Viscous Normalization in the Insight Toolkit

Technical Report: Design, Specifications, and User's Guide

Version 1.0 for ITK 3.12

Nathaniel Tippens and Alexander Liu

May 19, 2011

With guidance from:

Alfredo Quinones-Hinojosa

Hadie Adams

Russell Taylor

Contents

Introduction

This code package is intended to implement the viscous watershed transform, as described by Meyer and Vachier [1], in ITK. In order to implement this rather complicated approach to image segmentation, three new filters have been developed. To describe the code and functionality breakdown briefly:

ViscousImageFilter:
The low-level adaptive morphological filter described in [1].

MosaicWatershedImageFilter:
Uses the linear-time Watershed algorithm described in [2] to efficiently compute the mosaic watershed image.

ViscousWatershedImageFilter:
A composite filter which streamlines the viscous-normalized watershed segmentation pipeline.

[1] Vachier, C. and F. Meyer (2005). "The Viscous Watershed Transform." Journal of Mathematical Imaging and Vision 22(2): 251-267. http://www.cmla.ens-cachan.fr/fileadmin/Membres/vachier/JMIV05.pdf

[2] Cousty, J., G. Bertrand, et al. (2009). "Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle." Pattern Analysis and Machine Intelligence, IEEE Transactions on 31(8): 1362-1374. http://www.esiee.fr/~info/a2si/Ps/flow_watershedPAMI.pdf

ViscousImageFilter

About This Filter

The ViscousImageFilter normalizes the input image according to the algorithm described by Meyer and Vachier [1]. This is an adaptive morphological filter, meaning that it performs different morphological operations depending on local image properties. Specifically, the viscous normalization applies morphological reconstruction using different radii structuring elements, where the radius used is a function of pixel intensity.

Input

This filter requires at least two inputs: an input image, and a normalization function Rho.

A Note About the Input Image

Just like the standard ITK Watershed algorithm, the input image to this filter *should* be a gradient image. If you want to use this filter on a greyscale image whose gradient is not a meaningful height function, use the separate ViscousImageFilter directly, followed by the standard MorphologicalImageFilter included with ITK. See the ViscousImageFilter documentation for an example.

Specifying Rho

The normalization function Rho is most thoroughly described in [1] where it was first named and described. Briefly, Rho defines the radius of closing to impose on Watershed regions, as a function of their gradient magnitudes. For most users, the precise definition does not need to be understood; an adequate understanding can be learned from the example below.

However, it is first important to understand *how* you must pass the Rho function to ViscousWatershedImageFilter. It is passed as an operator() of a struct, whose type must be passed as a template class to ViscousWatershedImageFilter. See the example below.

Consider the following example code. Note that the Rho function *must* be declared before even declaring a ViscousWatershedImageFilter, as it is required to define the filter type. After the type has been passed, you can declare the filter type and instantiate a filter object. Finally, you need to pass your Rho object to the filter object!

```
// declare a new type of struct, RhoFuncType.
struct RhoFuncType
  {
  int operator()( int gradient_magnitude )
      {
      int radius;
      if( gradient_magnitude  > 200)
        radius = 0;
      else if( gradient_magnitude  > 150)
        radius = 1;
      else if( gradient_magnitude  > 100)
        radius = 2;
      else if( gradient_magnitude  > 50)
        radius = 3;
      else:
        radius = 4;
      }
  // Note that RhoFunction is now an object
  // of type RhoFuncType.
  } RhoFunction;

// Now that we have RhoFuncType, we can define the
// filter type:
typedef
      ViscousImageFilter<ImgType, ImgType, RhoFuncType>
      ViscousType;

// Declare filter object
ViscousType::Pointer ViscousFilter = ViscousType::New();
ViscousFilter->SetInput( input );

// We still need to pass the Rho *object*!
ViscousFilter->SetRho( RhoFunction );
```

MosaicWatershedImageFilter

About This Filter

The MosaicWatershedImageFilter computes a Watershed segmentation using a linear-time graph-cut-based algorithm described in [2]. This algorithm has the added perk of running on the source image instead of a gradient image.

Input

This filter only requires an image as input. Note that the image should not be a gradient image: the mosaic image is constructed by computing the watershed on the gradient of the input image and then computing the mean for each resulting watershed region.

ViscousWatershedImageFilter

About This Filter

The ViscousWatershedImageFilter produces a normalized Watershed segmentation of the input image, according to the algorithm described by F Meyer and C Vachier [1]. This is a composite filter which calls low-level morphological filters in a meaningful order, although two of these low-level filters are entirely novel as well: MosaicWatershedImageFilter and ViscousImageFilter. Descriptions of these novel filters are included as independent documents.

Input

This filter requires at least two inputs: an input image, and a normalization function Rho. See the ViscousImageFilter documentation for details on the normalization function, since this filter must be typed and passed the function struct in the same way as ViscousImageFilter must be.

A Note About the Input Image

Unlike the standard ITK Watershed algorithm, the input image to this filter should *not* be a gradient image. This composite filter requires information from the original image, and computes the gradient as necessary during the filter pipeline. If you want to use this filter on a greyscale image whose gradient is not a meaningful height function, use the separate ViscousImageFilter directly, followed by the standard MorphologicalImageFilter included with ITK. See the ViscousImageFilter documentation for an example.