Nicole Wallack and Kaliel Williamson
Bayesian Data Analysis
Professor Knuth
Final Project

ImageNEST: Computer Vision Using Nested Sampling

Abstract

A single image is a matrix of pixels, each with a set of three different numbers corresponding to the red, blue, and green channels for each pixel of the image. Bayesian classifiers have been developed that use trained neural networks to classify objects in an image based on this data. Here we use nested sampling, an alternative machine learning method to neural networks, based on Bayesian inference. Using this paradigm, different images and sub-image objects can be categorized using extracted information from the color, shape, and luminosity of an image. This was done by applying a Y'UV color transformation to the original RGB image, considering each channel as a histogram, and using horizontal-vertical projection algorithms to generate shape histograms. Nested sampling histogram models were systematically compared to the original histograms until an optimized model was found or the program terminated due to computational limitations. By generating Gaussian mixture models this system can be used for image segmentation and image query on a database using Bayes' Theorem. We present here a rudimentary computer vision process for image segmentation and characterization using these parameters and nested sampling.

Introduction

Images are made up of a matrix of pixels each of a different value. The matrix of pixels can be analyzed using various regimes. Y'UV is the color space that encodes the image in terms of one parameter for the luminosity of the image and two for the color components and can be transformed from RGB. A histogram can then be plotted for each of these parameters. Moreover, these histograms can be modeled as Gaussians for the purpose of image segmentation. For example, each Gaussian corresponds to a different object in the image.

The sum of Gaussian component densities can be determined through a parametric density function, called a Gaussian mixture model (GMM). GMMs are the superposition of a set of Gaussian functions. One application of these GMMs can be used for modeling continuous measurements and features in biometric systems. The parameters of GMM are determined by training data (Reynolds, 2008).

This application of computer vision uses nested sampling to find near-optimum Gaussian mixture models for a given image. Existing systems use more popular methods, especially the neural network for Bayesian inference GMM image characterization. Therefore, there is a novel application here by using nested sampling.

Compared to neural networks, nested sampling does not need to be trained before generating GMMs and characterizing data because of the GMM theory behind this. Like neural network database applications, nested sampling requires a trained characterizer. It is likely that for image classification, nested sampling is more efficient in achieving the end result because unlike neural networks, this does not require a very large network of virtual-neuron nodes, each changing some parameter pseudo-randomly based on a cost function. Here, parameters are changed randomly and redirected based on a cost function, but there are fewer levels of complexity.

Methods

Four different histograms were created using the Y'UV format- two for Chroma (blue-luminosity & red-luminosity) and one for the luminosity. These histograms were then smoothed and the peaks were determined. The ultimate purpose in doing this was to be able to model the histograms as superimposed Gaussian functions. Peaks were determined heuristically. For each peak, a Gaussian function could then be used to model the histogram. However, in order to plot the combination of the Gaussians, the standard deviation of the peak had to be determined. This was done using nested sampling.

Nested sampling is used when analytical methods for doing Bayesian calculations are not possible or are inadequate. Nested sampling uses Markov Chain Monte Carlo methods to determine the sorted likelihood using randomly chosen samples with respect to the prior. The randomly sampled objects are subject to the constraint that the likelihood cannot go below the current value of the sorted likelihood function. Nested sampling iterates over the prior probability space by restricting the priors to the minimum probability found during each iteration. Threshold probabilities are assigned if the model has a higher likelihood than the previous step in the Markov Chain. The next iteration will only be accepted if the probability is greater than the threshold probability. Therefore, the limiting value of the likelihood changes as the algorithm iterates, theoretically converging on the most optimum model. Nested sampling will terminate the iterations when the user decides that most of the evidence has been found (Sivia and Skilling, 2006).

The nested sampling algorithm was employed to determine the standard deviation of each of the peaks. The algorithm took a random walk up probability space generated by randomly assigned standard deviation values within a threshold until it found the most probable value or terminated due to large computational expense. From the values of the standard deviation of each of the peaks, all of the Gaussians from the histogram could be plotted to represent the original smoothed histogram. The Gaussians could then be combined to represent the model histogram. with each iteration, the generated model histogram is compared to the original histogram using a cost function that calculates Euclidean distance (given by $D(P,Q) = \sqrt{\sum_1^N (p_1 - q_1)^2}$ where P is the model histogram, Q is the original histogram, and N is the total number of bins in the histogram) between said histograms. This system therefore minimizes the histogram Euclidean

distance. The cost function $C(P,Q) = 1 - \frac{D(P,Q)}{M}$ must be optimized to 1, where M is the total pixels and D is the Euclidean distance.

To be able to determine information about the overall shape of the images, a shaping algorithm was created. This rudimentary algorithm works by determining if a pixel is below or above a threshold value, then summing its corresponding rows or columns if the pixel satisfies the threshold constraint. Here, we used a white background for the threshold, so that anything not white within the image is considered foreground. This basic algorithm therefore only works on images that are on plain white backgrounds. Image blobbing or edge calculation can be done to achieve similar results where backgrounds are not white by constraint.

Using the three color histograms and the two dimensional shape histograms, a determination could be made on an image database query utilizing ImageNEST about whether two or more images were similar and to what confidence level.

Results

The ImageNEST algorithm attempts to improve on existing systems such as the S-TAPPMOG (Cristani and Murino, 2007) scheme that models GMMs using a particle filtering on per-pixel parameters. The S-TAPPMOG is designed for foreground-background characterization of video paradigm. The probability integral for each Gaussian in the GMM can therefore be generated for this characterization. Like S-TAPPMOG, ImageNEST can assign foreground-background characterization to objects in a frame/image based on the GMMs by noting that pixel members of the same Gaussian tend to be part of the same object in an image, and the larger Gaussian areas tend to be foreground objects. With ImageNEST analysis, the Cartesian location for the center of the foreground object can be estimated along with properties like shape, color, luminosity. Unlike S-TAPPMOG however, ImageNEST uses nested sampling but it comes at a high computational cost that prevents it from having application in video processing and therefore any aid to background modeling that comes from video as told in the conclusion.

In order to test the ImageNEST algorithm, images of a black circle on a white background (Figure 1), and a more complicated image of a green triangle with different smaller colored circles on each side of it (Figure 2) were analyzed.

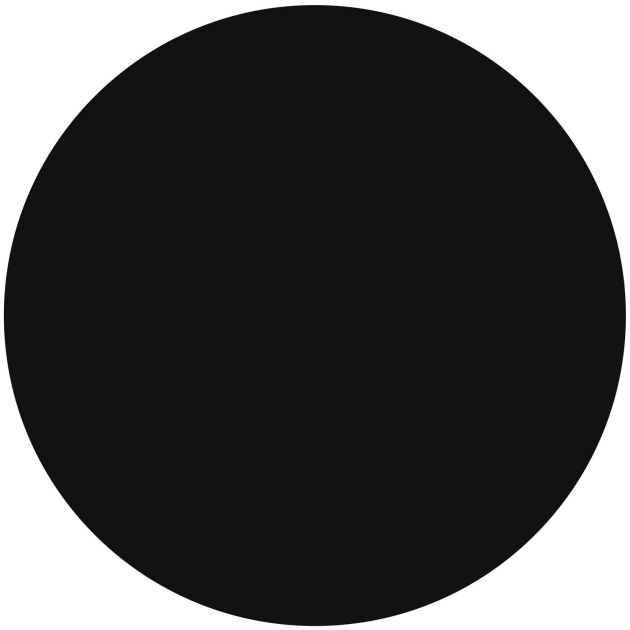

Figure 1: An image of a black circle on a white background that was analyzed by the ImageNEST algorithm.

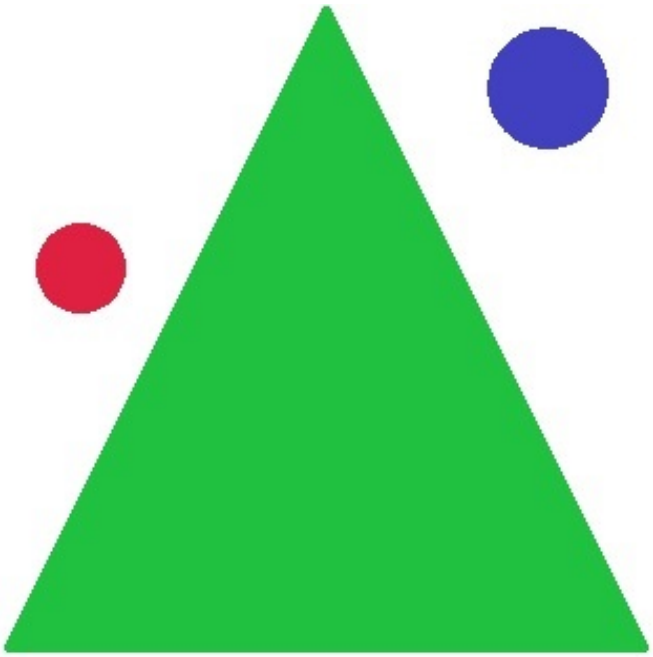

Figure 2: An image of a green triangle with different colored circles on either side on a white background that was analyzed by the ImageNEST algorithm.

The ImageNEST algorithm was first run on the black circle on the white background. A histogram for each of the two color channels, each of the two spatial channels, and the luminosity channel were created and smoothed (as shown in Figure 3, Figure 4, and Figure 5).

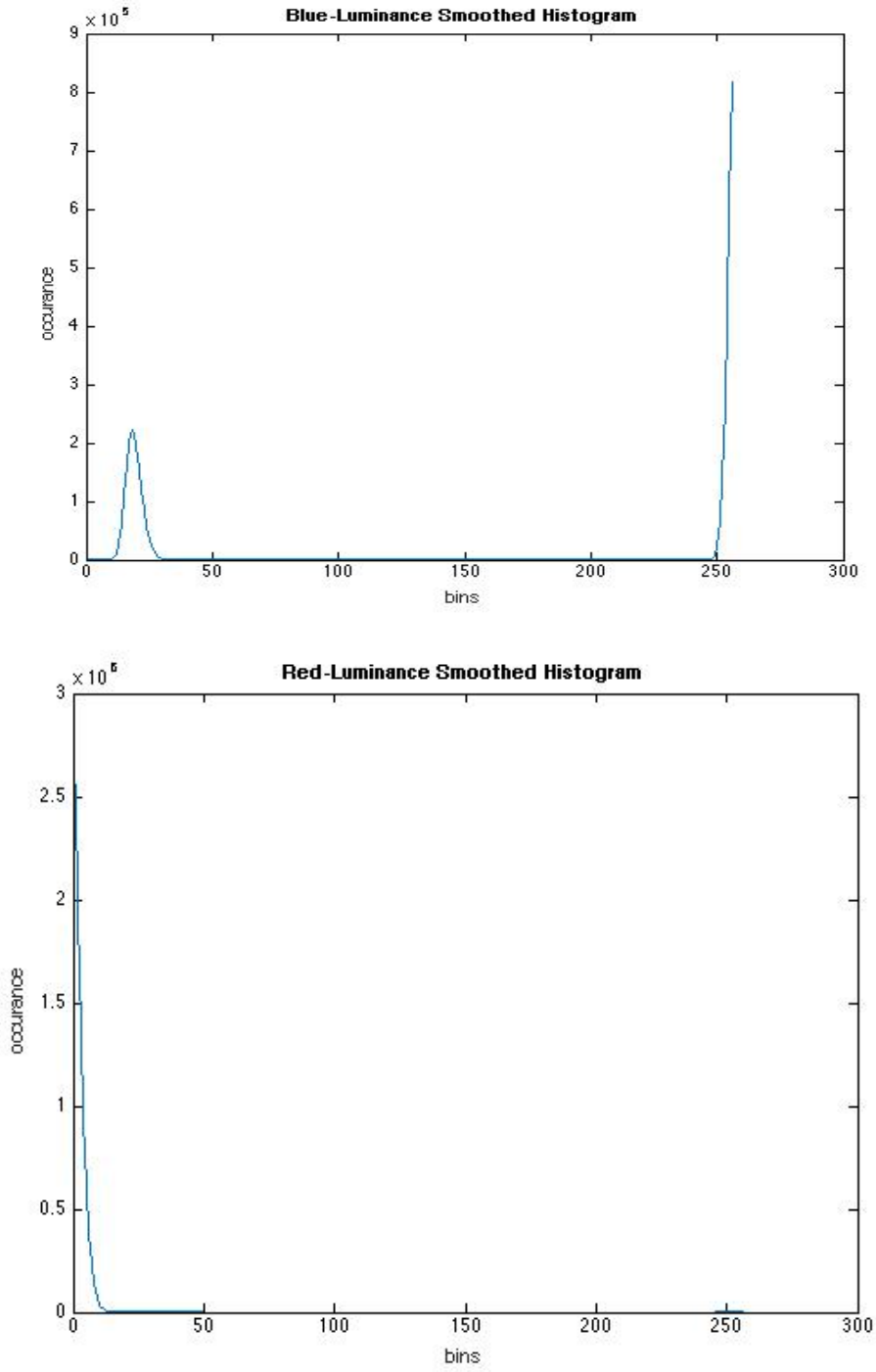

Figure 3: Smoothed histograms of the two color channels for the black circle.

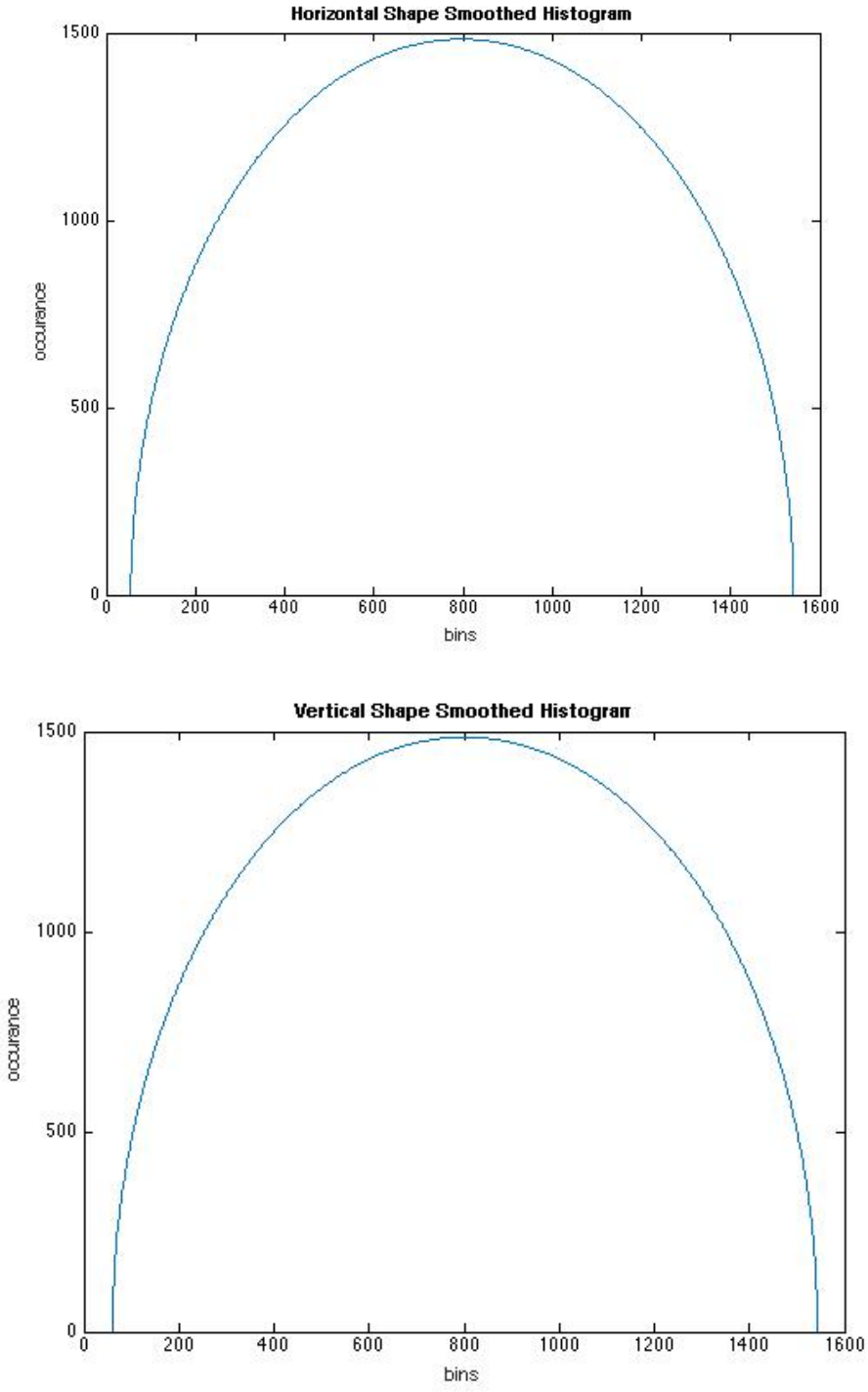

Figure 4: The smoothed histograms for the two spatial channels for the black circle.

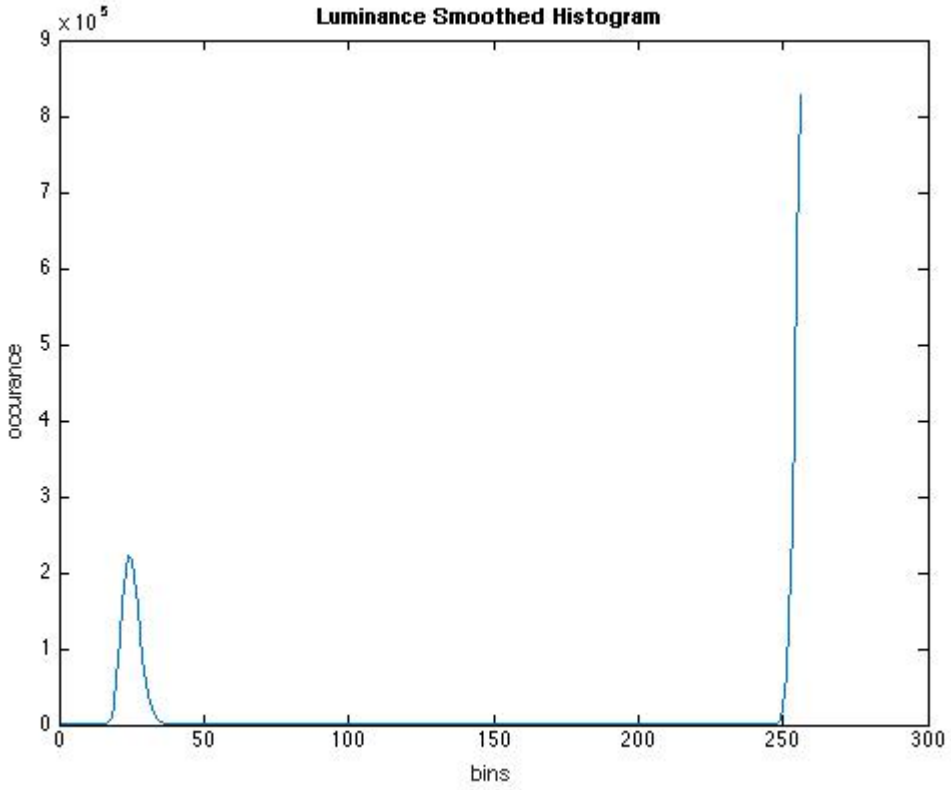

Figure 5: A smoothed histogram of the luminosity of the black circle.

The peaks in the smoothed histogram were determined and using ImageNEST, a Gaussian function was fitted to each of the calculated peaks. Using the GMM, the Gaussian functions were then mixed together (as shown in Figure 6, Figure 7, Figure 8, Figure 9, and Figure 10).

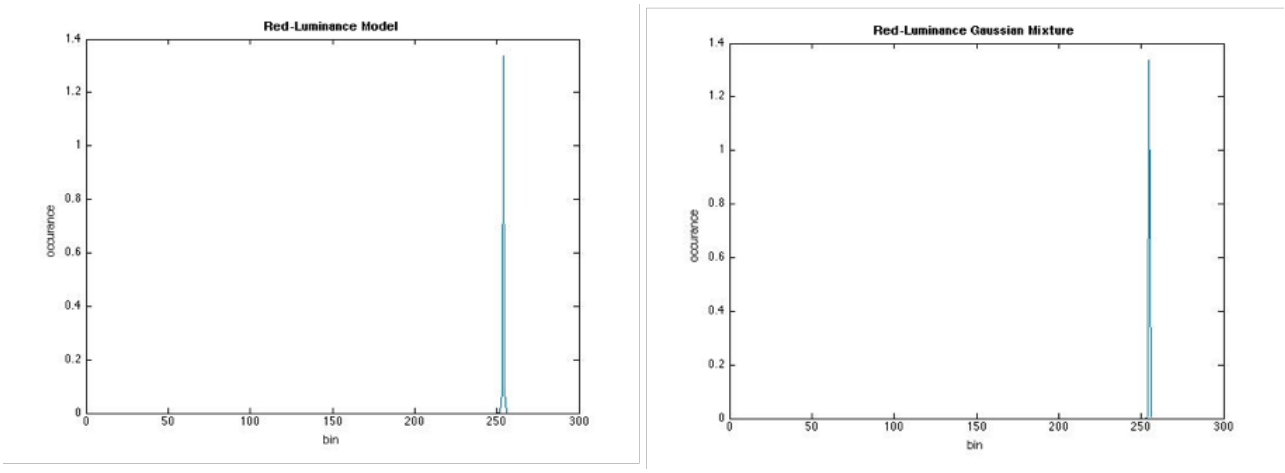

Figure 6: The Gaussian modeled peaks determined by ImageNEST on the left and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the red channel. There was only one Gaussian (because the image was only black), so the GMM did not actually combine the functions.

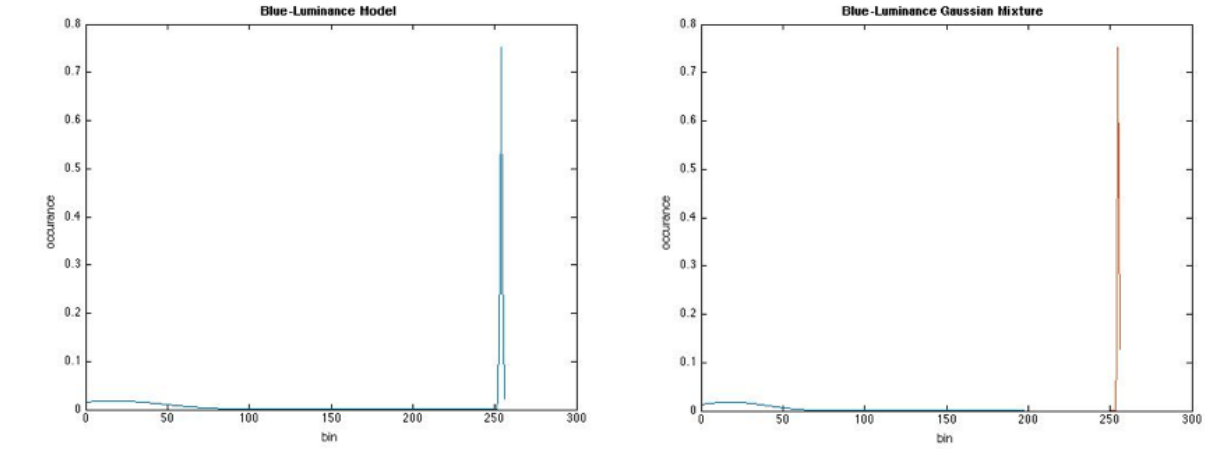

Figure 7: The Gaussian modeled peaks determined by ImageNEST on the left (each in a different color) and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the blue channel.

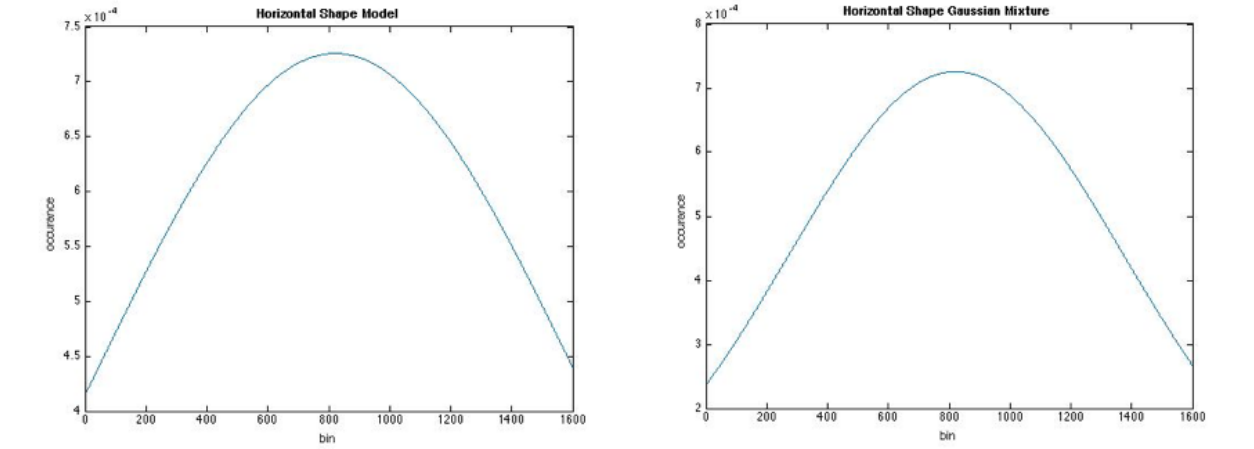

Figure 8: The Gaussian modeled peaks determined by ImageNEST on the left and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the horizontal channel. There was only one Gaussian (because the image was symmetric about the horizontal axis), so the GMM did not actually combine the functions.

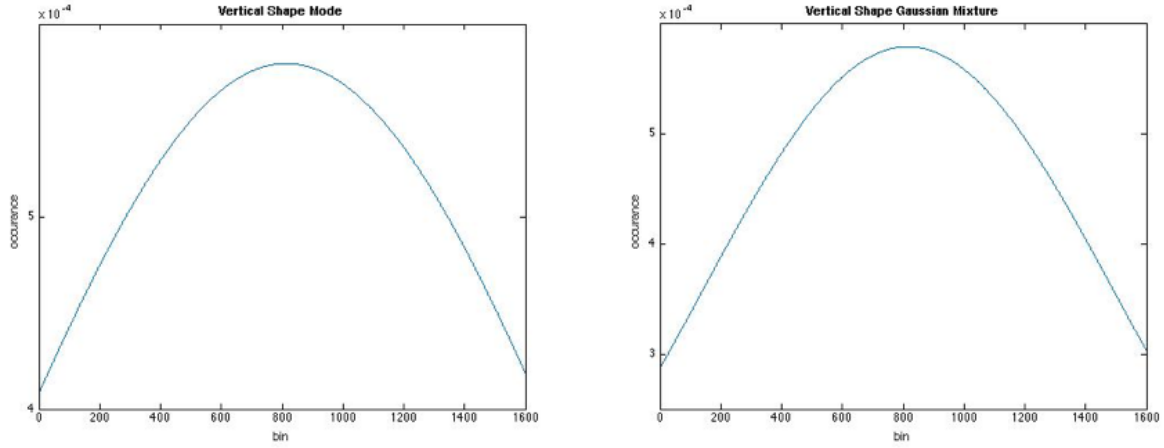

Figure 9: The Gaussian modeled peaks determined by ImageNEST on the left and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the vertical channel. There was only one Gaussian (because the image was symmetric about the vertical axis), so the GMM did not actually combine the functions.

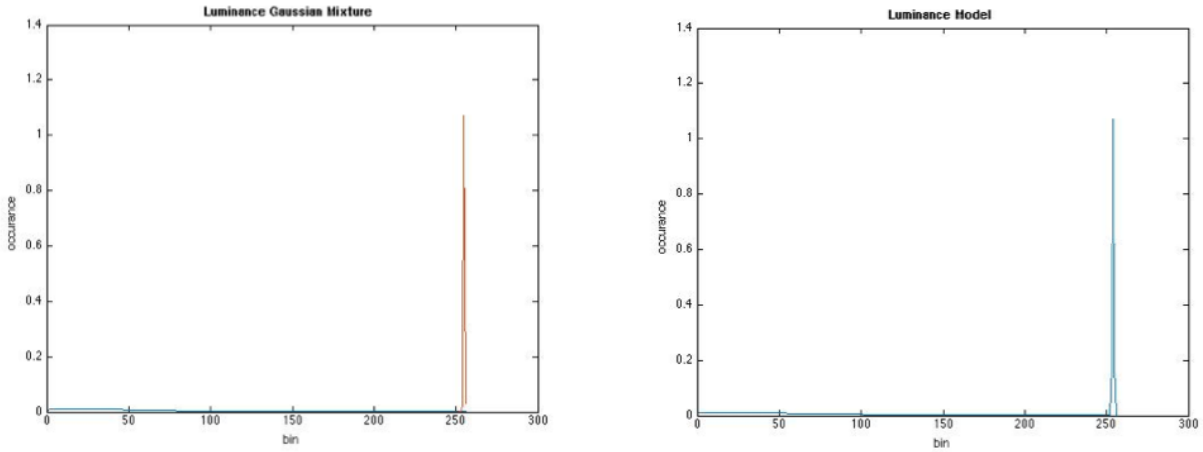

Figure 10: The Gaussian modeled peaks determined by ImageNEST on the left (each in a different color) and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the black circle.

The ImageNEST algorithm was then run on the triangle image. A histogram for each of the two color channels, each of the two spatial channels, and the luminosity channel were created and smoothed (as shown in Figure 11, Figure 12, and Figure 13).

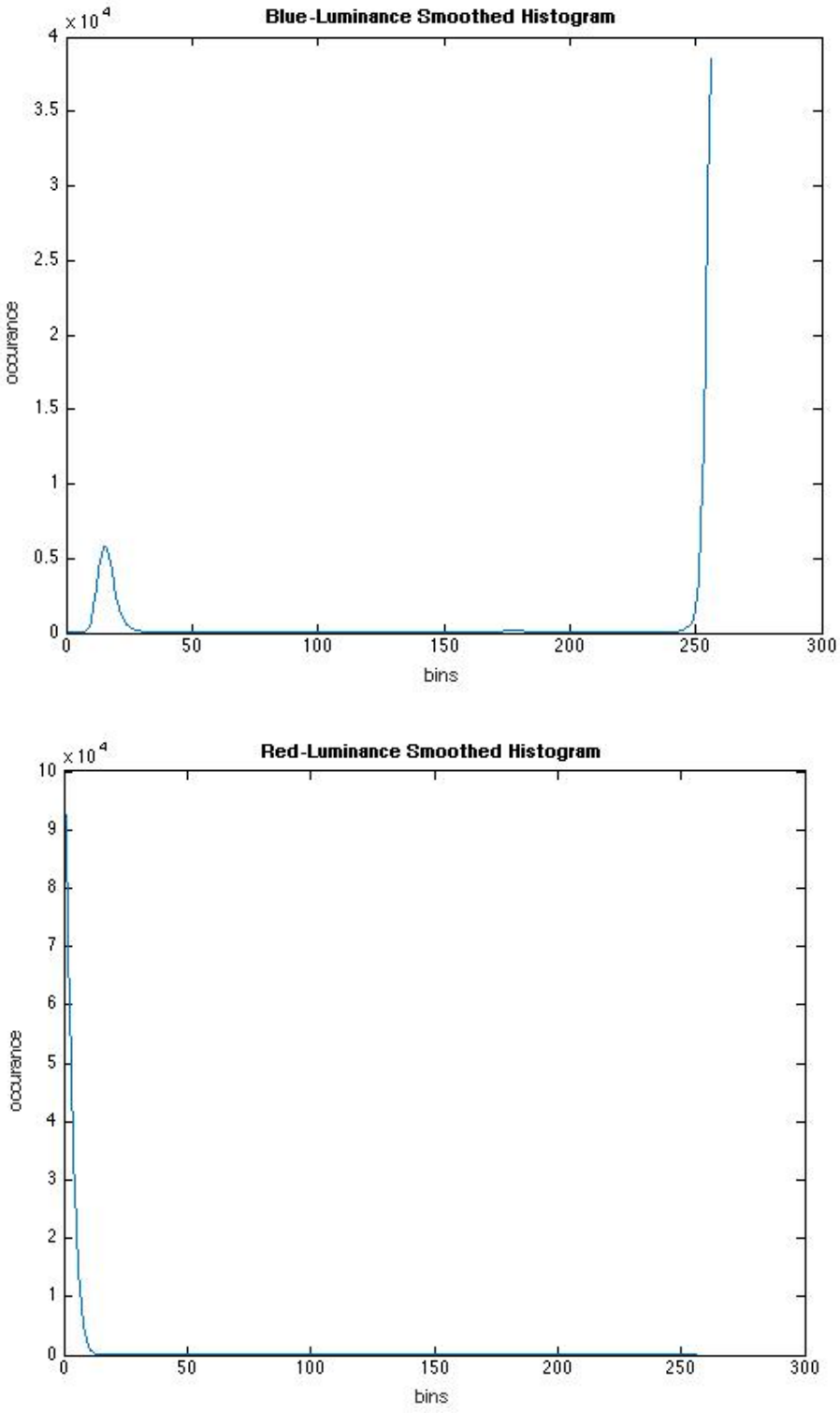

Figure 11: Smoothed histograms of the two color channels for the green triangle.

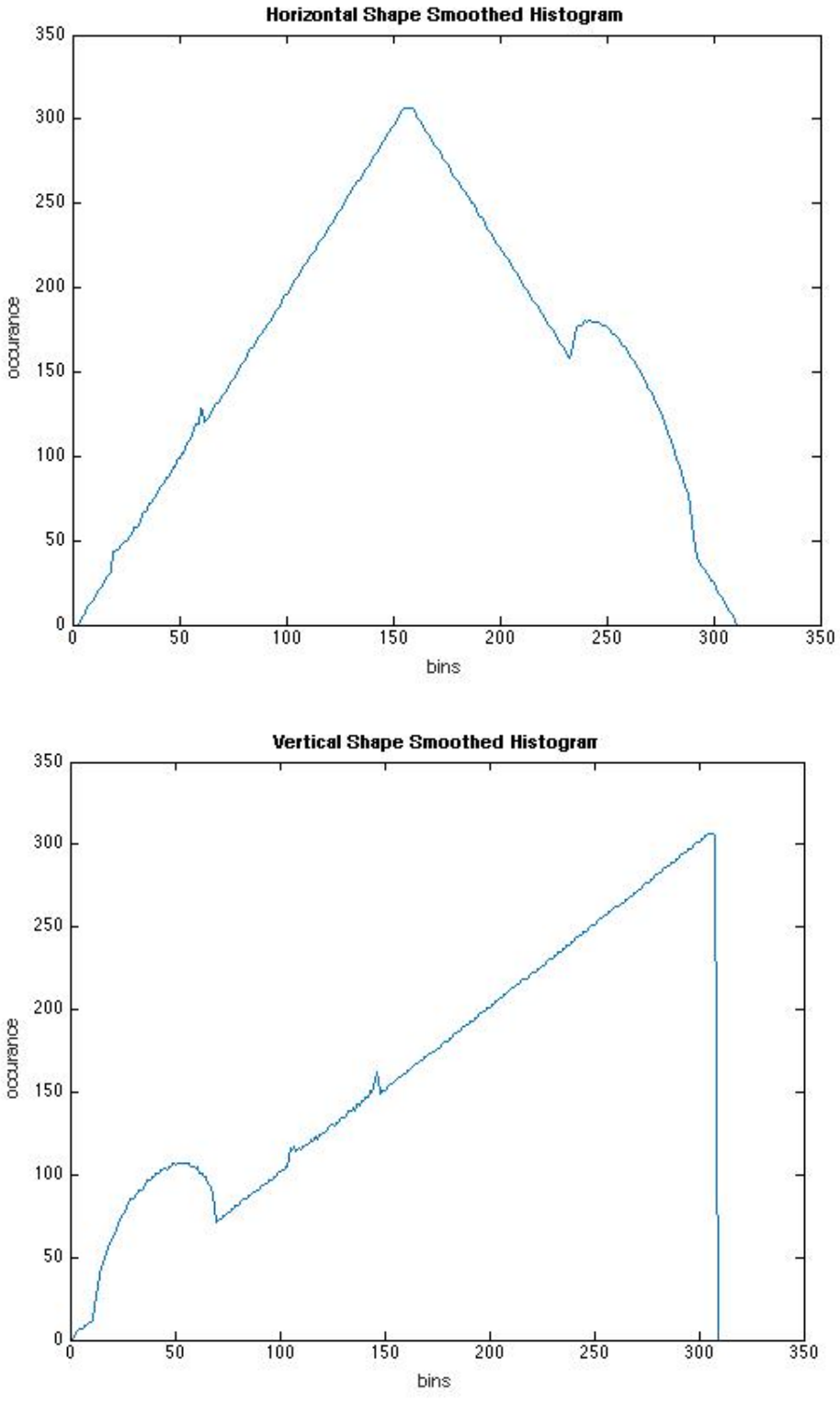

Figure 12: The smoothed histograms for the two spatial channels for the green triangle.

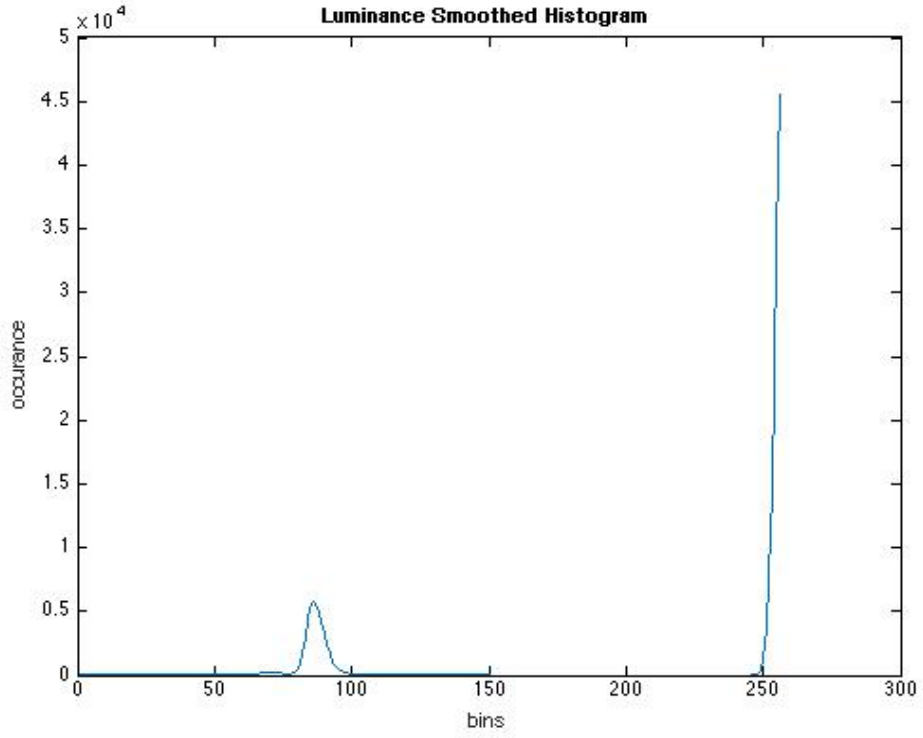

Figure 13: A smoothed histogram of the luminosity of the green triangle.

The peaks in the smoothed histogram of the green triangle were determined. Using ImageNEST, a Gaussian function was fitted to each of the calculated peaks. Using the GMM, the Gaussian functions were then mixed together (as shown in Figure 14, Figure 15, Figure 16, Figure 17, and Figure 18).

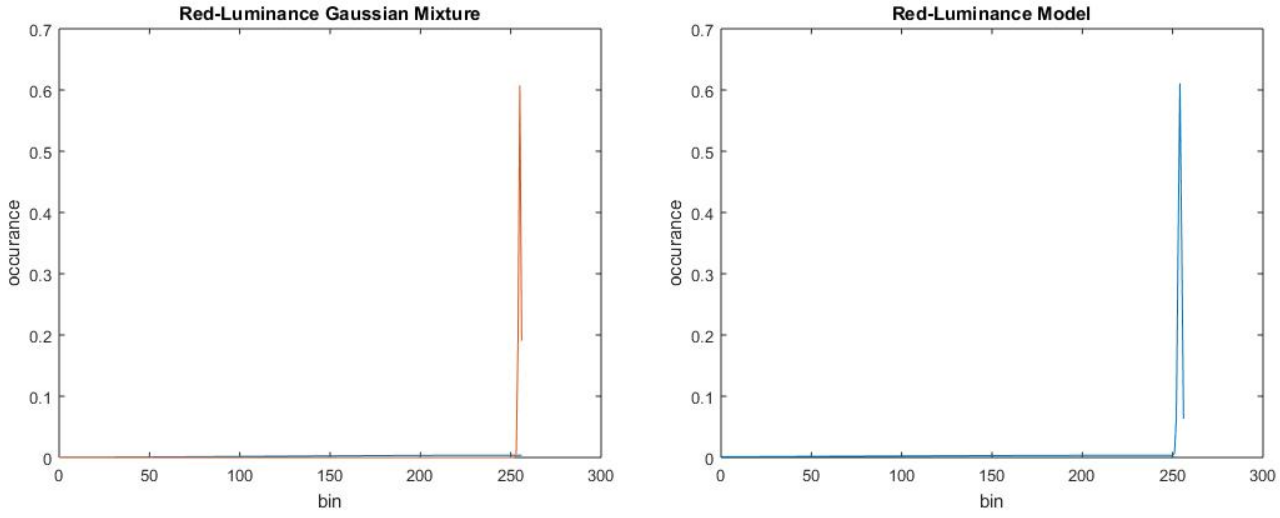

Figure 14: The Gaussian modeled peaks determined by ImageNEST on the left and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the red channel.

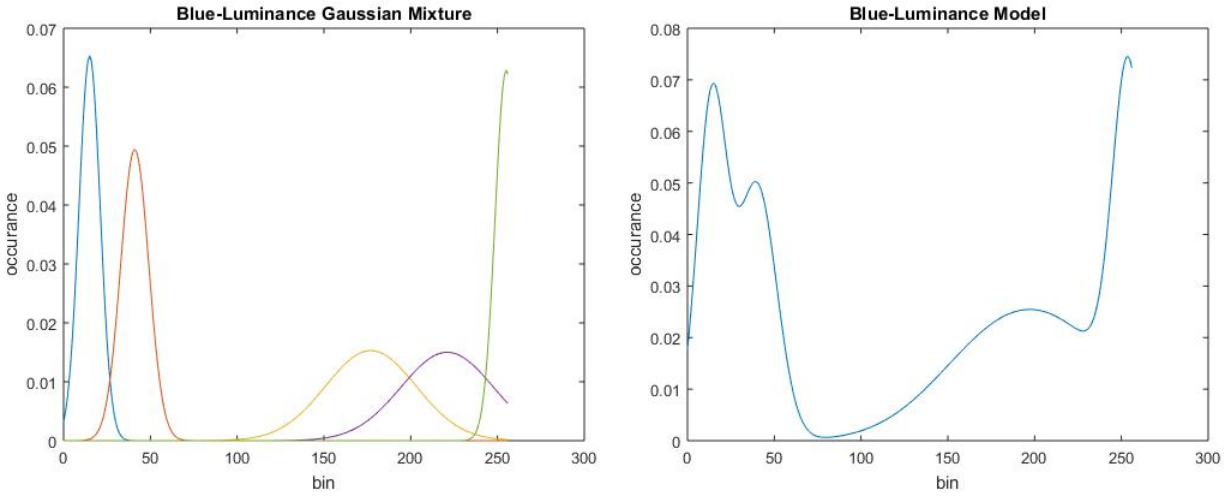

Figure 15: The Gaussian modeled peaks determined by ImageNEST on the left (each in a different color) and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the blue channel.

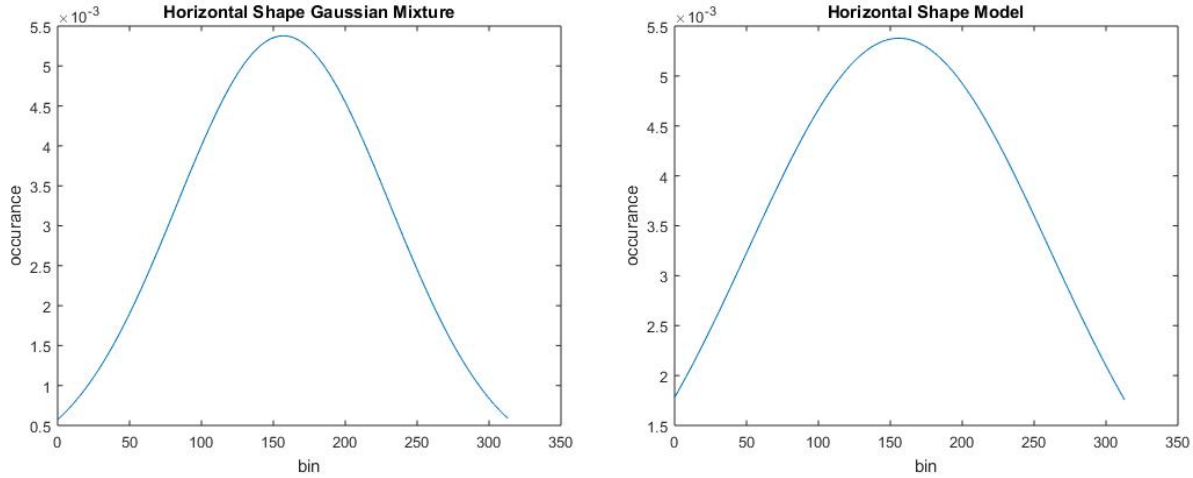

Figure 16: The Gaussian modeled peaks determined by ImageNEST on the left and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the horizontal channel. There was only one Gaussian, so the GMM did not actually combine the functions.

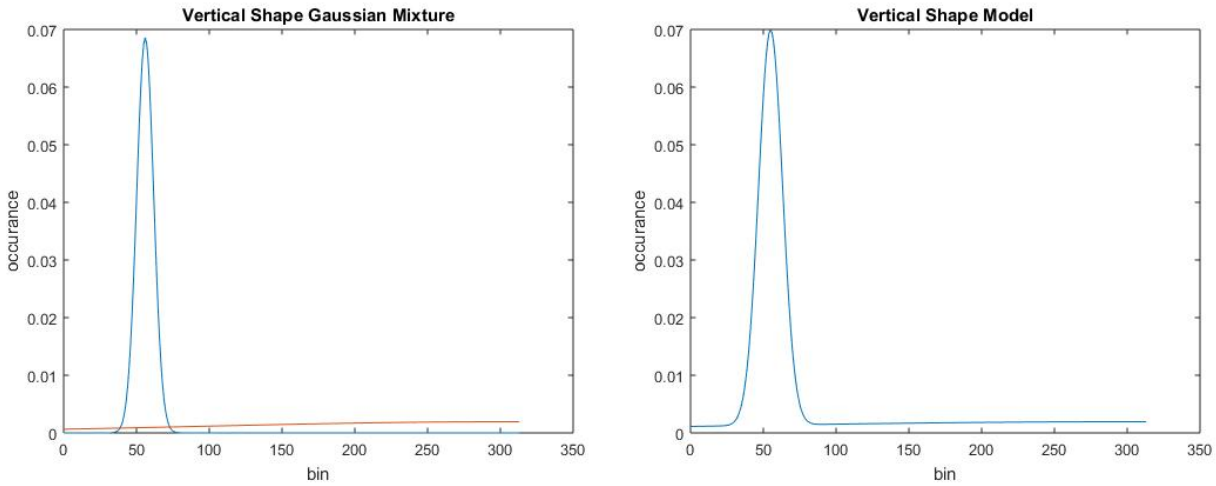

Figure 17: The Gaussian modeled peaks determined by ImageNEST on the left (with each peak in a different color) and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the vertical channel.

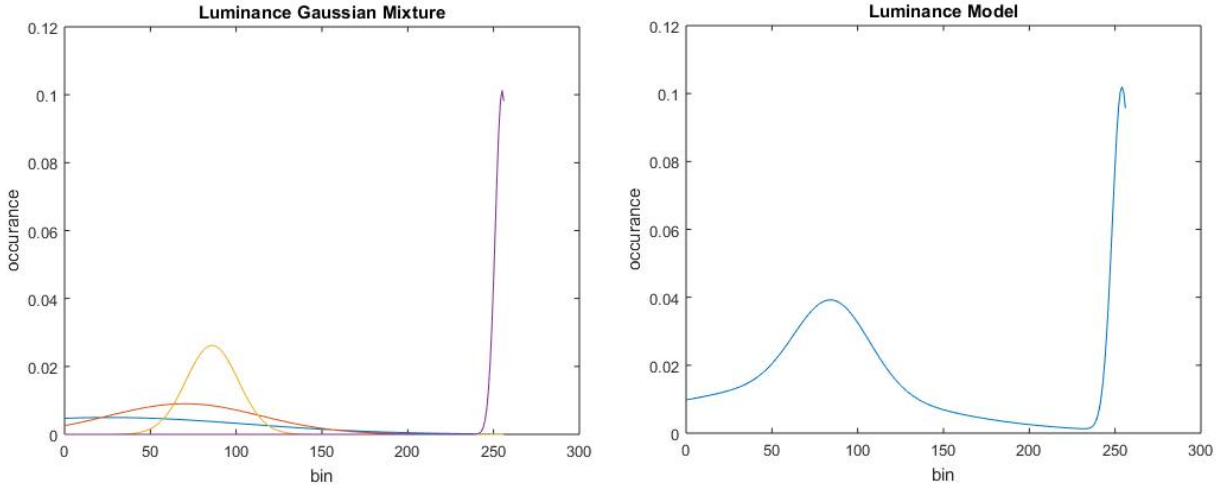

Figure 18: The Gaussian modeled peaks determined by ImageNEST on the left (each in a different color) and a plot of the GMM combined Gaussian peaks on the right for the luminosity of the green triangle.

The system is set to stop iterating if it has found around 20 increasingly probable histogram models or if it has reached a plateau in the probability space. Figure 19 depicts that neither situation has happened, that despite trying thousands of times, the system is unable to find 20 increasingly optimal histogram models. This could be an example of the system randomly spawning in a probability space local maximum, unable to find a global maximum or a higher local maximum, or some other traditional machine learning issue. Figure 20 shows the output of the ImageNEST algorithm.

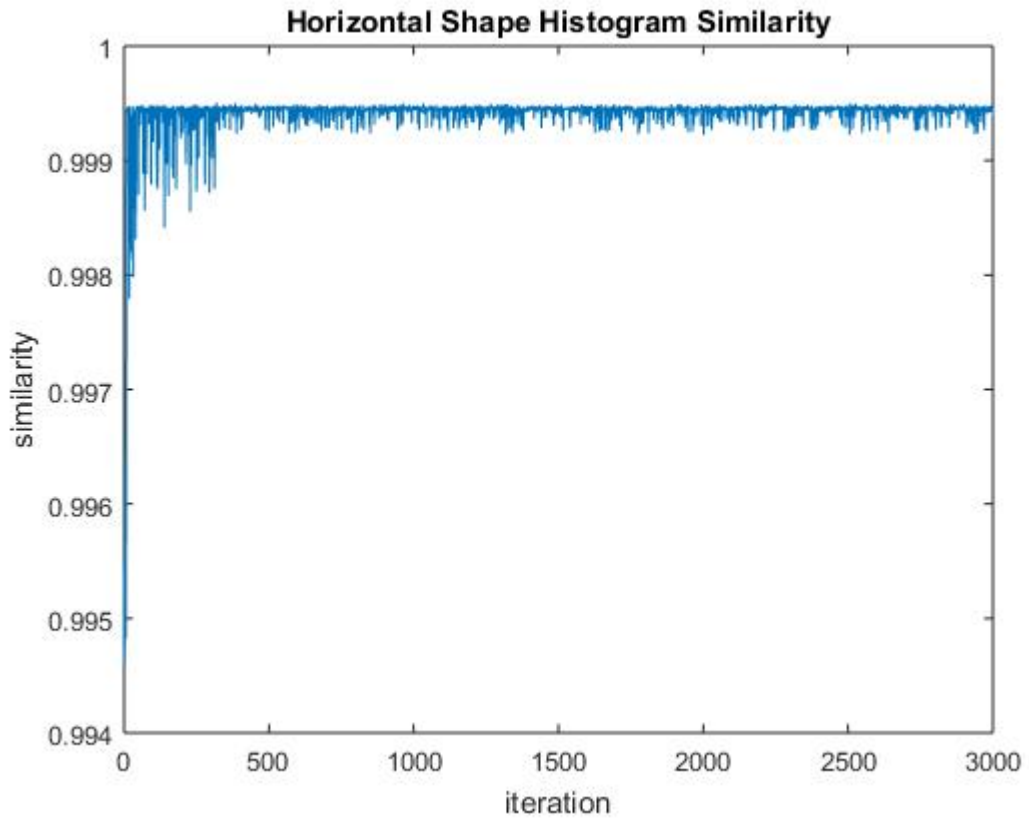

Figure 19: A description of the cost function over the maximum possible 3000 iterations.

Horizontal Shape
Foreground Object: 823
minimum threshold: -73.516
maximum threshold: 1719.52
number of peaks: 1
found appropriate points: 8
original threshold: 0.999838
final threshold: 0.999928

Vertical Shape
Foreground Object: 816
minimum threshold: -159.365
maximum threshold: 1791.37
number of peaks: 1
found appropriate points: 8
original threshold: 0.999554
final threshold: 0.999926

Luminance
Foreground Object: 24
minimum threshold: 23.9277
maximum threshold: 24.0723
number of peaks: 2
found appropriate points: 14
original threshold: 0.999905

```
final threshold: 0.999999
----------------------
Blue-Luminance
Foreground Object: 18
minimum threshold: 17.7478
maximum threshold: 18.2522
number of peaks: 2
found appropriate points: 9
original threshold: 0.999941
final threshold: 0.999999
----------------------
Red-Luminance
Foreground Object: 255
minimum threshold: 254.996
maximum threshold: 255.004
number of peaks: 1
found appropriate points: 7
original threshold: 0.999981
final threshold: 1
----------------------

ans =

elapsed time (s): 46.4201

>> runGauntlet
Horizontal Shape
Foreground Object: 157
minimum threshold: 85.7455
maximum threshold: 228.254
number of peaks: 1
found appropriate points: 9
original threshold: 0.995647
final threshold: 0.99913
----------------------
Vertical Shape
Foreground Object: 56
minimum threshold: -47.2102
maximum threshold: 159.21
number of peaks: 2
found appropriate points: 10
original threshold: 0.999374
final threshold: 0.999042
----------------------
Luminance
Foreground Object: 70
minimum threshold: 69.3518
maximum threshold: 70.6482
number of peaks: 4
found appropriate points: 12
original threshold: 0.997605
final threshold: 0.997748
----------------------
Blue-Luminance
Foreground Object: 177
```

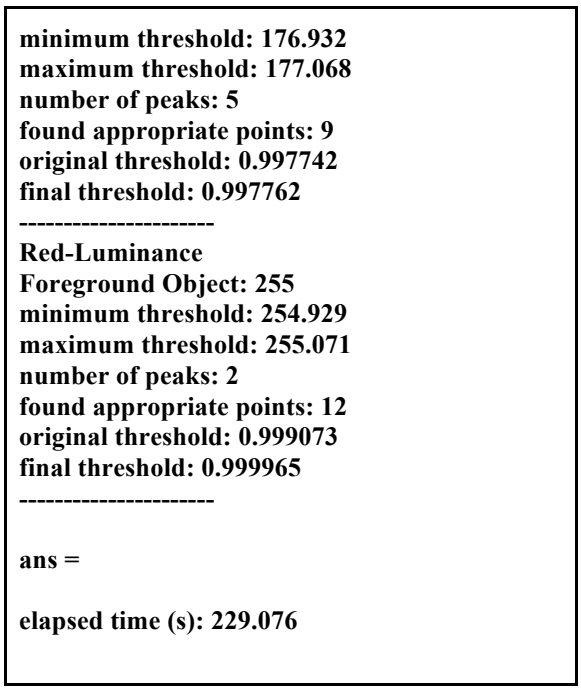

Figure 20: An example output for the analysis of the image in Figure 2. As can be noted, each property is modeled with high precision in the end as assessed by the Euclidean distance. The locations of foreground objects are also accurately shown.

Each image put through the ImageNEST system was chosen because it represented a test case scenario. The black circle represented an image that held only shape information and no color information. The green triangle image was chosen to represent a test case for color.

Conclusion

Nested sampling can be used to determine whether two images are the same. In our rudimentary examples, it was demonstrated that using nested sampling, the Gaussian composition of the histogram of the colors and luminosity of an image could be determined. Using these Gaussians, the overall characteristics of an image could be determined. From these determined properties, images could be compared.

Using our basic examples, it was unnecessary to further develop statistical methods to compare the images. To be able to determine the similarities in the images, the histogram distance was used. However, in the case that images are similar and the exact degree to which they are similar is desired, Bayes' theorem could be employed using the histogram distances to quantify the overall probability that the images are the same or their degree of similarity. This technique could be used when a database of images is employed. Furthermore, other properties can be extracted from the nested sampling generated GMMs, such as foreground-background characterization.

This ImageNEST algorithm could be used to find images in a database. The creation of such a database could compare an image to the database to find similar image to one that a user would input based on statistical comparison of the generated peak-histogram matrix for said

image and the ones existing the database. In addition, if such a database were to be created, Bayes' theorem could also be used to determine the degree of similarity between the user inputted image and the images from the database (in terms of the GMMs for each).

This project also lends itself to the possibility of analyzing videos. The video would be broken into a series of frames and the same algorithm as for still images would be used on each frame. The major drawback of this process involves the time that it takes to analyze each image and therefore each frame. When the software execution time was recorded, it took an average of 45.21 seconds (over 10 runs) to compute the Gaussians of the histograms for the Y'UV and shape for the black circle image. However, in order to compute the Gaussian for more complicated images, such as a sunrise, took on the order of 320 seconds. Therefore, it would take approximately that long to compute the histograms for each frame in a relatively visually complicated video as normal use would require. Using this system for image processing tends to be extremely computationally expensive and is therefore impractical for video processing. This system has an $O(n^n)$ complexity where n is the number of peaks for each histogram. In general, several high dimensionality probability spaces are generated when modeling the Gaussian mixture models of an image. The lack of real time processing could severely impede the usability of such an algorithm, especially in the video-processing domain. In essence, this work attempts to achieve many of the same things that Cristani and Murino (2007) deal with using S-TAPPMOG by modeling an image with Gaussian mixture models, then assigning pixels within 2.5 sigma of the most significant Gaussian part of the foreground. This is achieved by both systems, except ImageNEST is too computationally expensive to have practical application on video analysis.

References

Cristani, M., and Murino, V.: *A special mechanism for effective background subtraction*. In: Computer Vision Theory and Applications, pp. 403-412 (2007).

Nielsen, F. k-MLE: *A fast algorithm for learning statistical mixture models*. In ArXiv e-prints pp.1-31. 2012

Reynolds, D. *Gaussian Mixture Models*. 2008.

Sivia, D. S., and J. Skilling. *Data Analysis: A Bayesian Tutorial*. Oxford: Oxford UP, 2006. Print.