An Adaptive Thresholding Image Filter

Release 0.00

Kishore Mosaliganti, Arnaud Gelas and Sean Megason

October 23, 2009

Department of Systems Biology, Harvard Medical School, Boston, MA-02139, USA

Abstract

An Insight Toolkit (ITK) algorithm for adaptively thresholding images is presented in this paper. Currently, the usage of thresholding methods in ITK has made use of global thresholds, confidence connected thresholds and neighborhood strategies. The current work extends these family of filters by setting thresholds adaptively in local image regions. The user is not required to specify seed regions apriori which greatly eases the task of automatic segmentation. The thresholds are determined using Otsu's minimization of between-class variances in local image regions that are selected randomly throughout the domain. Using non-uniformly sampled thresholds, a continuous function is reconstructed throughout the image domain using a B-Spline approximation algorithm [1]. Hence, the image domain is adaptively sampled by making use of the reconstructed threshold function. Most imaging modalities introduce some intensity inhomogeneities that can be recovered by this method. We include 2D example code, parameter settings and show the results generated on embryonic images of the zebrafish from optical microscopy.

Contents

1 Introduction **2**

2 Implementation **2**

3 Usage **3**

4 Results **4**

1 Introduction

In image analysis, we are often interested in extracting the foreground object of interest as a preliminary step to more advanced processing operations. Therefore, a global threshold is often applied. However, most imaging modalities introduce some intensity inhomogeneities in the images for a variety of reasons. For example, in fluorescent microscopy, nuclei often exhibit varying intensities depending on the amount of the marker they contain. In light microscopy, there are spatial gradients in the image as a resut of the camera calibration across the focal plane. Similarly, due to lighting variations, there may be an illumination gradient. In all these cases, applying a global threshold to each pixel will most likely yield a poor result. Therefore, there is considerable interest in applying a threshold using local intensity fields.

In the current ITK framework, there are several thresholding filters namely, `itkBinaryThresholdImageFilter`, `itkOtsuThresholdImageFilter`, `itkRobustAutomaticThresholdImageFilter` and `itkKappaSigmaThresholdImageFilter` etc. These filters determine intelligent thresholds but globally. As a result, they are inadequate to our needs. Another class of thresholding filters namely, `itkNeighborhoodConnectedImageFilter` and `itkConfidenceConnectedImageFilter` operate in local regions but require the specification of seeds. In our current submission, we introduce a new filter `itkAdaptiveOtsuThresholdingImageFilter` that solves the above problems. We are able to apply intelligent thresholds determined from local regions that are either user-specified or selected randomly. The threshold function is them smoothly interpolated throughout the image domain to determine local thresholds for each pixel.

2 Implementation

This filter derives from the base class `itkImageToImageFilter`. The input consists of an intensity image and the filter outputs a binarized image. Adaptive thresholds determined for all pixels are also output as a separate threshold image. In the default setting, a randomized image iterator walks through the image domain and picks up random points to determine thresholds. At all other points of the image domain, the threshold value is smoothly interpolated using a B-Spline approximation method [1] via the filter `itkBSplineScatteredDataApproximationImageFilter`. The B-Spline methods work in a heirarchical manner across multiple levels. As a result, the user can specify the number of levels, the number of control points and the spline order to be used. We now describe each of the parameters, their range and typical values. The filter utilizes the process of Otsu thresholding in `itkOtsuThresholdImageCalculator` that minimizes the within class variance and maximizes the in-between class variance. The user also has the option of specifying the sample points for computing Otsu thresholds.

- `m_InsideValue` - Pixels inside the foreground object are set to this value. Usually set as 1.
- `m_OutsideValue` - Pixels inside the foreground object are set to this value. Usually, set to 0.
- `m_NumberOfHistogramBins` - The number of histogram bins to collect intensity data and determine the Otsu bimodal threshold. In an 8-bit image, it is usually set to the maximum of 255 bins.
- `m_NumberOfControlPoints` - The number of control points to be used in the base level of B-Spline approximation of the threshold surface. This depends on the number of samples being used, the topology of the intensity inhomogeneity and the nature of the fit being considered. Usually set as $10-50$.

- `m_NumberOfLevels` - The number of levels of spline reconstruction refinement to be used. Usually, set in the range $3-5$ for large datasets.
- `m_NumberOfSamples` - The number of random samples to collect at various points in the image. A larger number increases the running time proportionately and leads to better accuracy.
- `m_Radius` - The radius of the neighborhood to compute local histograms for Otsu thresholding.

Our implementation makes use of CMake 2.6 version for compilation and has been tested using ITK 3.16 release.

3 Usage

We begin by including the appropriate header files for the filter.

```
#include "itkAdaptiveOtsuThresholdImageFilter.h"

...
int main(int argc, char *argv[])
{
  ...
  const     unsigned int    Dimension = 2;
  typedef unsigned char PixelType;
  typedef itk::Image< PixelType, Dimension > ImageType;

  typedef itk::ImageFileReader< ImageType > ReaderType;
  typedef itk::ImageFileWriter< ImageType > WriterType;
```

The following typedef for the filter is required.

```
  typedef itk::AdaptiveOtsuThresholdImageFilter< ImageType, ImageType > FilterType;
```

After initialization, the user may specify settings for the number of samples and parameters of the spline reconstruction of the threshold function. All parameters have default settings that can be adjusted optionally.

```
  FilterType::Pointer filter = FilterType::New();
  filter->SetInput( reader->GetOutput() );
  filter->SetInsideValue( 1 );
  filter->SetOutsideValue( 0 );
  filter->SetNumberOfHistogramBins( 256 );
  filter->SetNumberOfControlPoints( atoi( argv[6] ) );
  filter->SetNumberOfLevels( atoi( argv[5] ) );
  filter->SetNumberOfSamples( atoi( argv[4] ) );
  filter->SetRadius( m_radius );
  filter->Update();
```

The output consists of the adaptively threshold image. The exact thresholding function can be recovered by calling GetThresholdImage() which usually provides an idea of the intensity inhomogeneity.

```
  output = filter->GetOutput();
  thresholdImage = filter->GetThresholdImage();
```

4 Results

The results in this example can be obtained by using `AdaptiveOtsuThresholdImageFilter2D.cxx` on the input image `input.png`. In this example, nuclei of zebrafish ear cells are detected in a microscopy image. The image constains different tissue types (ear and hind brain) and the nuclei have differing intensities. Hence, this image serves as a good example of the robustness of our filter. The output consisting of the nuclei foreground written out to the image `output.png`. The parameters to the filter are set at the command line to facilitate easy modification and exploration by the user. The command to run this particular executable is as follows:

```
Usage: ./adaptiveOtsuThresh2D
 inputImage
 outputImage
 radius
 numOfSamples
 numOfLevels
 numOfControlPts

./adaptiveOtsuThresh input.png output.png 20 5000 3 30
```

The results of the thresholding are shown graphically in Figure 1.

References

[1] N. J. Tustison and J. C. Gee. *n-d c^k* b-spline scattered data approximation. *The Insight Journal*, 2006.

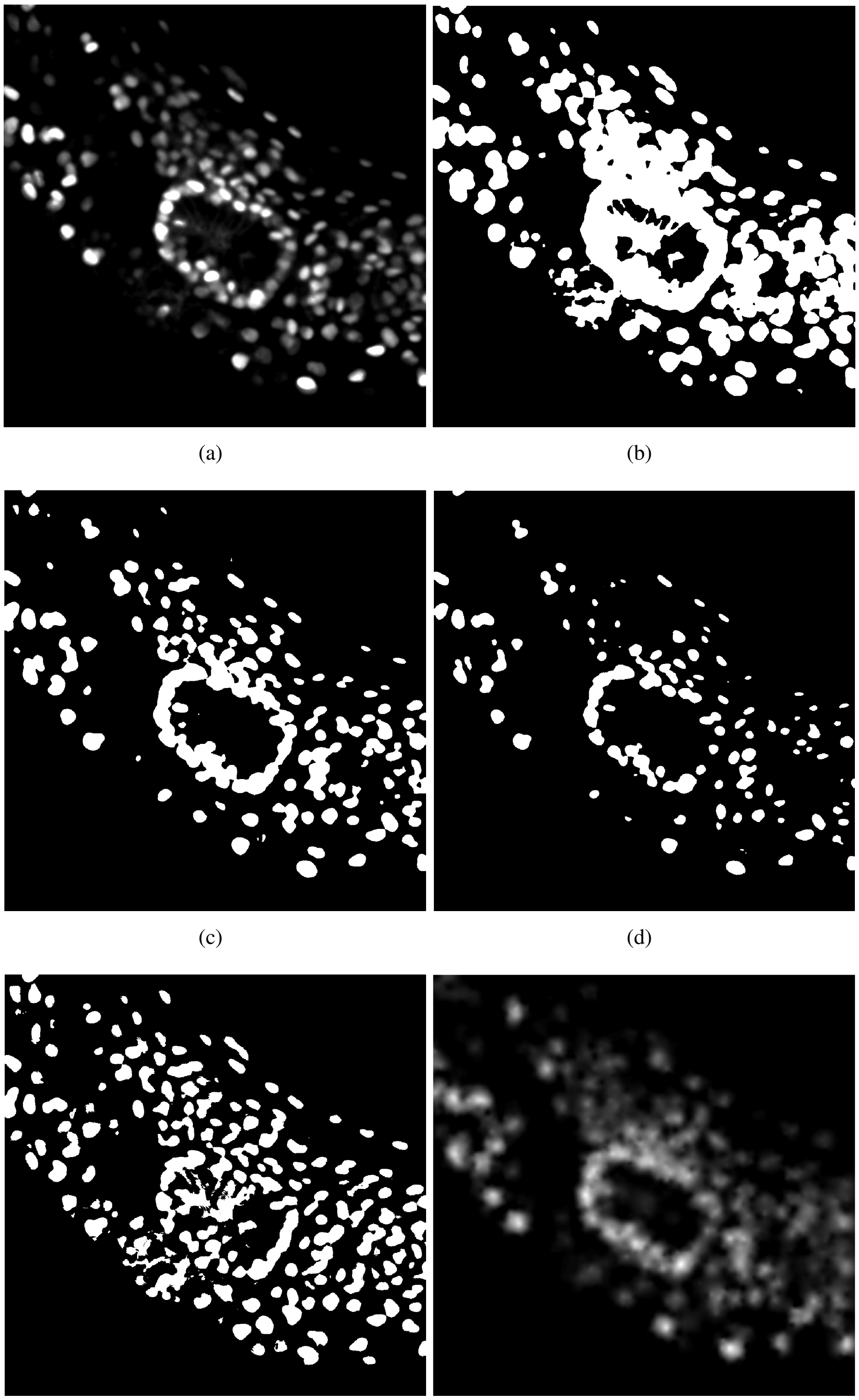

Figure 1: Adaptive thresholding example on a zebrafish nuclei image ($1K \times 1K$): (a) Input image, (b) Global threshold at: 10 (c) 100 (d) 200 (e) Adaptive threshold binary image (f) Reconstructed threshold function