DIP Fall 2024 #58195
HW2
Jeffrey Kerley
29 September 2024

Segmentation using Otsu Thresholding

Abstract:

The use of advanced mathematics for image processing extends to linear algebra, clustering, and even statistics. Using ideas about variance and means, and associated pixels intensity values to different pixel classes, max or min class variance can be applied with different effects. Using Otsu thresholding the pixel classes are foreground and background where the variance can be maximized or minimized with respect to inner class vs between class. These yield the same thresholding results since the result is foreground with similar pixel values and background pixels with similar values within the image histograms range of values. Running Otsu thresholding on RGB images can show how often a single channel holds the discrimination information for the thresholding value.

Introduction:

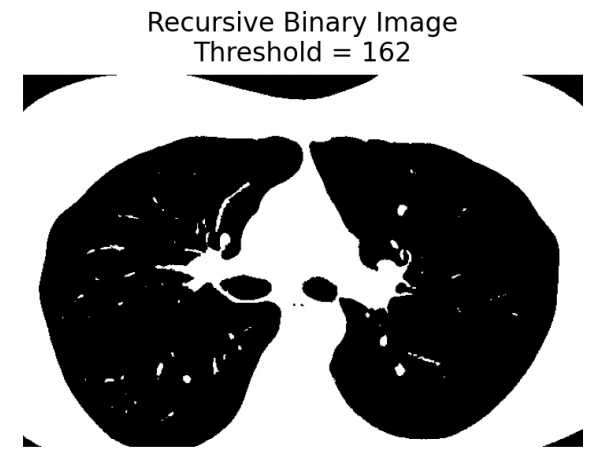
Otsu's method works by defining a foreground and a background, which are the 2 classes we are max or minning (inter vs intra) class variance, and then finding a threshold value (n pixel intensity threshold values that divide the image histogram into 2 classes) which corresponds to our min intra-class variance, seen as figure 1 (wikipedia).

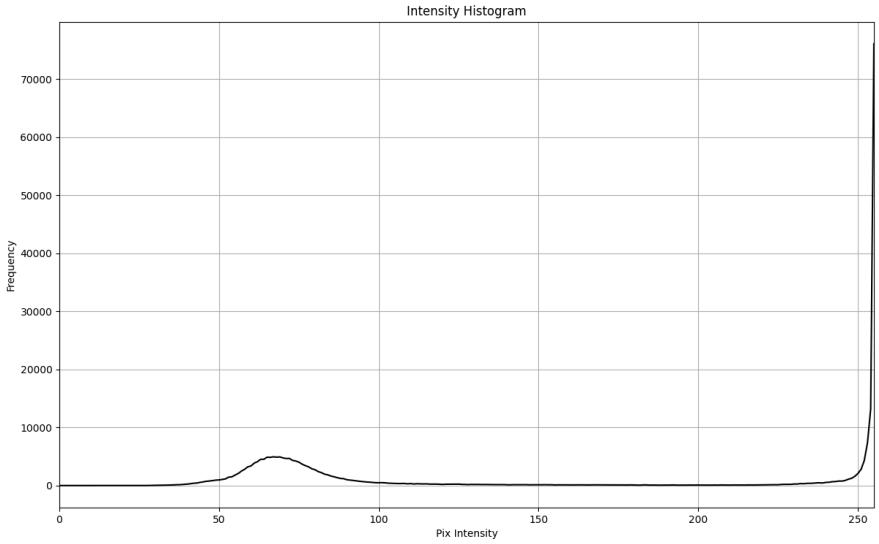
$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Fig 1: Otsu thresholding min-class variance equation described as a weighted sum of the variances. We use the image histogram bins to form our class probabilities given a

threshold value t. This is also used to get our variance, which uses the foreground and background means (again, all based on the threshold value currently selected).

Experiments and Results:

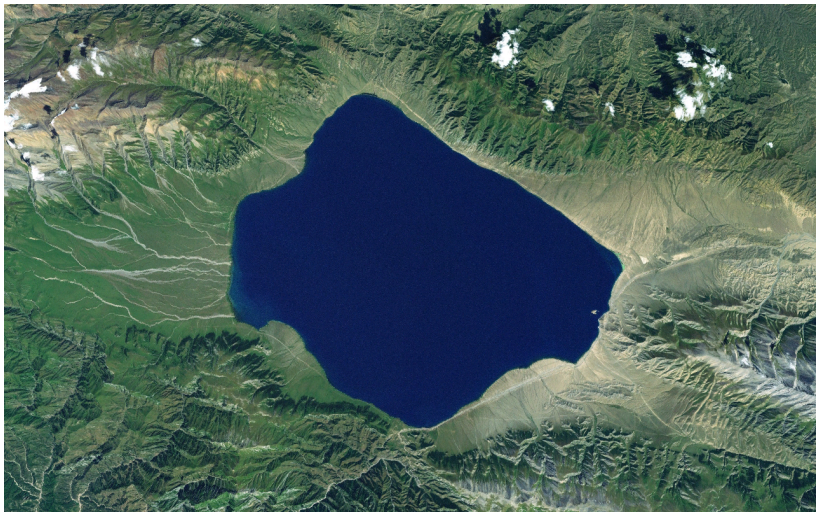




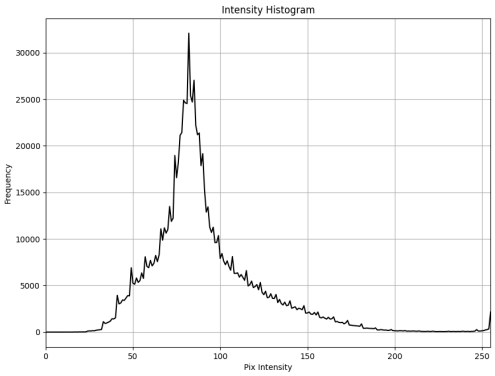
TIME:

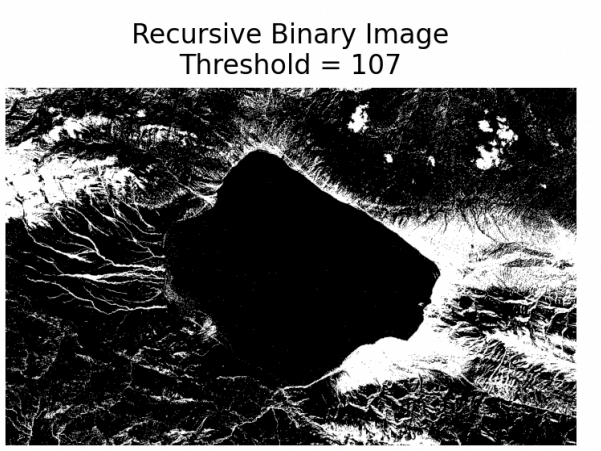
Recursive Threshold: 162, Time Taken: 0.001393 seconds

Iterative Threshold: 162, Time Taken: 0.001146 seconds

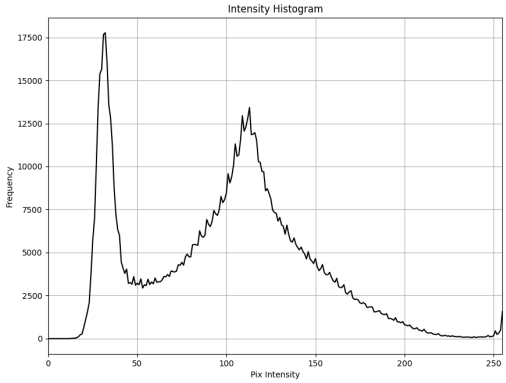


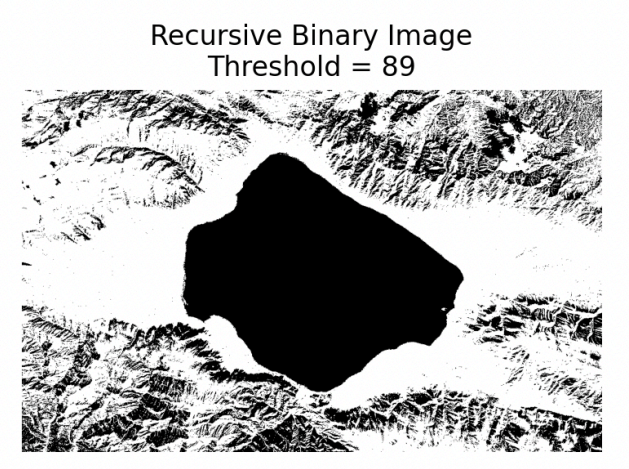
BLUE:


Intensity Histogram
Frequency
Pix Intensity
0
5000
10000
15000
20000
25000
30000
0
50
100
150
200
250

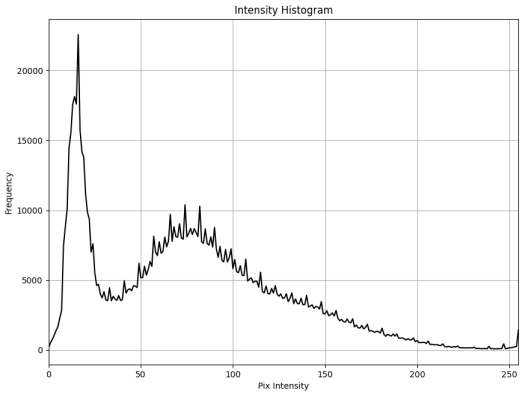

Recursive Binary Image
Threshold = 107

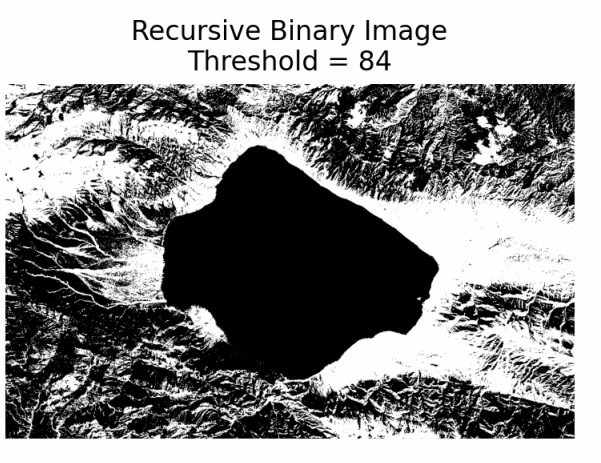
GREEN:


Intensity Histogram
Frequency
Pix Intensity
0
2500
5000
7500
10000
12500
15000
17500
0
50
100
150
200
250


Recursive Binary Image
Threshold = 89

RED:





TIME:

Blue Channel:

Recursive Threshold: 107, Time Taken: 0.002591 seconds

Iterative Threshold: 107, Time Taken: 0.001028 seconds

Green Channel:

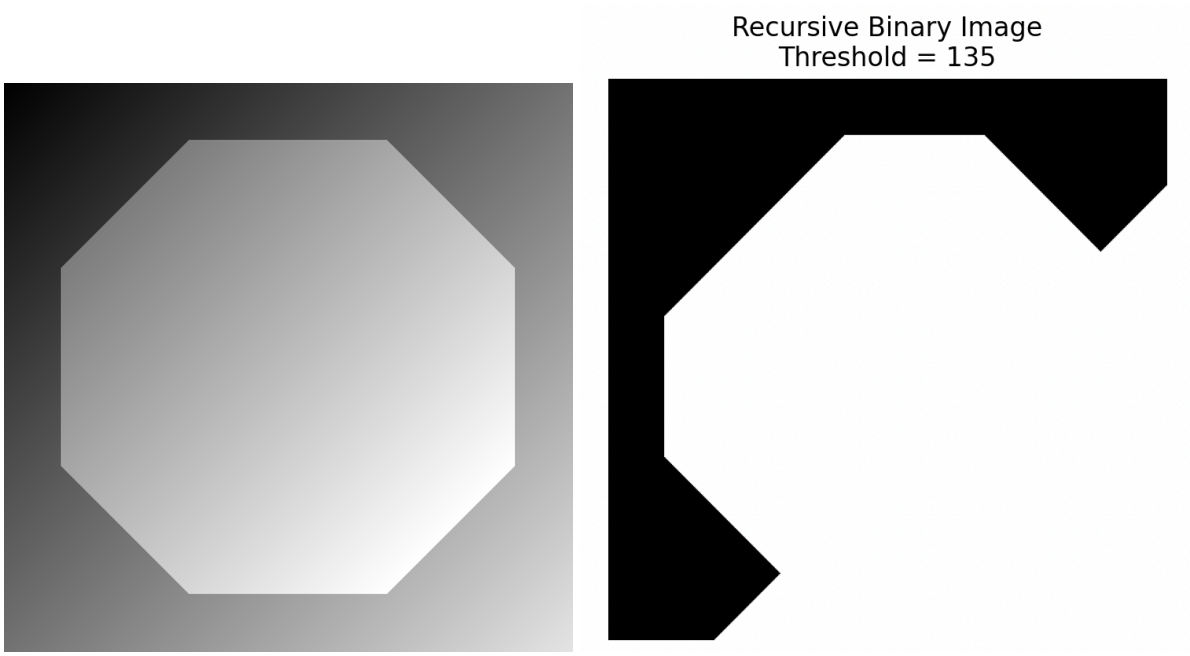
Recursive Threshold: 89, Time Taken: 0.003364 seconds

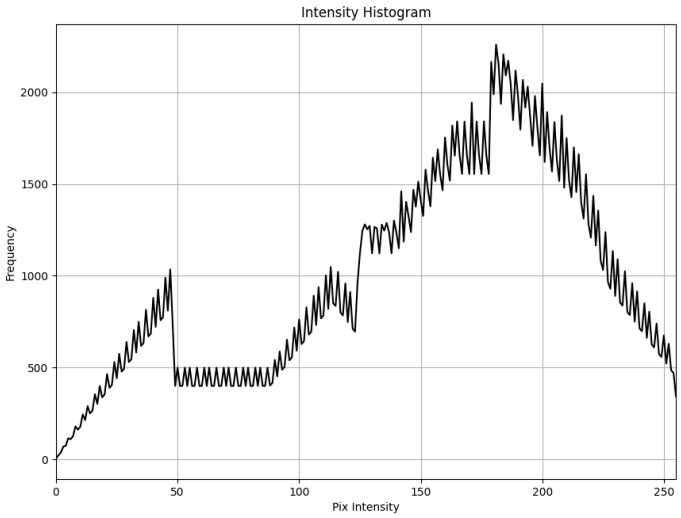
Iterative Threshold: 89, Time Taken: 0.002658 seconds

Red Channel:

Recursive Threshold: 84, Time Taken: 0.003644 seconds

Iterative Threshold: 84, Time Taken: 0.003076 seconds





TIME:

Recursive Threshold: 135, Time Taken: 0.004719 seconds

Iterative Threshold: 135, Time Taken: 0.003841 seconds

Discussion/Conclusions:

Overall, using Otsu's method results in a decision boundary between two classes (foreground and background) that will maximize between class variance, which results in a binary thresholded image. If two classes exist within the image histogram, then the otsu thresholding will find an optimal threshold value to put the classes into binary masked image with class 0 and class 1. Seeing how certain channels, such as the green channel in the lake example, contribute to a majority of the binary mask features is interesting but makes sense when looking at the histogram intensity values where there are two distinct classes with high intensity values respectively. The red channel for this lake example has two classes, but the intensity of the respective classes are lower, and so the threshold value chosen does not result in a better segmented image (segmenting the image into background mountain/land vs foreground lake). The octagon example shows a similar fate as the blue channel in the lake image, where there is really just one strong 'class' and so separating into foreground and background using mean and variance does not result in a clearly segmented image since the octagon edges do not fall into the two distinct classes.

References/Appendix: (As needed)

Otsu Method,
https://en.wikipedia.org/wiki/Otsu%27s_method