ITK Implementation Of The Minimum Error Image Thresholding Algorithm

Release 1.00

Yousef A. Al-Kofahi

November 30, 2008

Rensselaer Polytechnic Institute (RPI)

Troy, NY, USA

Abstract

An implementation of the minimum error thresholding algorithm is presented. In this implementation, two versions of the minimum error thresholding algorithm are made available, one utilizing a Gaussian- and the other utilizing a Poisson- mixture model. Two ITK classes are created: The first one (`MinErrorThresholdImageCalculator`) is used to compute the threshold; the second (`MinErrorThresholdImageFilter`) is used as an image-thresholding filter. The results indicate code correctness and are comparable to those of Otsu's thresholding algorithm.

Contents

1 Introduction

Image binarization is used as the first step in several image segmentation tasks. Thresholding is considered as the simplest method for image binarization. Several image thresholding techniques have been proposed in the literature. Examples include histogram-, clustering- and entropy-based methods. A survey on image thresholding methods can be found in [1].

In this article, an implementation of a clustering-based method, called minimum error thresholding, is presented. Assuming that the image histogram is bi-modal, a binarization threshold is computed based on fitting a mixture of two distributions. The original minimum error thresholding algorithm was introduced by Kittler *et al.*[2] where the histogram was modeled as a mixture of two Gaussians. However, based on the theory of formation of an ideal image [3], it was shown in [4] that such a histogram can be more accurately modeled by a mixture of two Poisson distributions.

Based on the thresholding results of several 2-D and 3-D images, the Poisson mixture model provides better results than the Guassian mixture model. Hence, it is used as the default choice in the presented implementation. In addition, the minimum error thresholding results are comparable to those obtained by the ITK implementation of the famous Otsu's thresholding algorithm [5].

2 Threshold Computation

The normalized image histogram represents an estimate of a mixture of two distributions, and is given as:

$$p(g) = \sum_{j=0}^{1} P_j \, p(g \mid j)$$

where g is an intensity value in the range $\{0, ..., g_{\max}\}$, and P_j is the prior probability of the j^{th} component of the mixture; i.e., $p(g \mid j)$. This is assumed to be either a Poisson probability distribution (default) or a Gaussian probability distribution as follows:

Poisson Case (default): $p(g \mid j) = \dfrac{{\mu_j}^g}{g!} e^{-\mu_j}$

Gaussian Case : $p(g \mid j) = \dfrac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(g-\mu_j)^2}{2{\sigma_j}^2}}$

Given a threshold t, the mixture parameters are defined as follows:

$$P_j(t) = \sum_{g=a}^{b} p(g), \qquad \mu_j(t) = \frac{\sum_{g=a}^{b} g \; p(g)}{P_j(t)}, \qquad \sigma_j^2(t) = \frac{\sum_{g=a}^{b} p(g)\left(g - \mu_j(t)\right)^2}{P_j(t)},$$

$$\mu(t) = P_0(t)\mu_0(t) + P_1(t)\mu_1(t)$$

where $a = \begin{cases} 0, & \text{j=0} \\ t, & \text{j=1} \end{cases}$ and $b = \begin{cases} t+1, & \text{j=0} \\ g_{\max}, & \text{j=1} \end{cases}$

For both distributions μ_j is the j^{th} mixture component mean. On the other hand, σ_j is the standard deviation in the Gaussian case. Finally, μ is the total mean of the mixture.

The goal is to find the threshold value t^* that minimizes an error criterion function given as:

$$\text{Poisson Case: } t^* = \arg\min_t \left\{ \mu - P_0(t)\left(\ln P_0(t) + \mu_0(t)\ln \mu_0(t)\right) - P_1(t)\left(\ln P_1(t) + \mu_1(t)\ln \mu_1(t)\right)\right\}$$

$$\text{Gaussian Case: } t^* = \arg\min_t \left\{1 + 2[P_0(t)\ln \sigma_0(t) + P_1(t)\ln \sigma_1(t)] - 2[P_0(t)\ln P_0(t) + P_1(t)\ln P_1(t)]\right\}$$

3 Implementation

The work presented in this article is implemented in two classes: `MinErrorThresholdImageCalculator` and `MinErrorThresholdImageFilter`. The former, derived from class `Object`, is used to compute the threshold and to estimate the mixture distribution parameters. The second class, derived from class `ImageToImageFilter`, takes an ITK image as an input and uses `MinErrorThresholdImageCalculator` to compute the threshold. This is illustrated in the diagram shown in Fig. 1.

In addition to the input image, the algorithm requires two input parameters. The first one is the number of histogram bin (default is 128) and the second one is the mixture type (default is a Poisson mixture). If these two parameters are not provided, then the default values will be used.

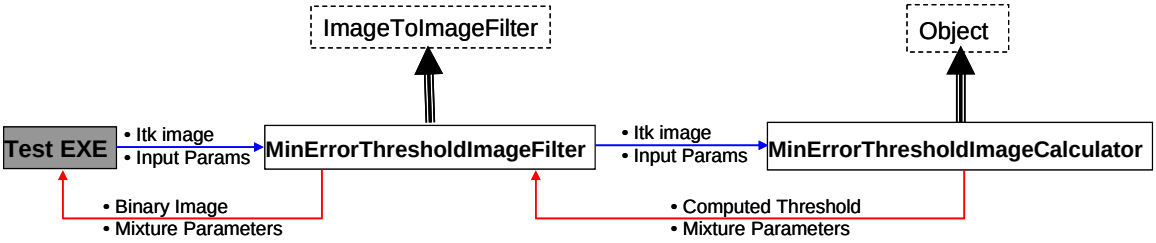

Figure 1: A diagram showing the relationships and the data flow between the different classes. Blue arrows represent inputs and red arrows represent outputs.

4 Code Example

The next code example illustrates how to use the minimum error thresholding filter. The code shown below is also available in the test source file (*itkMinErrorThresholdImageFilterTest.cxx*) included with the submitted source code.

```
#include "itkMinErrorThresholdImageFilter.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"


int main(int argc, char* argv[] )
{
  if( argc < 5 )
    {
    std::cerr << "Usage: " << argv[0];
    std::cerr << " inputImageFile outputImageFile numberOfHistogramBins ";
    std::cerr << "mixtureType <1=Gaussian, 2=Poisson>";
    std::cerr << std::endl;
    return EXIT_FAILURE;
    }

  //Input and Output pixel types
  typedef  short  InputPixelType;
  typedef  unsigned char  OutputPixelType;

  //Input and output images
  typedef itk::Image< InputPixelType,  3 >   InputImageType;
  typedef itk::Image< OutputPixelType, 3 >   OutputImageType;

  //The binarization filter
  typedef itk::MinErrorThresholdImageFilter<
              InputImageType, OutputImageType >  FilterType;

  //Image reader/writer
  typedef itk::ImageFileReader< InputImageType >  ReaderType;
  typedef itk::ImageFileWriter< OutputImageType >  WriterType;

  //creat the filter, the reader and the writer
  ReaderType::Pointer reader = ReaderType::New();
  FilterType::Pointer filter = FilterType::New();
  WriterType::Pointer writer = WriterType::New();

  //The main pipeline
  reader->SetFileName( argv[1] );
  filter->SetInput( reader->GetOutput() );
  filter->SetNumberOfHistogramBins ( atoi(argv[3]) );
  filter->SetOutsideValue( 255 );
  filter->SetInsideValue(  0  );
  filter->SetMixtureType( atoi(argv[4]) );
  writer->SetInput( filter->GetOutput() );
  writer->SetFileName( argv[2] );

  //updating the writer triggers the update on the previous filters in the pipeline
  writer->Update();


  //Print the resulting threshold and the estimated mixture parameters
  std::cout << "Computed Threshold is: " << filter->GetThreshold() << std::endl;
  std::cout << "Estimated Mixture Parameters are: " << std::endl;
  std::cout << "Background Mean = " << filter->GetAlphaLeft() << std::endl;
  std::cout << "Foreground Mean = " << filter->GetAlphaRight() << std::endl;
  std::cout << "Background Prior = " << filter->GetPriorLeft() << std::endl;
  std::cout << "Foreground Prior = " << filter->GetPriorRight() << std::endl;

 return EXIT_SUCCESS;

}
```

5 Results

The code was tested on several 2-D and 3-D images using different parameters. Fig2. shows a 2-D test image and the resulting binary images using both Poisson and Gaussian mixtures with 128 histogram bins. Fig. 3 shows a comparison between the output of the minimum error thresholding algorithm (using a Poisson mixture) and the output of Otsu's thresholding algorithm (using itkOtsuThresholdImageFilter).

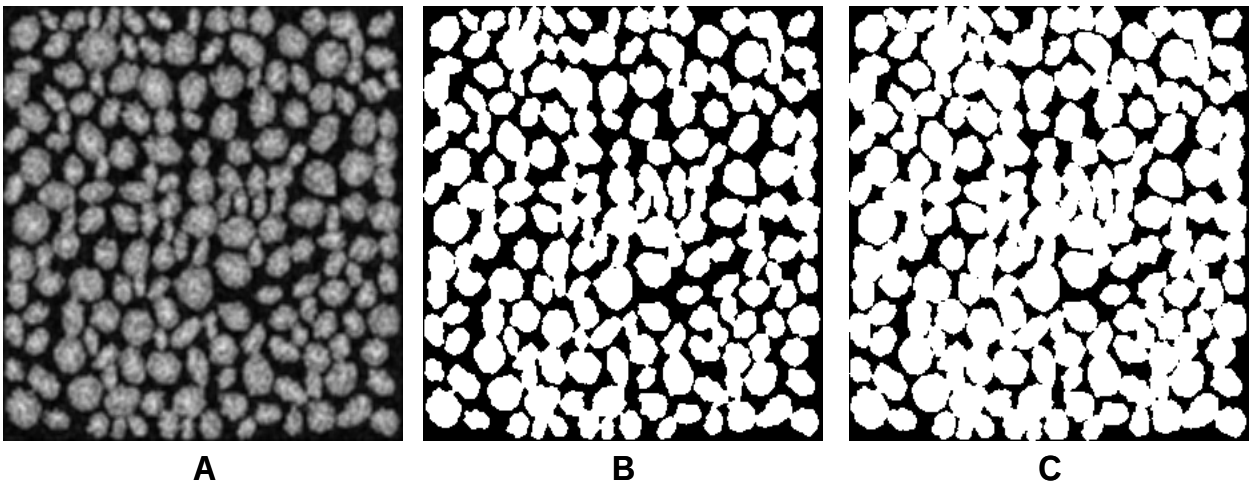

Figure 2: (A) 2-D test image. (B)Thresholded image using a Poisson mixture.
(C) Thresholded image using A Gaussian mixture

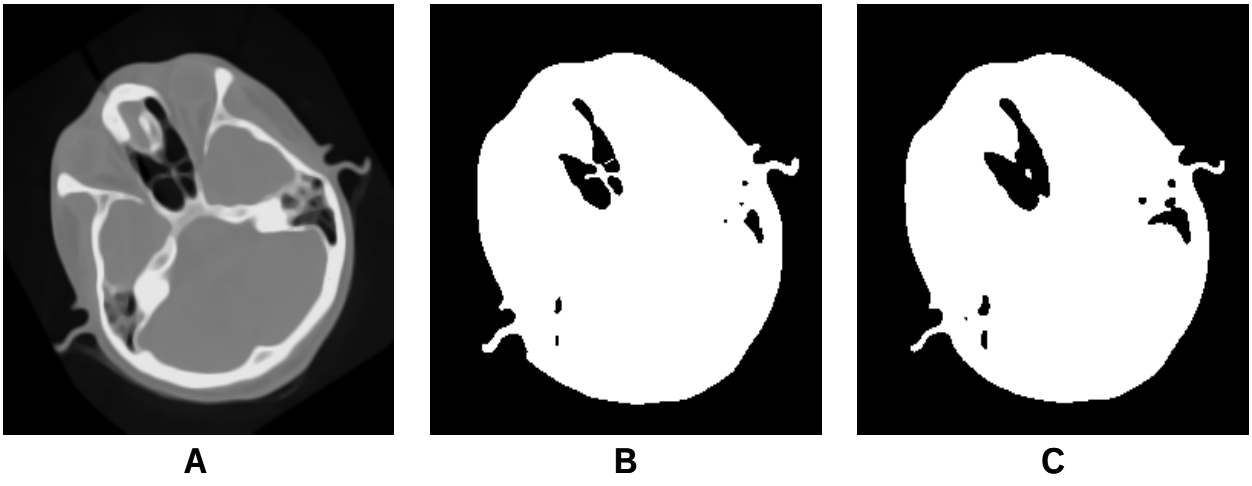

Figure 3: (A) 2-D test image. (B)Thresholded image using a Poisson mixture Minimum Error Thresholding method. (C) Thresholded image using Otsu's Thresholding method

Reference

[1] Mehmet Sezgin and Bulent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*. 13: 146-165, 2004

[2] J. Kittler and J. Illingworth. Minimum error thresholding. Pattern Recogn. 19: 41-47, 1986

[3] Dainty J.C., and Show R. Image Science. Academic Press, New York, 1974

[4] Pal N.R., and Pal S.K., 1991. Image model, Poisson distribution and object extraction. *International Journal of Pattern Recognition Artificial Intelligence*, 5: 439–483.

[5] N. Otsu. A threshold selection method from gray level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1): 62-66, 1979