CSC 249 Homework 3 Report

Chris Dalke

March 17, 2017

1 Introduction

For this homework assignment, we were tasked with implementing the K-Means Segmentation algorithm in MATLAB and using it to process an input image. The algorithm assigns labels to pieces of an image by looking at the pixel values as positions in a 3d color space. We processed the images using several color spaces: RGB, HSV, and LST, described below in more detail.

2 How to Run

2.1 Dependencies

Some of my code may require the Image Processing Toolkit to run. If it is not installed, MATLAB will show an error indicating so.

2.2 Running the program

To start the program, please run `Homework03.m`. The program will load the image that was provided for this project from the `Input` folder, and output images representing all stages of the process into the `Output` folder. The files outputted are as follows:

- `outputHSV`: The output of the algorithm for the HSV color space.
- `outputRGB`: The output of the algorithm for the RGB color space.
- `outputLST`: The output of the algorithm for the LST color space.

Note: All of the output images have been configured to use a scaled color map where the minimum value is blue and the maximum value is yellow. This makes it easy to identify the unique values for the array. These images are also displayed in this document.

3 Procedure

3.1 Generating HSV, LST Images

I generated the HSV and LST images using a mix of MATLAB functions and my own functions. MATLAB has a function to convert a color image to HSV,

and for conversion to LST I created my own function that used the formula for conversion given in the assignment prompt. One particular aspect I needed to pay attention to was to make sure the data was being manipulated as floating point numbers since integer math would break the algorithm.

3.2 Algorithm

I followed the description for the algorithm given in the homework description. The K-Means Segmentation Algorithm works by examining the values of the pixels in a given color space, and then assigning labels from within that color space.

The first part of the algorithm assigns every pixel a label by calculating the Euclidean distance between the pixel value and each of the region centers in the color space, and assigning the label of the region with the closest center to the pixel.

The second part of the algorithm takes the mean of the pixel values for all pixels currently in a region, and moves the center of that region to the mean of the pixels contained in the region. This allows for more accurate labeling when the algorithm returns to the first step.

The algorithm jumps between these two stages until the sum of the change in centroid positions has decreased below a threshold (indicating the labels have stabilized) or the algorithm passes a max iteration count

3.3 Choice of Algorithm Parameters

There are several parameters of the K-Means clustering algorithm that influence the results. The first is the value of K, which determines how many regions or labels will be applied to the image pixels. I found that the value of 3-4 labels was a good amount, since this could be used to represent the background and multiple foreground objects. Choosing the best K for a color image can best be done by examining the number of distinct objects / color groupings in an image. For example, in this image, there is a dark background, red peppers, and green peppers, so a value of k=3 is a good number to start with since this could be used to categorize the three types of objects.

The second parameter of the algorithm is the positions of the initial region centers, which heavily influences the results of the algorithm. I tried to evenly space out my region centers so that one of the labels would match the background (dark colors), one of the regions would match the red peppers, and one would match the green peppers. For example, the RGB instance of the algorithm, I set the centers to be (0,0,0), (255,100,100) and (100,255,100), or black, light red, and light green.

In order to obtain "good" results from the algorithm, I played around with the k-values and region centers until each algorithm outputted a satisfactory image.

3.4 Handling Specular Highlights

Although I did not complete this as part of the assignment, a technique for handling specular highlights would be to write an algorithm that examines the pixel values in the 3d color space. If there is a "branch" of the values that

appears to branch off of another region of color values, that branch is most likely a specular highlight if it is pointed upwards relative to the original region. The key traits of a specular highlight in a color space would be that the base of the branch is connected to a parent region of pixel values, and the highlight pixels would have an upward facing direction in the color space.

4 Results and Analysis

4.1 Color Spaces

4.1.1 RGB

The RGB color space corresponds directly to brightness values that are displayed for the red, green, and blue components on a scheme. One critical aspect of this color space is that the intensity and color are linked; there is no saturation value but instead saturation is controlled by altering the proportion of a color in relation to its other colors.

4.1.2 HSV

In this color space, the Hue (color) of a pixel is a separate value from the saturation and brightness of the pixel. This is valuable because you can examine these pieces of the image separately, for example, just the color of an image or just the brightness values of an image.

4.1.3 LST

The LST color space, also known as the LAB color space, also separates intensity from color values, but in a slightly different way from HSV: Instead of storying the saturation, the green/red and blue/yellow values are stored, which can be used to control saturation and color. LST also stores the lightness (intensity) of a pixel.

4.2 Algorithm Results

The results of the algorithm are shown below on all three of the color spaces, along with the algorithm parameters that I described in the Procedures section for each of the color spaces.

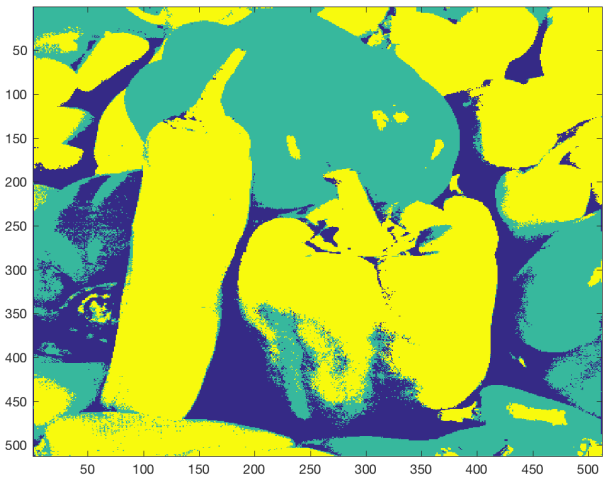

Figure 1: Output using RGB Color Space, k=3

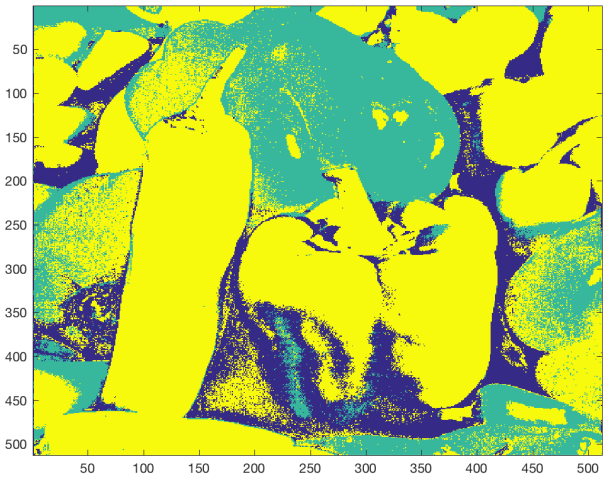

Figure 2: Output using HSV Color Space, k=3

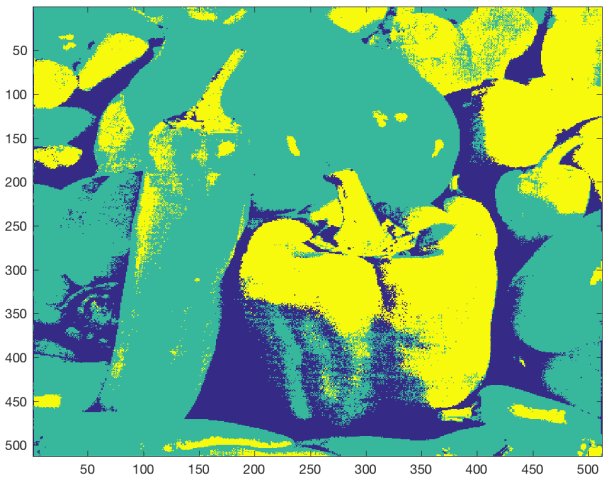

Figure 3: Output using LST Color Space, k=3

Based on the figures above, it appears that using the RGB color space gave me the best results. I suspect the reason for this is because the particular image has a large amount of variation in brightness in both the red elements and green elements, which made it harder to find effective region initial centers for the color spaces that separate color and intensity values. In the RGB color space, it was easier to find a good value for the initial centers because I could simply set a label center around the red component and a center around the green component, which produced a cleaner result.

I acknowledge that these results are skewed by my ability to choose the correct initialization values; it's entirely possible that with the correct initialization values the other color spaces could be superior to RGB. The biggest difficulty is finding those correct values to produce a useful result.

5 Academic Honesty

I did not collaborate on this assignment and all work is my own.