Optimum Global Thresholding Using **Otsu's Method**

Michael Metz
Illinois State University
Normal, IL 61761
msmetz@ilstu.edu

ABSTRACT
Thresholding is a technique in image processing that reduces an image into a binary image. Many algorithms use binary images for how simple they are to process. The quality of the result is dependent on the threshold value you select. Otsu's method uses statistical analysis of an images histogram to determine the optimal threshold value. Otsu's method performs best when the image satisfies a bimodal histogram. Intelligent preprocessing of an image can help satisfy this constraint. The standard implementation of Otsu's method is straightforward and can be done in linear time.

Keywords
Image Processing, Image Segmentation, Thresholding

1. INTRODUCTION
Image segmentation is the concept of dividing a digital image into multiple sections. This is done with the intention of making it easier to view and analyze its parts. Thresholding is one of the simplest forms of image segmentation. With this, an image composed of many intensity's (shades of gray) is reduced to a smaller amount of intensity's. Typically, thresholding reduces an image to a binary image. That is the intensities of each pixel can either be black or white. The algorithm for this is trivial, pick an intensity to serve as the threshold intensity. Then change any pixel under the threshold to black, and any pixel above the threshold to white. The quality of the results is dependent on the threshold value you select. A simple yet inefficient approach is to guess and visually check every value. Otsu's method[1] provides an intuitive way to automatically select a threshold based on statistics of the images histogram.

The structure of the paper is as follows, First, we introduce the notation that is used to denote a digital image. Second, we discuss the simple function of global thresholding. Third we introduce Otsu's method a provide a pseudocode implementation. Then, we perform Otsu's method on a few images, and discuss the results. Finally, we discuss some of the improvements that have emerged from the original algorithm.

2. Background
A digital image is represented as a two-dimensional array of pixels. Each pixel has an integer intensity that can range 0 – (L-1). L is the maximum number a pixel can represent. Otsu's method is meant for grayscale images, so for this paper we consider L to be 256. 0 being pure black and 255 being pure white, with the integer values between representing the gradual increase in shade. We say the function $f(x, y)$ returns the intensity value of the pixel at the x'th row and y'th column. For example, $f(0,0)$ returns the intensity value of the northeast most pixel. A digital images histogram shows the sum of pixels of a given intensity, see figure 1.1, 2.1, and 3.1

3. Thresholding
3.1 Global Thresholding
Global thresholding is one of the simplest forms of image segmentation. An image composed of many intensity's, (shades of gray) is reduced to a smaller amount of intensity's. Consider the image $f(x, y)$ and its threshold function,

$$g(x,y) = \begin{cases} 0 & if\ f(x,y) \le T \\ 1 & if\ f(x,y) > T \end{cases} \quad where\ 0 \le T \le L-1$$

The image $g(x, y)$ is considered a binary image since every pixel has an intensity of either 0 or 1. Remember *T* is the intensity where you draw the line for changing the pixels from black to white. With global thresholding a value for T must be decided on beforehand and remain constant.

3.2 Otsu's Method
The quality of thresholding is dependent on the selection of the threshold intensity. Otsu's method determines the optimal threshold intensity based on the images histogram. This works best with histograms that follow a bimodal distribution. That is there are two distinct peaks or classes (see figure 1.1 and 3.1). Otsu's method minimizes the weighted within-class variance[1]. That is, it computes the function $\sigma_w^2(t)$ for every intensity value in the image. The intensity value that results in the smallest $\sigma_w^2(t)$ is the best value to use for the threshold function.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

To calculate the equation above you must first build a normalized histogram. The normalized histogram is represented with the probability function $P(i)$. This function returns the percent of pixels that are of intensity *i*. In other words, $P(i) = \frac{sum\ of\ pixels\ with\ intensity\ i}{total\ pixels\ in\ image}$. This function is used to calculate the weights of each class. The left class is all the intensity values from 0 to *t* and the right class is all the intensity values from *t+1* to *L-1*.

$$q_1(t) = \sum_{i=0}^{t} P(i) \qquad q_2(t) = \sum_{t=t+1}^{L-1} P(i)$$

Next, we must calculate the means of each class. Notice that the probability function and the weight function are used to calculate the mean. Therefore, you must calculate those before calculating the means.

$$\mu_1(t) = \sum_{i=0}^{t} \frac{i * P(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{t=t+1}^{L-1} \frac{i * P(i)}{q_2(t)}$$

Finally, we calculate the variance for each class. This is straightforward since we already calculated the weights and means.

$$\sigma_1^2(t) = \sum_{i=0}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \qquad \sigma_2^2(t) = \sum_{t=t+1}^{L-1} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

3.2.1 Pseudocode

The following code is considering a 8 bit grayscale image. Therefore, a pixel can range from 0 – 255 in intensity. This implementation is done in linear time O(N). For a c++ implementation of this pseudocode you can visit https://github.com/Michael-Metz/image-processing

```
Otsu(image)
  //build histogram and normalized histogram (probabilities)
  histogram[256]
  for i = 0 to image.totalpixels
        intensity = image.getPixel (i)
        histogram[intensity] = histogram[intensity] + +
  probability[256]
  for i = 0 to 255
        probability[i] = histogram[i] / image.totalPixels()

  //loop over all possible Threshold values and find min variance
  minThreshold
  minVariance
  for t = 0 to 255
    //calculate class 1 and class 2 weights
    c1Weight = c2Weight = 0
    for i = 0 to t
        c1Weight += probility[i]
    for i = t +1 to 255
        c2Weight += probility[i]

   //calculate class 1 and class 2 means
    c1Mean = c2mean = 0
    for i = 0 to t
        c1Mean += (i * probility[i]) / c1Weight
    for i = t +1 to 255
        c2Weight += p(i * probility[i]) / c2Weight

   //calculate class 1 and class 2 variances
    c1Var = c2Var = 0
    for i = 0 to t
        c1Var += (i – c1Mean)^2*(probability[1]/c1Weight)
    for i = t +1 to 255
        c2Var += (i – c2Mean)^2*(probability[1]/c2Weight)
    //compute the weighted within class variance
   weightedVar = c1Weight * c1Vari + c2Weight * c2Var
   if(weightedVar < minVariance)
      minVariance = weightedVar
        minThreshold = t
  end for
return minThreshold
```

4. Experiments

The implementation of Otsu's method was done in c++ using the 2011 standard. The code was compiled with g++ and ran with the Mac OSX platform. The algorithm was executed on single thread using a quad core 2.7 GHZ processer with 16 Gigabytes of RAM. One scene was used, with three different scenarios. The first being a properly exposed "clean" image. Second, A properly exposed image, that is ridden with "noise". Third, the same "noisy" image but with an averaging filter applied thus smoothing out the noise. To the right of each image scenario we have the binary image obtained from performing Otsu's method, and below we have the histogram representation of the original image.

4.1.1 Clean image

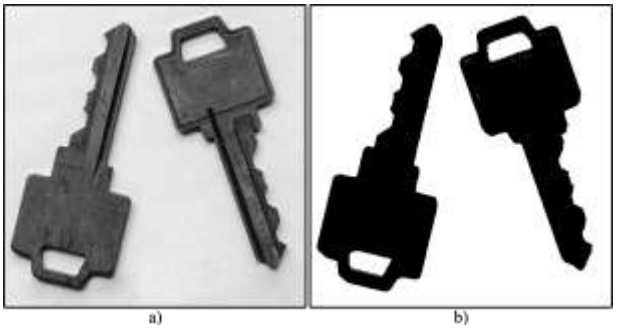

Figure 1. a) grayscale image b) binary image using Otsu's method

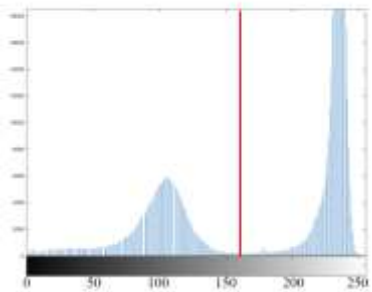

Figure 1.1. bimodal histogram for figure 1 a) Otsu's method selected intensity level 163 (red line)

4.1.2 Noisy Image

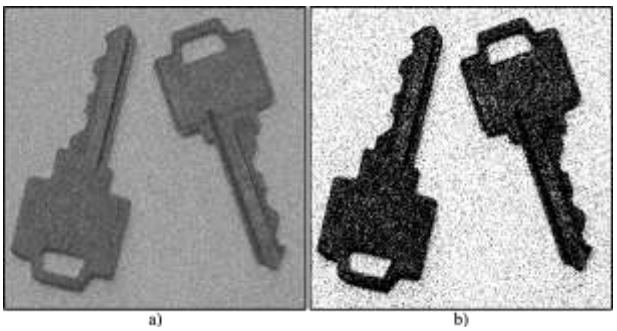

Figure 2. a) grayscale image with noise b) binary image using Otsu method

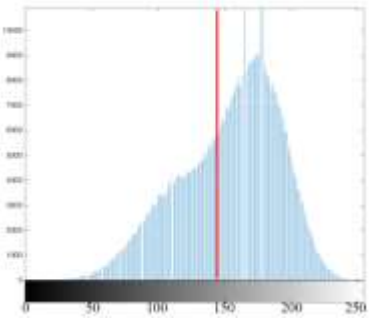

Figure 2.1 Left Skewed histogram for figure 2 a) Otsu's method selected intensity level 145 (red line)

4.1.3 Noisy Image with Averaging

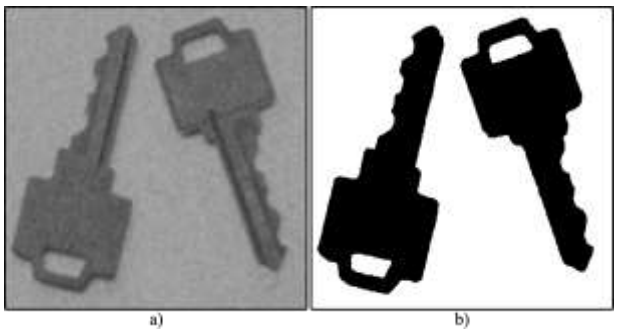

Figure 3.1 Bimodal histogram for figure 3 a) Otsu's method selected intensity level 145 (red line)

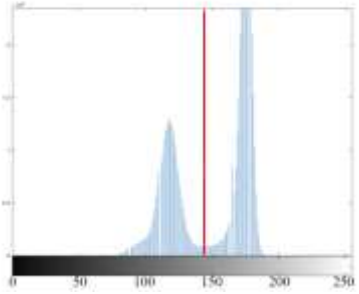

Figure 3. a) grayscale image with noise but averaged b) binary image using Otsu method

5. Analysis

Figure 2 had very poor results with Otsu's method. You can see noise has also been segmented from the original image. This is due to the source image being ridden from noise which caused its histogram(figure 2.1) to become left skewed. Otsu's method uses statistics to exploit bimodal histograms (figure 1.1 and 2.1). But even though we don't have an image with a bi-modal histogram we can try to make it into one. Figure 3 does exactly that by applying an averaging filter to the same image used in Figure 2. Then using Otsu's method on the averaged image.

6. Improvements

Further investigation shows Otsu's method can be performed by maximizing the between class variance. In the pseudocode we preformed Otsu's method by minimizing the weighted within class variance.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

That is for every intensity level from 0 to L we calculate $\sigma_w^2(t)$ and the intensity level that results in the minimum value of $\sigma_w^2(t)$ is the best level. Now with some algebra we find that the equation for total variance is

$$\sigma^2(t) = \sigma_w^2(t) + q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2$$

The total variance of an image $\sigma^2(t)$ will remain constant for all values of t. Examining the equation we see that total variance is just the sum of *the weighted within class variance* $\sigma_w^2(t)$ and the *between class variance* $q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2$. Using the original method, we need to find the value for *t* that minimizes $\sigma_w^2(t)$. Therefore this value of *t* will yield the maximum value of $q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2$. Thus, another approach to Otsu's method is to find the value for *t* that yields the maximum value for the *between class variance.* This approach can be done recursively.

The clear limitation of Otsu's method is that the image must have a bi-modal histogram. A paper published in 2012 improves this limitation by slightly tweaking Otsu's method[2]. The paper proposes to use the median in place of mean, when it comes to calculating the average for each class. With this modification, they noticed improved results for images with skewed histogram distributions (such as figure 2.a). In the experiment we demonstrated that images with noise can be preprocessed using an average filter. This operation can return a skewed histogram back to a bi-modal state. With this we can perform the original mean based Otsu's method. The downside is that the averaging filter made the image softer and the softness is noticeable in the binary image. The improved median based Otsu's method can be done on the original image with the skewed histogram. Thus preserving the sharpness of the original image.

7. Conclusion

Otsu's method allowed us to determine the ideal threshold value to segment foreground objects from an image. The algorithm works best with bi-modal histograms, but there are many techniques to get around this limitation. Finally, Otsu's method is straightforward to implement in linear time; therefore, it deserves a spot in any image processing library.

8. ACKNOWLEDGMENTS

Thanks to ACM SIGCHI for allowing us to modify templates they had developed.

9. REFERENCES

[1] Otsu, Nobuyuki. (1979). A Threshold Selection Method from Gray-Level Histograms. Systems, Man and Cybernetics, IEEE Transactions on. 9. 62-66.

[2] Yang, Xiaolu & Xuanjing, Shen & Long, Jianwu & Chen, Haipeng. (2012). An Improved Median-based Otsu Image Thresholding Algorithm. AASRI Procedia. 3. 468–473. 10.1016/j.aasri.2012.11.074.