Well-Composed Image Filters for Repairing 2-D and 3-D Binary Images

Nicholas J. Tustison[1], Marcelo Siqueira[2], and James C. Gee[1]

August 30, 2006

[1]Penn Image Computing and Science Laboratory
University of Pennsylvania
and
[2]Departamento de Computação e Estatística
Universidade Federal de Mato Grosso do Sul

Abstract

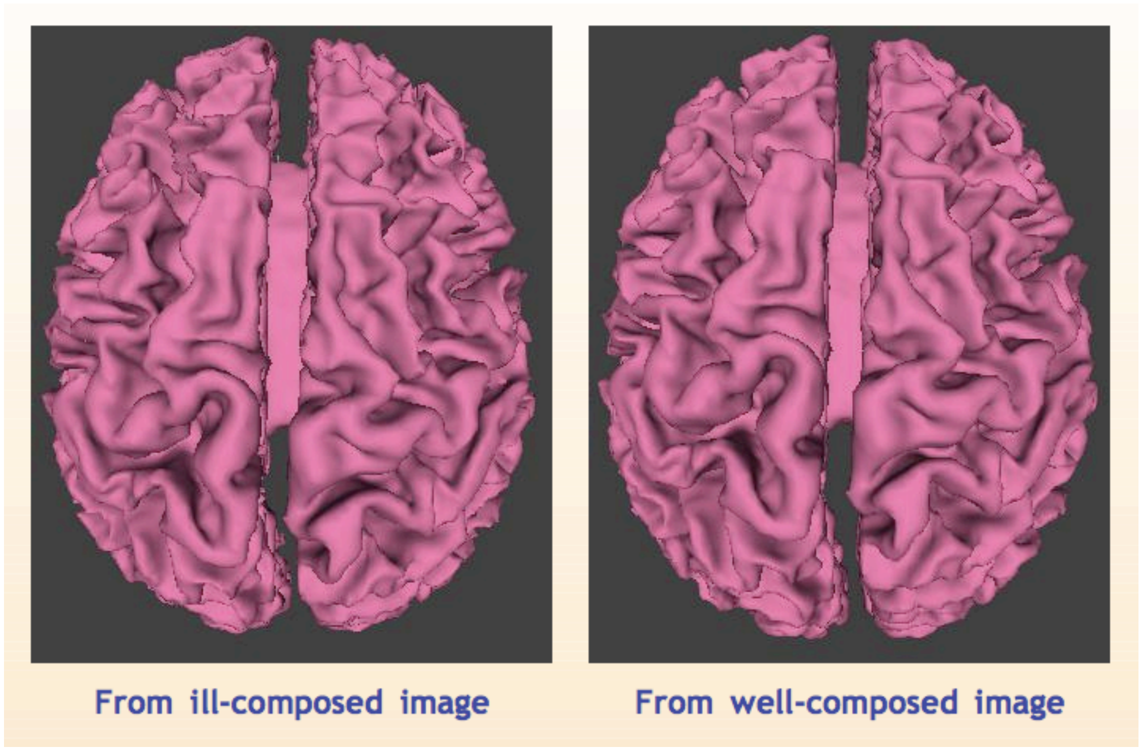
We consider 2-D and 3-D digital binary images characterized by their *well-composedness*. Well-composed images exhibit important topological and geometrical properties not shared by their ill-composed counterparts. These properties have important implications for various algorithms used by the ITK community such as thinning algorithms and Marching Cubes. We introduce two image filters which repair images that are ill-composed such that the output images are well-composed.

Keywords: *binary images, digital topology, well-composedness*

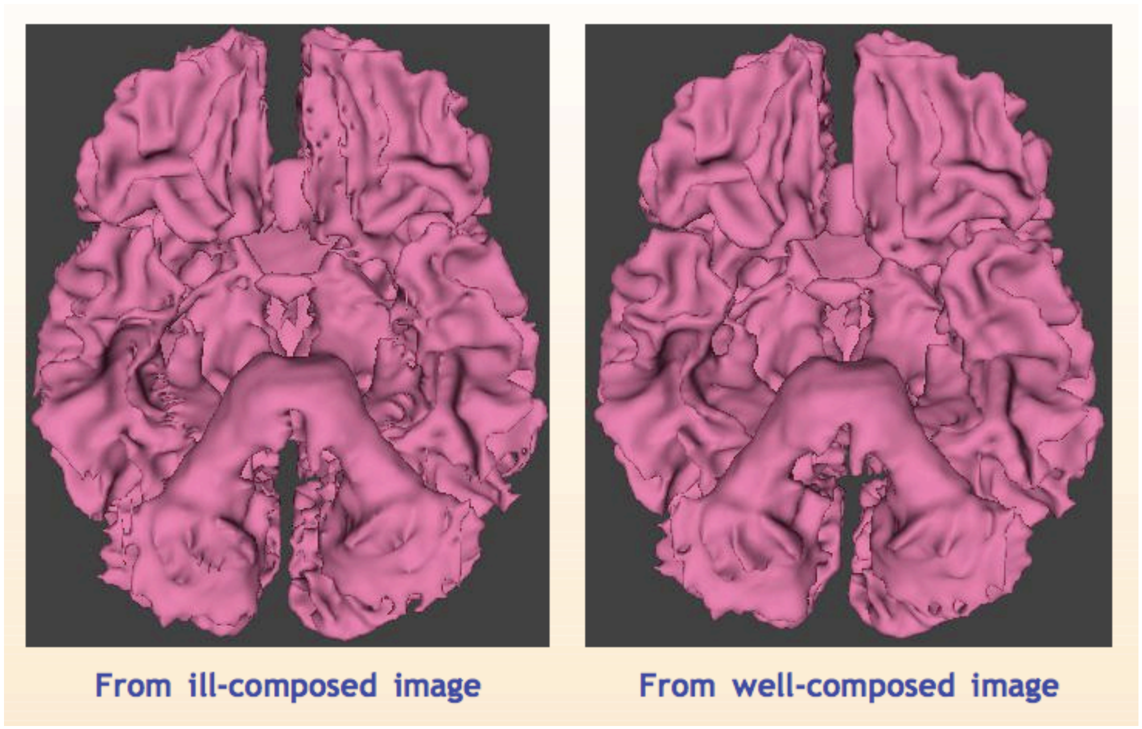
1 Introduction

Latecki et al. defined well-composed sets for the imaging community in [1]. For $n = \{2,3\}$, an n-D binary digital image is said to be *well-composed* if and only if the set of points in the pixel boundaries shared by the foreground and background points of the image is a $(n-1)$-D manifold. These digital images have notable salient properties such as adherence to the Jordan curve theorem (2-D) and the existence of a single connectedness relationship between points of the image [1]. Such images also simplify the well-known Marching Cubes algorithm [5, 6] (see Figure 1).

The well-composed property can be understood by illustrating the *critical configurations* which, if present, render an image ill-composed. These critical configurations for both the 2-D and 3-D case are illustrated in Figure 2. 2-D critical configurations (Figure 2(a)) exist whenever there are two neighboring background or foreground pixels that are 8-connected but not 4-connected. The 3-D case has two such critical configurations (which include reflections and 90° rotations) that are



(a)



(b)

Figure 1: Output of the marching cubes algorithm using ill-composed versus well-composed images. Unlike well-composed images, ill-composed images are susceptible to ambiguous cases for surface division using the marching cubes algorithm. Illustrated is (a) the superior-axial view and (b) the inferior-axial view of the brain.

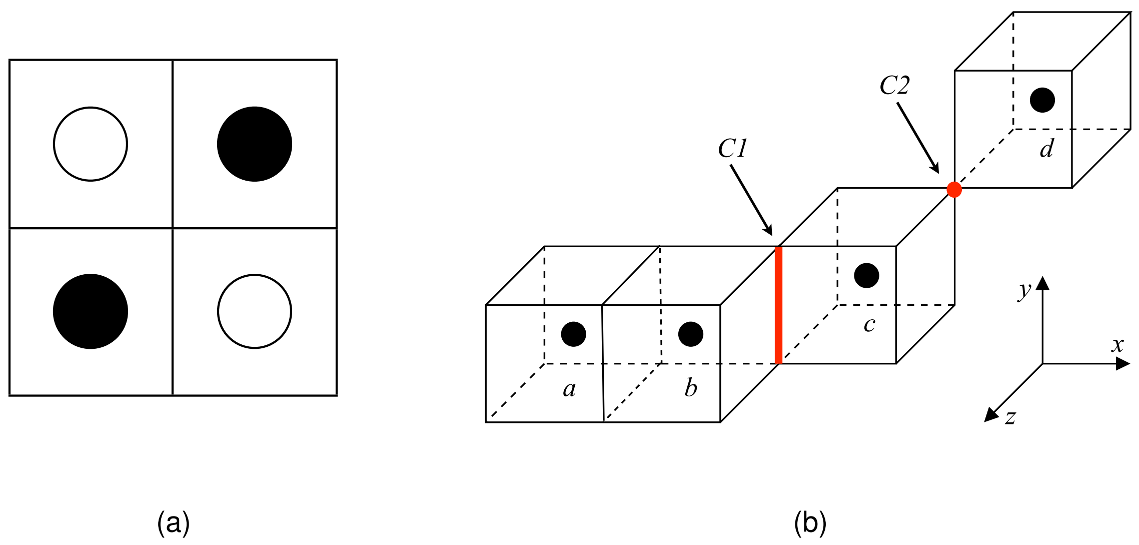


Figure 2: Ill-composed images are characterized by the presence of *critical configurations*. A simple critical configuration for 2-D images is illustrated in (a). Black and white dots represent background and foreground pixels, respectively. Two types of critical configurations are present in 3D images. These two types are shown in (b) in red and are labeled as $C1$ and $C2$.

illustrated in Figure 2(b). Note that the implicit voxels in the $2 \times 2 \times 2$ neighborhood surrounding the critical configurations in Figure 2(b) are background pixels.

If sufficiently high resolution is employed for 2-D digitization the image of an object will be guaranteed to have the same topology as the object itself [3]. However, given the unlikelihood of satisfying this condition for all digitization processes, some images exhibit topological ambiguities. These topological ambiguities can be corrected by changing certain background pixels to foreground pixels, or vice-versa. Our implementations converts problematic background pixels to foreground pixels. While there is no guarantee concerning the topological equivalence between the original object and its corresponding well-composed image, if the resulting well-composed image is “similar” to the ill-composed one, a repairing algorithm can be very useful in the context of several image-based applications.

This submission includes two image filters (one 2-D and one 3-D) for repairing digital images which are ill-composed. The deterministic 2-D algorithm that we include guarantees that the minimum number of changes occurs during the repairing process [2]. While the randomized 3-D algorithm has a larger upper bound for the minimum number of changes that are made, in practice the number of changes made is usually relatively small [5].

2 Algorithmic Implementation

The 2-D algorithm is taken from [2] whereas the 3-D algorithm is adapted from [5]. Both are derived from the `InPlaceImageFilter` class considering the repairing nature of the algorithm. Typical usage is given by the following code snippet:

```
 typedef itk::BinaryWellComposed2DImageFilter<ImageType> FilterType;
 FilterType::Pointer filter = FilterType::New();
 filter->SetInput( reader->GetOutput() );
 filter->DebugOn();
 filter->SetBackgroundValue( static_cast<PixelType>( 0 ) );
 filter->SetForegroundValue( static_cast<PixelType>( 1 ) );
 filter->SetFullInvariance( true );
```

As mentioned in [2], p. 67, “This method is invariant up to 90° rotations and reflections. If full invariance does not matter, we can repair a given picture to obtain a well-composed picture by adding fewer points. This can be achieved by considering only configurations given above and their reflections at a vertical axis (this is equivalent to considering only south-neighborhoods).” To accommodate the case where full invariance is not required, a simple function call setting the appropriate boolean variable is called, e.g.

```
 filter->SetFullInvariance( true );
```

Aside from the full invariance option, the API for the 3D filter is identical.

3 Results

3.1 2-D

2-D results of our algorithm are illustrated in Figure 3. We generated a random binary image of size 11×11 using the `RandomImageSource` class. We padded the resulting random image with a border width of five pixels for visual clarity (Figure 3(a)). The relevant code snippet is given as follows (taken from `Source/BinaryWellComposed2DImageFilter.cxx`):

```
itk::RandomImageSource<ImageType>::Pointer random
    = itk::RandomImageSource<ImageType>::New();
 float spacing[ImageDimension];
 unsigned long size[ImageDimension];
 float origin[ImageDimension];
 for ( unsigned int i = 0; i < ImageDimension; i++ )
   {
   spacing[i] = 1;
   size[i] = 17;
   origin[i] = 0;
   }
 random->SetMax( 2 );
 random->SetMin( 0 );
 random->SetOrigin( origin );
 random->SetSpacing( spacing );
 random->SetSize( size );
 random->Update();

 typedef itk::ConstantPadImageFilter<ImageType, ImageType> PadderType;
 PadderType::Pointer padder = PadderType::New();
 padder->SetInput( random->GetOutput() );
 padder->SetConstant( 0 );
```

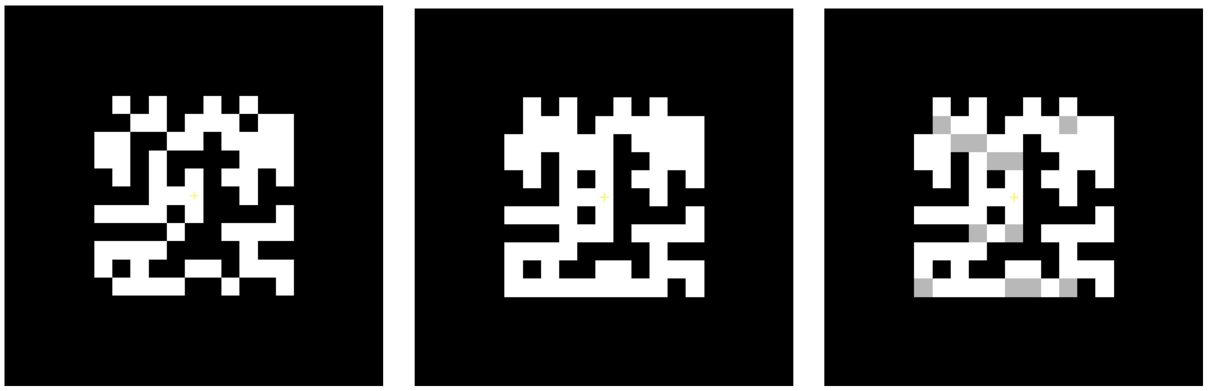


Figure 3: Results from our 2-D filter. An ill-composed random image is given in (a). After repairing the image with the algorithm, the image is now well-composed. The gray pixels in (c) are those pixels that were changed from background in the original image to foreground in the well-composed image.

```
unsigned long lowerfactors[ImageDimension];
unsigned long upperfactors[ImageDimension];
for ( unsigned int i = 0; i < ImageDimension; i++ )
  {
  lowerfactors[i] = 5;
  upperfactors[i] = 5;
  }
padder->SetPadLowerBound( lowerfactors );
padder->SetPadUpperBound( upperfactors );
padder->UpdateLargestPossibleRegion();

typedef itk::BinaryWellComposed2DImageFilter<ImageType> FilterType;
FilterType::Pointer filter = FilterType::New();
filter->SetInput(padder->GetOutput());
filter->SetBackgroundValue( static_cast<PixelType>( 0 ) );
filter->SetForegroundValue( static_cast<PixelType>( 1 ) );
filter->Update();
```

Notice that in the resulting well-composed image given in Figure 3(b) there are no 8-connected neighboring foreground or background pixels. The image in Figure 3(c) provides a clearer perspective of the changes that were made to produce the well-composed image. The pixels in gray are those pixels that were changed from background to foreground values.

3.2 3-D

We demonstrate the results of our 3-D algorithm using a segmented image of the brain with white matter voxels corresponding to foreground values. Three perpendicular views of the repaired images are given in Figure 4. Pixels in gray correspond to those image voxels that were changed from background to foreground values. Below each of the three perpendicular views are the magnified regions corresponding to the regions outlined in red. The repaired image was produced by the following code snippet (taken from `Source/BinaryWellComposed3DImageFilter.cxx`):

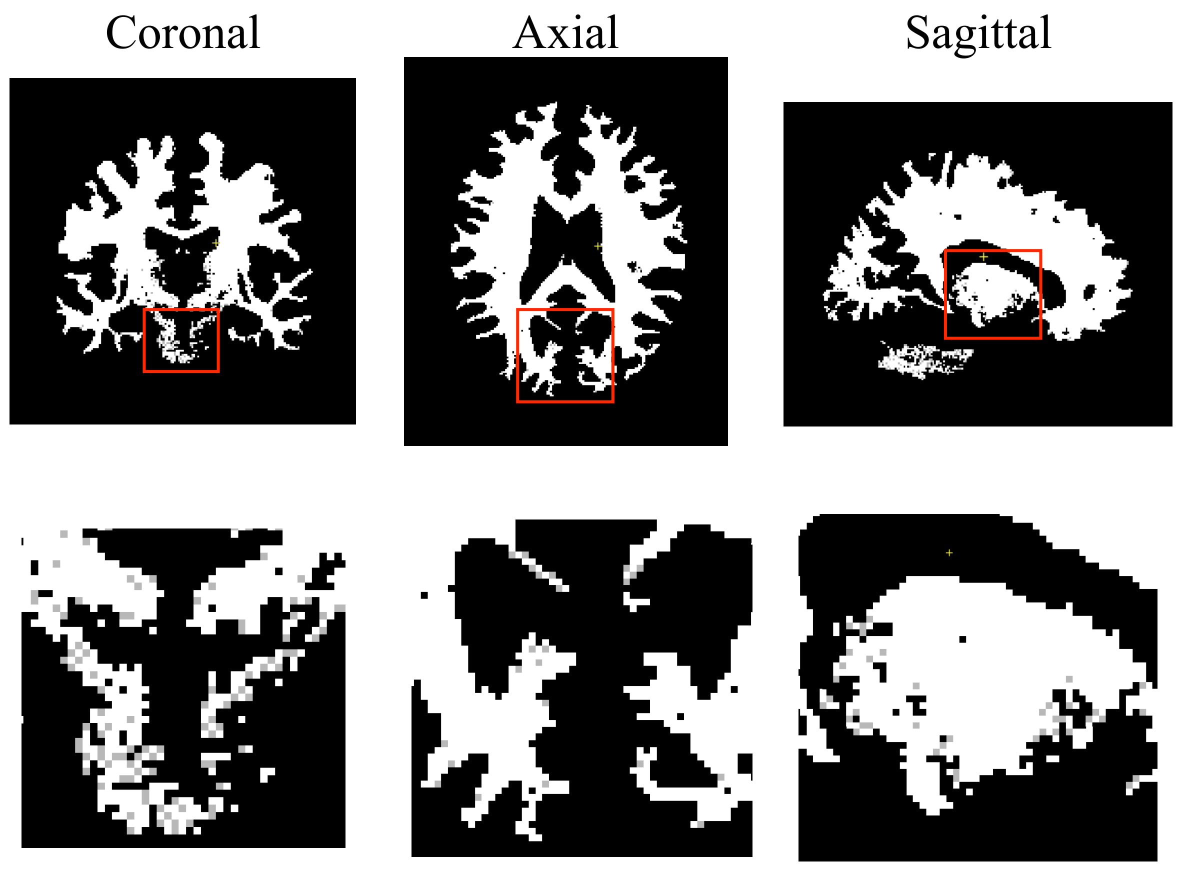


Figure 4: Results from our 3-D filter. An ill-composed 3-D segmentation image of white matter is repaired. We illustrate the repairs made by rendering the changed voxels (from the background value to the foreground value) in gray.

```
typedef itk::BinaryWellComposed3DImageFilter<ImageType> FilterType;
FilterType::Pointer filter = FilterType::New();
filter->SetInput( reader->GetOutput() );
filter->SetBackgroundValue( static_cast<PixelType>( 0 ) );
filter->SetForegroundValue( static_cast<PixelType>( 1 ) );
filter->Update();
```

References

[1] L. Latecki, U. Eckhardt, and A. Rosenfeld, “Well-Composed Sets”, *Computer Vision and Image Understanding*, 61(1):70-83, 1995. 1

[2] L. J. Latecki, *Discrete Representation of Spatial Objects in Computer Vision*, Kluwer Academic Publishers, 1998, pp. 66-67. 1, 2, 2

[3] A. Gross and L. J. Latecki, “Digitizations preserving topological and differential geometric properties”, *Computer Vision and Image Understanding*, 62(3):370-381, 1995. 1

[4] P. Stelldinger, L. J. Latecki, and M. Siqueira, ”Topological Equivalence between a 3D Object and the Reconstruction of its Digital Image”, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (accepted for publication).

[5] M. Siqueira, L. J. Latecki, and J. Gallier, “Making 3D Binary Digital Images Well-Composed”, *Proc. of the IS&T/SPIE Conference on Vision Geometry XIII*, SPIE Vol. 5675, 150-163, 2005. 1, 1, 2

[6] M. Siqueira, L. J. Latecki, and J. Gallier, “Making 3D Binary Digital Images Well-Composed”, *Technical Report MS-CIS-04-22*, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA, 2004. 1