attempt to produce a more vibrant sky color. Fig. 14 shows our results and compares an erosion of 1, 2, and 5. In general, lowering the structuring element strength for erosion made the image lighter and increasing the strength made the image darker. We propose that, based on the desired output, the erosion of each binary image for each region (i.e. the tree, the grass, the sky) be treated separately and optimized individually to produce the most contrasted and clear pseudocolored image.

Figure 13: In an attempt to improve our pseudocoloring algorithm we added labels based on 8 connectivity to our edge detection segmentation. On the left, we have the original pseudocolored image using edge detection. On the right is the image pseudocolored using segmentation to detect various regions. Although we were able to successfully separate the grass and tree using segmentation, our algorithm still failed to accurately detect the tree.

4. Improving Segmentation Results

To improve on the results of the previous section, we implemented a region growing algorithm and a k-means segmentation based on a feature set [18]. Our region growing algorithm was not successful in effectively segmenting the image into different regions (i.e. the foreground, background, object 1, object 2, etc.) but using a feature vector which defined the intensity information, neighborhood texture information, and spatial information about each pixel resulted in a more effective clustering that was able to successfully segment the tree shown in Fig. 12.

4.1 Region Growing Algorithm and Results

Figure 15: In an attempt to improve our segmentation of the grayscale images we wanted to pseudo-color, we implemented a region growth algorithm based on the variance of a 3 x 3 pixel neighborhood. However, our algorithm was unsuccessful in accurately detecting objects based on an initial seed pixel.

Our region growing algorithm asks the user to pick one point on the image as the initial seed point. Based on the seed point, the algorithm uses a 3 x 3 kernel to calculate the neighborhood of the seed and calculates both the weighted average and variance of the neighborhood. The variance is then used as the threshold with which the region growing algorithm compares other pixels to the seed pixel. Our result is given as follows

$$Segmented\ Image = \begin{cases} I(x,y), & SS < T \ \wedge I(x,y)\ is\ 8\ -conn\ with\ I(m,n) \\ 0, & SS > T \end{cases} \quad (4)$$

Where *I(x,y)* is the value of the input image at *(x,y)*, *I(m,n)* is the seed point, *SS* is the threshold calculated at *(x,y)* and *T* is the threshold calculated at the seed. Note that the segmented image is only set equal to the original pixel if the sum of squares at that point is less than the threshold

and if the point is 8 connected to the seed point. If a pixel is 8-connected and its threshold value is equal to or below the seed threshold, the pixel is appended to the seeded pixel. If the threshold value is greater, its corresponding position in the segmented image is set to zero. We define *SS* as

$$SS = \sum_{1}^{9} (I(x_i, y_i) - avg)^2 \quad (5)$$

Where *avg* is the weighted average of the pixel neighborhood given as

$$\frac{1}{16} * \Sigma \begin{bmatrix} I(x-1,y-1) & 2*I(x,y-1) & I(x+1,y-1) \\ 2*I(x-1,y) & 4*I(x,y) & 2*I(x+1,y) \\ I(x-1,y+1) & 2*I(x,y+1) & I(x+1,y+1) \end{bmatrix} \quad (6)$$

T is *SS* calculated at the seed pixel. As shown in Fig. 15, our method did not successfully segment the objects from the input images.

4.2 K-Means Segmentation Using a Feature Vector

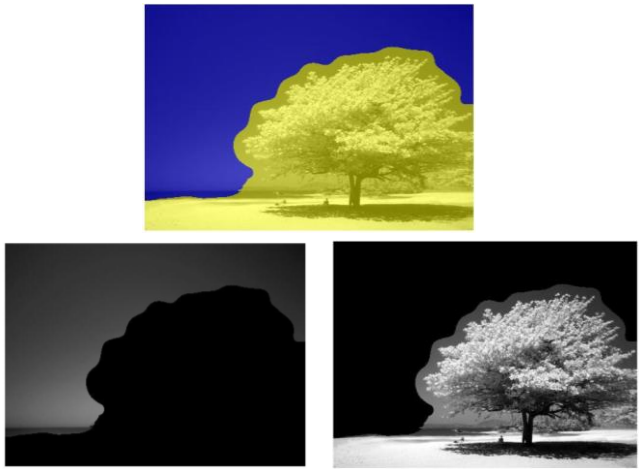


Figure 16: Using texture, spatial, and intensity information we were able to use k-means clustering to more accurately detect objects in the grayscale images we wished to color. Despite the improved accuracy, the segmentation is not exact as the segmented tree image contains some regions of the sky.

The k-means clustering algorithm we discussed in section 3 can be improved upon by using texture and spatial information to calculate the clusters. In section 3, the clusters were calculated based on color information. However, the resulting clusters (see supplemental material) were not entirely accurate and had to be filtered to obtain the desired regions. By including texture and spatial information about the pixels, the k-means algorithm has additional information which it can use to narrow down the results of the cluster and increase accuracy. For instance, a portion of the tree in the grayscale image in Fig. 12 has similar intensity/color values as the sky, resulting in an inaccurate segmentation. In contrast, the two objects have different textures. Therefore, by including texture information in our k-means segmentation, we can more accurately segment our image as shown in Fig. 16.

The texture information was obtained using a Gabor filter, which is an orientation-sensitive bandpass filter [19, 20]. Essentially, a Gabor filter is a sinusoidal signal of some frequency *f* and a particular orientation modulated by a Gaussian wave and is given as

$$g_{mn}(x) = \frac{1}{2\pi a_n b_n} e^{-\frac{1}{2}x^T A_{mn} x} e^{jk_{0mn}^T x} \quad (7)$$

where A_{mn} determines the orientation and frequency range, or bandwidth, of the filter and k_0 is the modulation frequency vector [21]. Once the texture information is obtained through the application of Gabor filters, the pixel positions of each pixel in the image are extracted. Finally, the intensity, texture, and spatial information are all put into one array which we will call the feature vector. This feature vector was then used as the input for the k-means clustering algorithm.

Although using a feature vector led to a more accurate segmentation compared the edge detection and labelling methods described in Section 3, the tree and grass were segmented as one object. To separate the two, we manually selected four pixels from the grass region to form a rectangular mask. Applying this mask to the tree cluster allowed us to separate the grass and tree. Our final pseudocolored image is presented in Fig. 18.

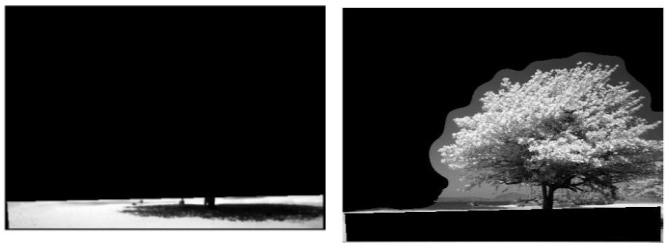


Figure 17: Separated grass and tree objects. Our k-means clustering algorithm segmented the tree and grass as one object whereas our goal was to segment the two separately. To extract the grass region from the tree + grass cluster, we manually selected pixels from the grass region to form a rectangular mask. This mask was then applied to the cluster to extract the grass.

Figure 18: Our final pseudocolored image

Conclusion

We developed an algorithm to create color maps for different objects in a true color image. We then used the color maps to pseudocolor similar objects in grayscale images. We chose to forgo histogram equalization as our initial analysis of the impact of histogram equalization on pseudocoloring showed that the results were subjective and varied from image to image.

Our method achieved mixed results. For solid objects without any holes or gaps, such as a rose, our method was very effective in both generating color maps and applying them to the target image. On the other hand, for objects with less connectivity/starker changes in intensity in the grayscale image, our algorithm failed to accurately detect the target object. We then proposed a modified version of our method and used k-means segmentation with texture, spatial, and intensity information to segment the image we wished to color. Although this method was a great improvement on our original method and the segmentation accuracy significantly improved, there was still reasonable error in our segmentation. We propose that an algorithm that uses matching or learning to detect and label objects in an image would be more effective and accurate at detecting the target object and applying the color map.

For future research we wish to incorporate a matching algorithm in place of using either edge detection or segmentation. By doing so, we can automate the pseudocoloring process and decrease user involvement. Currently, we must manually change the threshold level, the structuring element value, the number of clusters, and must manually segment the grass in section 4 after the initial k-means segmentation. Producing an algorithm which can guess, test, and optimize such parameters on its own would lead to a much more efficient process.

References

[1] S. E. .Umbaugh. Digital Image Processing and Analysis. 2nd ed. CRC Press. 2010.

[2] J. Orwig. Iconic Hubble images are actually black-and-white. Insider. 2015.

[3] Selvapriya B., Raghu B. “A color map for pseudo color image processing of medical images,” International Journal of Engineering and Technology 7(3.34), pp 954-958, 2018.

[4] M. Grundland. N. A. Dodgson. Color histogram specification by Histogram Warping. Society of Photo-Optical Instrumentation Engineers, 5667 (2005).

[5] K. A. Navas, P. R. Naveen, G. Thottan, R. Jayadevan and S. Assim, "A novel frequency domain approach to automated pseudo-colouring of images," 2011 IEEE Recent Advances in Intelligent Computational Systems, 2011, pp. 533-536, doi: 10.1109/RAICS.2011.6069369.

[6] J. Afruz, V. Wilson, S. E. Umbaugh. “Frequency domain pseudo-color to enhance ultrasound images,” 2010 CCSE Computer and Information Science 3(4).

[7] R. Jayadevan, K. A. Navas, A. Ananthan, K. N. Latha. “A review on recent pseudo-coloring techniques,” IJSTE 1(11), pp 334 – 348, 2015.

[8] Image Processing Toolbox User’s Guide. Mathworks.

[9] F. Karstens. What is the RGB color space? Baslerweb.com.

[10] T. Helland. Seven grayscale converstion algorithms. Tannerhelland.com. 2011.

[11] R. C. Gonzalez, R. E. Woods. “Digital Image Processing,” 3rd ed. pp 407 - 414, 2008.

[12] M. Grundland, N. A. Dodgson. The decolorize algorithm for contrast enhancing, color to grayscale conversion. 2005.

[13] T. Mouw. LAB Color Values. X-Rite. 2018.

[14] Identifying Color Differences Using L*a*b* or L*C*H* Coordinates. Konica Minolta.

[15] A. Likas. N. Vlassis. J. J. Verbeek. The global k-means clustering algorithm. [Technical Report] IAS-UVA-01-02, 2001, pp.12. ffinria-00321515f.

[16] D. T. Pham. S. S. Dimov. C. D. Nguyen. Selection of K in K-means clustering. J. Mechanical Engineering Science, 219:103-119, 2004.

[17] ML – Clustering K-Means Algorithm. Tutorialspoint.

[18] imsegkmeans. Mathworks Documntation. (2021)

[19] Aach, T.; Kaup, A.; Mester, R.: ,On texture analysis: Local energy transforms versus quadrature filters." In Signal Processing, vol. 45, pp. 173-181, 1995

[20] P. Joshi. “Understanding Gabor Filters”. Perpetual Engima (2014).

[21] Gabor Filters, The Computer Vision Lab at GET, University of Paderborn (1998).