CS 485 Programming Assignment 2

Alexander Novotny

April 7, 2019

Problem 1 - Toon Shading

Approach

1. Break the image into three separate intensity sub-images based on color space.
2. Discretize each sub-image intensity into l buckets, where $l = 2, 4, 8, 16$.
3. Loop through each pixel of each sub-image, putting them into a bucket by changing the intensity value to the bucket number. Also keep track of the average intensity in each bucket.
4. Go back through the image, replacing each bucket value with the average intensity for that bucket.

Color Space

The BGR color space was used, as this is the default color space used by OpenCV. Since each member of the color space was compressed equally, there wasn't any particular need to use another color space.

Equations used

To quantize the space, we have $b : \mathbb{Z}^2 \to \{0, 1, 2, \ldots, l-1\}$,

$$b(x, y) = \left\lfloor \frac{l * I(x, y)}{I_{max} + 1} \right\rfloor, \tag{1}$$

where $b(x, y)$ is the bucket the pixel at (x, y) belongs to, l is the quantization-level, $I(x, y)$ is the intensity of the pixel at (x, y) and I_{max} is the maximum intensity the pixel can have (with the assumption that the minimum is always 0). For all B, G, R we have $I_{max} = 255$. Note that for $I(x, y) = I_{max}$ then $b(x, y) = l - 1$.

As well, I kept track of an average intensity value μ_n for each pixel in each bucket,

$$\mu_n = \frac{I_1 + I_2 + \cdots + I_n}{n}, \tag{2}$$

where I_i is the intensity of a pixel in the bucket and there are n such pixels in the bucket. Since at any point in time we don't know how many more pixels will be in this bucket, I wanted to be able to calculate and update this number without storing each previous intensity value. A naive way to do this might be to keep track of n and $S_n = I_1 + I_2 + \cdots + I_n = S_{n-1} + I_n$, however this can easily overflow. So instead I considered

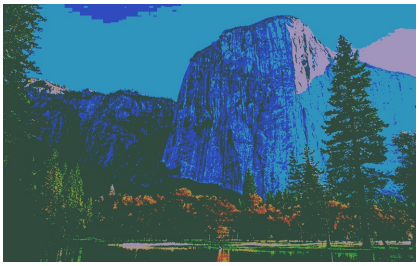
a moving average

$$\begin{aligned}\mu_{n+1} &= \frac{S_{n+1}}{n+1} \\ &= \frac{S_n + I_{n+1}}{n+1} \\ &= \frac{S_n}{n+1} + \frac{I_{n+1}}{n+1} \\ &= \frac{S_n}{n}\left(\frac{n}{n+1}\right) + \frac{I_{n+1}}{n+1} \\ &= \mu_n\left(\frac{n}{n+1}\right) + \frac{I_{n+1}}{n+1},\end{aligned} \tag{3}$$

so I only have to keep track of the current moving average μ_n and the number of pixels stored in each bin n.

Results

Figure 1: Original ElCapitan.jpg



(a) Toon shading with $l = 2$

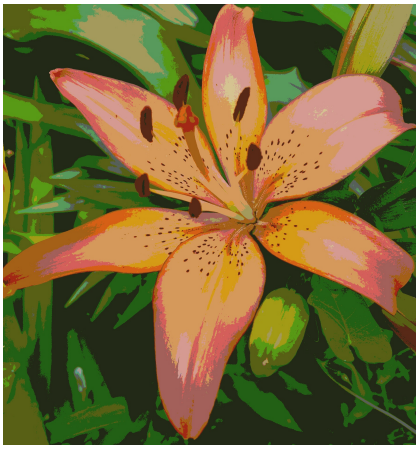
(b) Toon shading with $l = 4$

(c) Toon shading with $l = 8$

(d) Toon shading with $l = 16$

Figure 3: Original Lilly.jpg

(a) Toon shading with $l = 2$



(b) Toon shading with $l = 4$

(c) Toon shading with $l = 8$

(d) Toon shading with $l = 16$

Figure 5: Original Orchids.jpg

(a) Toon shading with $l = 2$

(b) Toon shading with $l = 4$

(c) Toon shading with $l = 8$

(d) Toon shading with $l = 16$

Figure 7: Original OrchidsY.jpg

(a) Toon shading with $l = 2$

(b) Toon shading with $l = 4$

(c) Toon shading with $l = 8$

(d) Toon shading with $l = 16$

Figure 9: Original Parrot.jpg

(a) Toon shading with $l = 2$

(b) Toon shading with $l = 4$

(c) Toon shading with $l = 8$

(d) Toon shading with $l = 16$

Problem 1 - Color Compression

Approach

1. Break the image into three separate intensity sub-images based on color space (HSV).
2. Discretize the saturation sub-image.
3. Loop through each pixel of the sub-image, finding the bucket each pixel belongs to and replacing the intensity with the minimum intensity for that bucket.

An average approach was considered, similar to toon shading, however this did not produce images similar to the ones in the project description.

Color Space

The color space which was used was HSV, as the type of color is encoded entirely in Hue, so we can compress saturation without affecting the type of color present in the pixel.

Results

Figure 11: Original ElCapitan.jpg

(a) Color compressed with $l = 2$

(b) Color compressed with $l = 4$

(c) Color compressed with $l = 8$

(d) Color compressed with $l = 16$

Figure 13: Original Lilly.jpg

(a) Color compressed with $l = 2$

(b) Color compressed with $l = 4$

(c) Color compressed with $l = 8$

(d) Color compressed with $l = 16$

Figure 15: Original Orchids.jpg

(a) Color compressed with $l = 2$

(b) Color compressed with $l = 4$

(c) Color compressed with $l = 8$

(d) Color compressed with $l = 16$

Figure 17: Original OrchidsY.jpg

(a) Color compressed with $l = 2$

(b) Color compressed with $l = 4$

(c) Color compressed with $l = 8$

(d) Color compressed with $l = 16$

Figure 19: Original Parrot.jpg

(a) Color compressed with $l = 2$

(b) Color compressed with $l = 4$

(c) Color compressed with $l = 8$

(d) Color compressed with $l = 16$

Problem 2 - Hough Transform Circle Detection

Approach

1. Discretize possible radius values into buckets
2. Find all trees of each radius bucket, starting from the smallest
3. If a tree already exists in one of these locations, there can't be another. So throw out the duplicate

Results

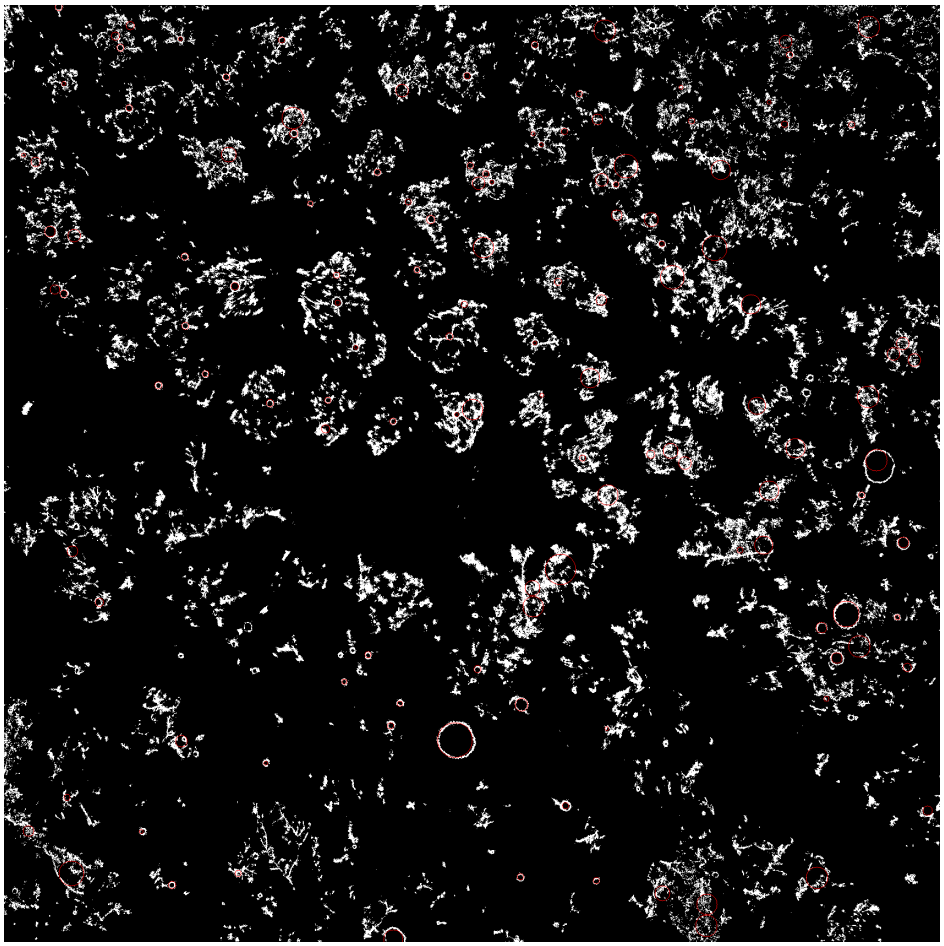


Figure 21: LIDAR with trees circled

The run time for the program was .468716 seconds.