Region Growing Using a Criterion on the Region Boundary

Release 0.00

David Doria

August 7, 2009

Rensselaer Polytechnic Institute, Troy NY

Abstract

This document presents a set of classes which implement a region growing algorithm whose criterion for growth is different than existing ITK region growing algorithms. The region is grown into the query pixel based only on the difference between the query pixel and neighboring pixels that are already members of the region. This allows the region to be grown along paths that change gradually over time and end up looking nothing like they originally did. This is usefully in many situation, including growing surface patches around bends using growth criteria on the normal vectors. The usage and structure of these classes follow the style of itkConnectedThresholdImageFilter. We propose these classes as addition to the Insight Toolkit ITK `www.itk.org`.

Contents

1 Introduction

The Insight Toolkit currently provides two region growing methods. The first is a very non-dynamic growth criterion which makes the user specify a range ([low, high]) for all pixels in the region to fall into. The second growth criterion (itkConfidenceConnectedImageFilter) uses statistics of the entire region that has been grown so far to make a decision about the next pixels membership. These two methods are very standard and well suited to many image segmentation applications. However, it is often useful to grow a region based only on the query pixels immediate neighbors that currently belong to the region. That is, only using the boudary (or edge) of the region in the n-connected neighborhood of the query pixel. This set of classes implements this algorithm.

2 The New Image Function - itkRegionEdgeFunction

The Image Function is the class that the itkFloodFilledImageFunctionConditionalIterator uses to decide if a query pixel should be added to the region. The standard Image Function that is used with itkConnectedThresholdImageFilter is itkBinaryThresholdImageFunction. Again, in an attempt to follow as closely as possible to the existing framework, we model itkRegionEdgeFunction after itkBinaryThresholdImageFunction. The main difference is that we now need access to the OutputImage of the itkConnectedRegionEdgeThresholdImageFilter in the Image Function. This is because the Image Function needs to know if the query pixels neighbors are already in the region. Since itkRegionEdgeFunction is not a subclass of itkConnectedRegionEdgeThresholdImageFilter, we accomplish this by adding an additional template parameter to itkRegionEdgeFunction that defines the type of the outputImage. We can then pass a pointer to the output image using a new function `SetOutputImagePointer(outputImage);`.

In itkRegionEdgeFunction, we use an itkConstNeighborhoodIterator to check all of the query pixels neighbors. If any neighbor is already in the region and “close enough” to the query pixel, the query pixel is added to the region.

Algorithm 1 Algorithm to add a query pixel to the region

```
for all Neighbors of the query pixel do
  if The current neighbor is already in the region then
    if ((Neighbor - Lower) $\leq$ QueryPixel) && (Neighbor + Upper) $\geq$ QueryPixel ) then
      Add the query pixel to the region
    end if
  end if
end for
```

3 Parameters and Usage

To make usage simple, we follow the existing itkConnectedThresholdImageFilter class structure. The SetLower() and SetUpper() functions, however, now have different meaning. These values are the amounts that the query pixel is allowed to differ relative to its neighboring pixels already in the region. That is (like shown in Algorithm 1), if the value of the query pixel is between $(Neighbor - Lower)$ and $(Neighbor + Upper)$ for any of its neighbors that are already in the region, it should be added to the region.

4 Examples

4.1 Example - Region with Gradient

Choosing any seed pixel in the curved region will produce the following segmentation.

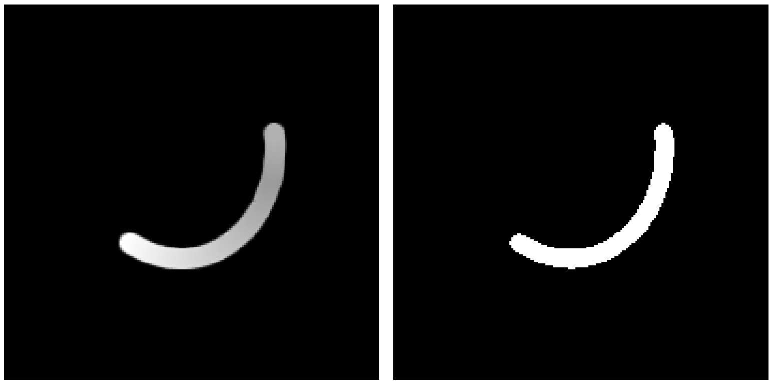
(a) Original Image (b) Segmented Image

Figure 1: An example in 2D.

4.2 Usage example

```
// define the filter
typedef itk::ConnectedRegionEdgeThresholdImageFilter < ImageType, ImageType> ConnectedFilterType;
ConnectedFilterType::Pointer connectedThreshold = ConnectedFilterType::New();

// ... define Lower and Upper (floats)
connectedThreshold->SetLower(Lower);
connectedThreshold->SetUpper(Upper);

connectedThreshold->SetReplaceValue(255);

//set the seed
ImageType::IndexType seed;
// ... define the seed location (unsigned int) ...
   seed[0] = Seed;

connectedThreshold->SetSeed(seed);

//setup pipeline
connectedThreshold->SetInput(image);
```