Well-Composedness and the Topological Repairing of 2-D and 3-D Digital Images

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

January 9, 2007

[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

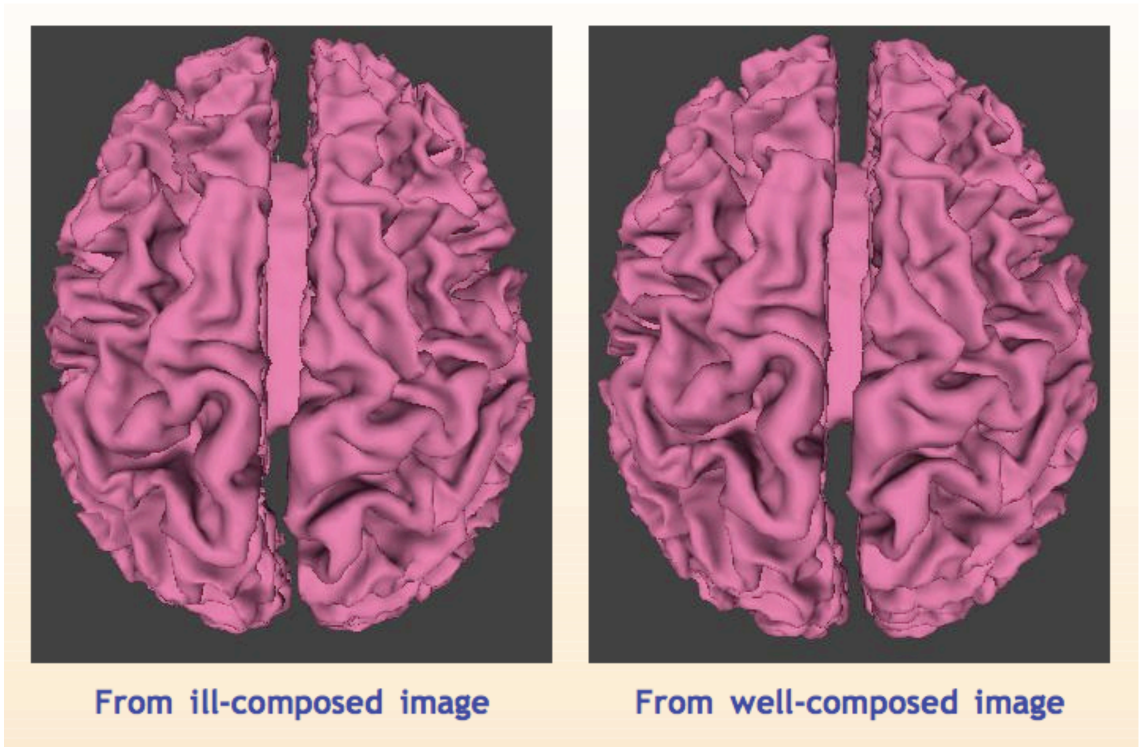
In an earlier submission [8], we considered 2-D and 3-D digital *binary* images topologically characterized by their *well-composedness*. Well-composed images exhibit important topological and geometrical properties not shared by their ill-composed counterparts. These properties have important implications for various algorithms used by the ITK community such as thinning algorithms and Marching Cubes.

A natural follow-up inquiry to our original submission concerns the possible extension of the binary algorithm to accommodate images with multiple labels. In this paper, we describe such a generalization of our previous submission which also subsumes the binary approach. For completeness and clarity of exposition, we develop the idea of well-composedness within the binary image context and later generalize the discussion to include those images with multiple labels.

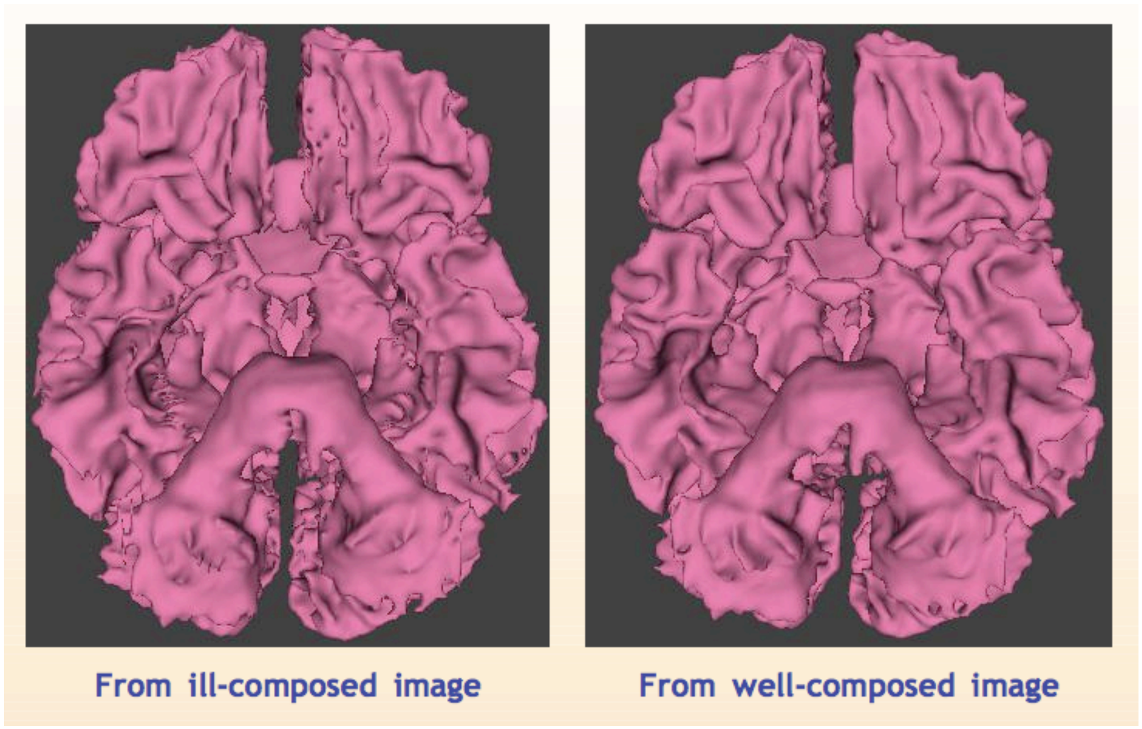
Keywords: *digital images, digital topology, well-composedness*

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).



(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.

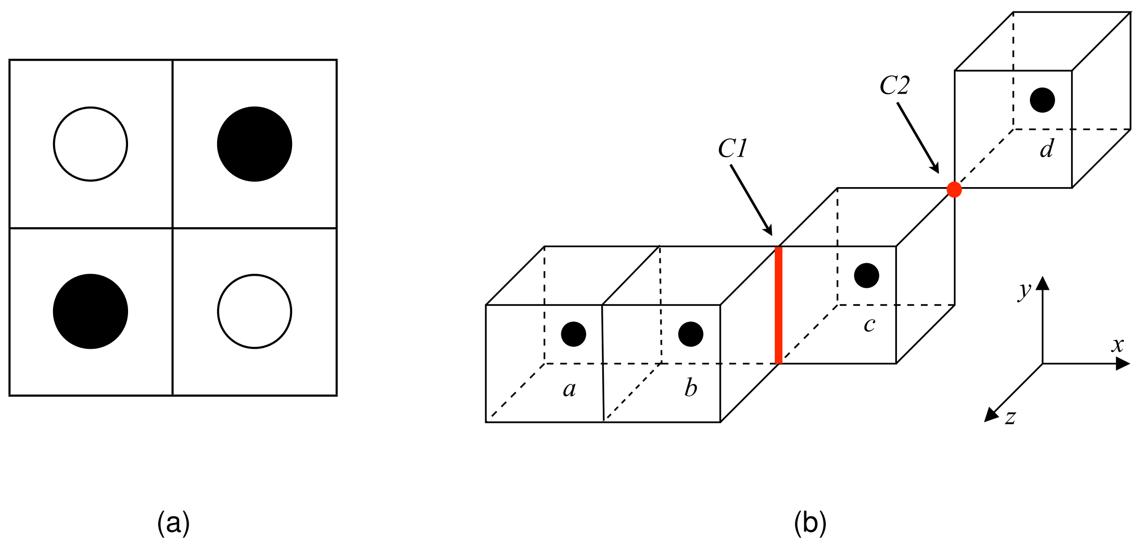


Figure 2: Ill-composed images are characterized by the presence of *critical configurations*. A simple critical configuration for 2-D images is illustrated in (a). Black and white dots represent background and foreground pixels, respectively. Two types of critical configurations are present in 3D images.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

1.1 Well-Composedness for Binary Images

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 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. Analagous research into 3-D objects and their 3-D pictorial digitization is discussed in [4].

Such 2-D and 3-D topological ambiguities can be corrected by changing certain background pixels to foreground pixels, or vice-versa. Our implementation 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 for facilitating image processing.

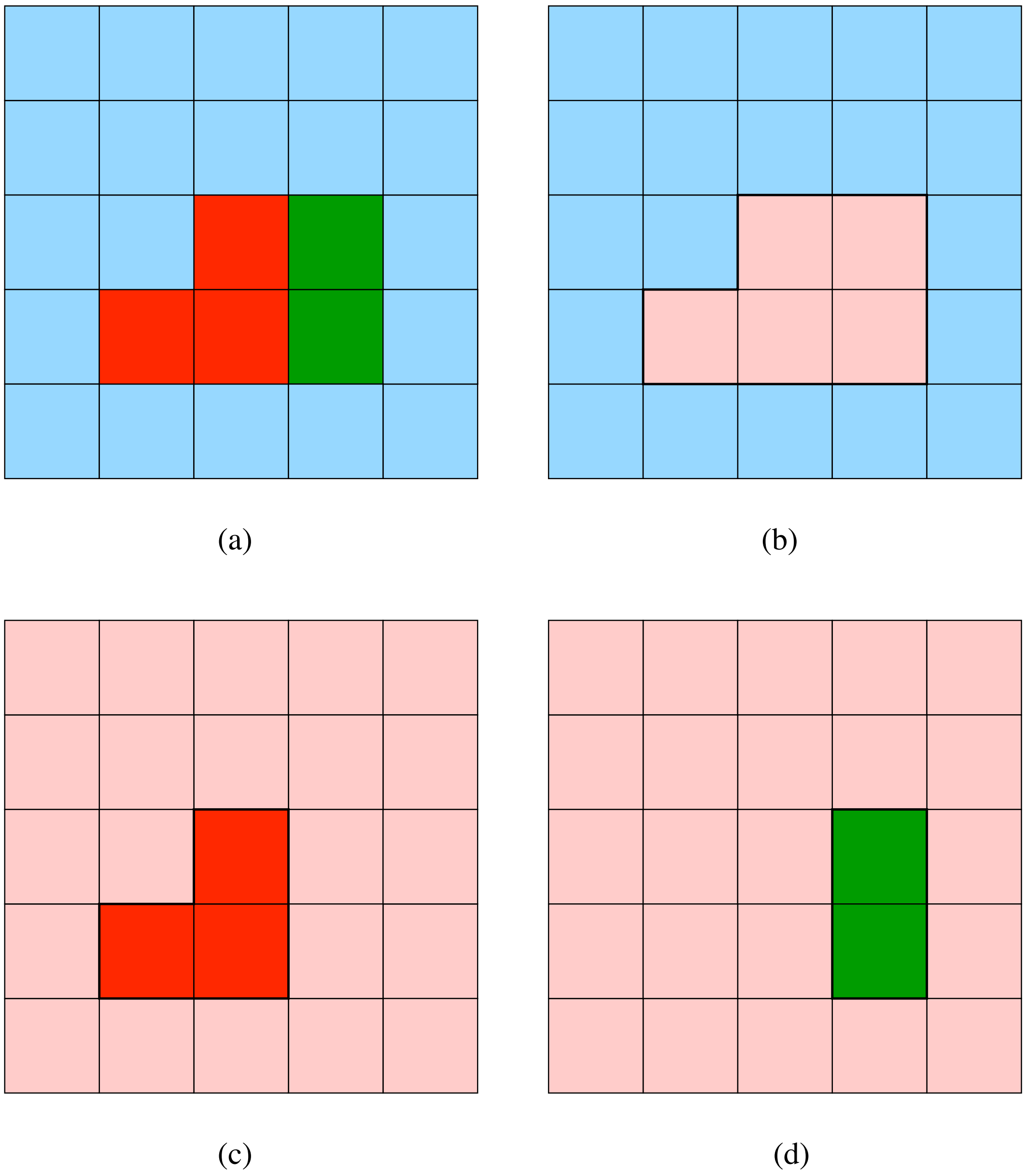


Figure 3: (a) Well-composed image with three colors. (b) Blue component and its complement. (c) Red component and its complement. (d) Green component and its complement.

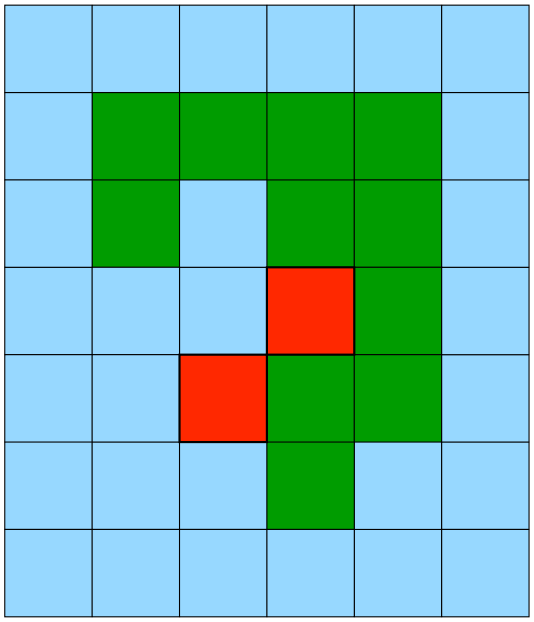


Figure 4: An example to demonstrate that the naive extension of the repairing algorithm for binary images does not always work: by adding a red pixel to eliminate the critical configuration, the algorithm will cause either the blue or the green component to become ill-composed. If the red component is the last one considered by the algorithm, then the resulting image will be ill-composed.

1.2 Well-Composedness for Multilabel Images

Defining well-composedness for images with multiple labels is straightforward. An n-label image (i.e. each image voxel has a value from the set $\{0,1,\ldots,n-1\}$) is well-composed if each of the set of n binary images derived by considering each label and its complement are well-composed. We illustrate this concept for 2-D images in Figure 3. Naive intuition might lead one to consider repairing multilabel images by applying the binary algorithm to the n-label image n times—once for each label. However, such an approach would generally fail. Consider the step of repairing the image for label i, where $i \leq n-1$. Any repairs could potentially "break" the image for label j where $j < i$. This is demonstrated in Figure 4.

2 Algorithmic Overview

Both the 2-D and 3-D algorithms require locating problematic configurations of pixels and then repairing them in such a way that no new or minimally new problematic configurations are created with the repair. The 2-D binary algorithm is discussed in [2] whereas the 3-D binary algorithm is introduced in [5]. The extension to multilabel 3-D images is given in [7]. Given below is a brief overview of the binary algorithm. At its core, the multilabel algorithm utilizes the binary algorithm with a slight modification. For interested readers, we refer discussion of the modification to the relevant literature [7].

The accompanying code submission consists of a single image filter for repairing 2-D and 3-D digital multilabeled 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 expected number of changes that are made, in practice the number of changes made is usually relatively small [5]. We wish to emphasize that the 2-D and 3-D algorithms are theoretically distinct requiring that a portion of the software handle the 2-D image case and the remaining portion handle the 3-D image case.

2.1 2-D Binary Algorithm

The deterministic 2-D algorithm requires a single pass through the image. At each pixel, the 9-neighborhood is inspected for problematic configurations. There are four distinct configurations (excluding 90° rotations and reflections) with one of those four configurations potentially exhibiting a special case. These five cases are illustrated in Figure 5. Due to the duality of well-composedness (the set S is well-composed if and only if and its complement S^c is well-composed), well-composedness can be obtained by swapping the foreground and background values in the cases shown. Further details can be found in [2].

2.2 3-D Binary Algorithm

The randomized 3-D algorithm is slightly more complex. An initial pass is made through the entire image. The voxel index for each problematic $C1$ and $C2$ configuration (see Figure 2) is added to a list. Problematic voxels in the list are then repaired such that no new problematic configurations are created. For the cases in which this is not possible, a new random critical configuration is created and added to the list. It can be shown that the expected number of new critical configurations created by the repairing algorithm is less than $m/2$ where m is the original number of critical configurations [5]. The various cases for the critical $C1$ and $C2$ configurations are given in Figures 6 and 7.

3 Implementation

Implementation of both algorithms is contained in the `WellComposedImageFilter`. Which algorithm is evoked depends upon the dimensioality of the templated image type. We derive the filter from `InPlaceImageFilter` class which seems like a natural choice considering the repairing nature of the algorithm. **Please note that the algorithm expects as input an ordered label image such that the voxel values are taken from the set $\{0, 1, \ldots, n-1\}$ for an n-label image.** An unordered label image can be corrected beforehand by filtering with the ITK class `RelabelComponentImageFilter`. Shown below is a simple 2-D illustration of implementation usage.

```
int itkWellComposedImageFilterTest2D( int argc, char **argv )
{
  typedef int                          PixelType;
```

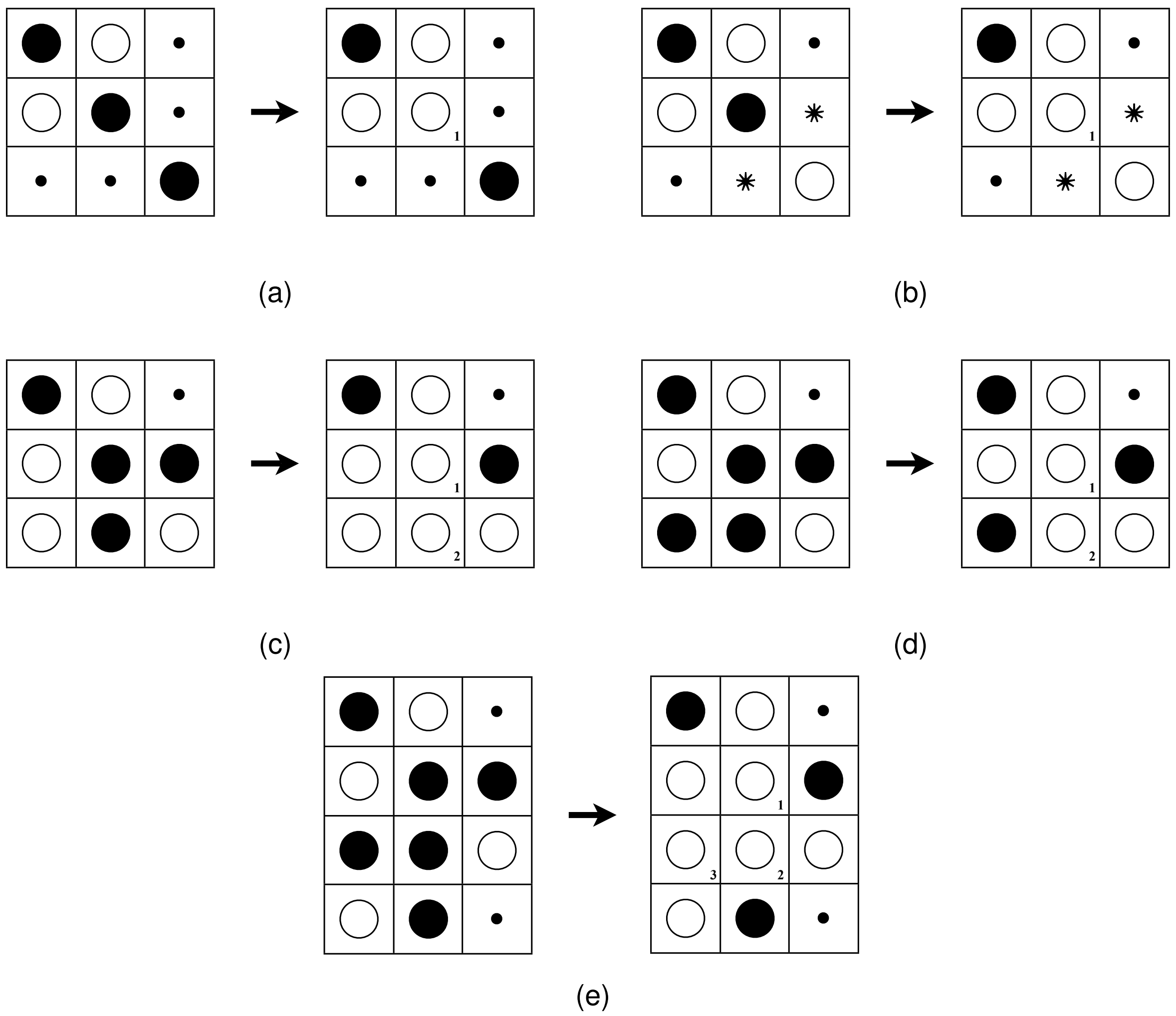


Figure 5: (a)-(d) The four canonical 9-neighborhood configurations for 2-D ill-composed images and the repairs required to make the image well-composed. (e) Special case of the configuration in (d). Repairs are made by changing background pixels (represented by the large '•') to foreground pixels (represented by the large '○'). The small '○' denote that the pixel at the spot can have a foreground or background value. (a)-(b) Repairs are made by simply changing the center pixel. In (b), one of the two pixels denoted by '$*$' must have a foreground value. In (c) and (d), the middle pixel and the neighbor directly below it are converted to foreground values. However, in the case of (e) the diagonal pixel below and to the left must also be converted.

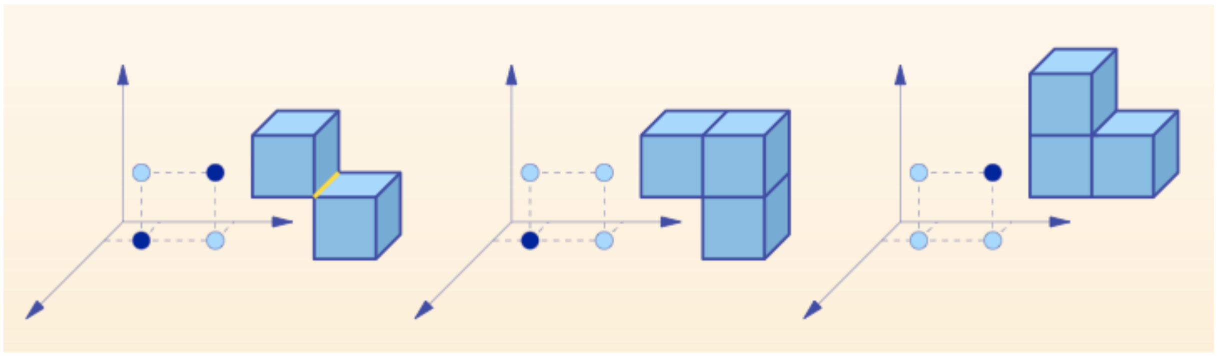


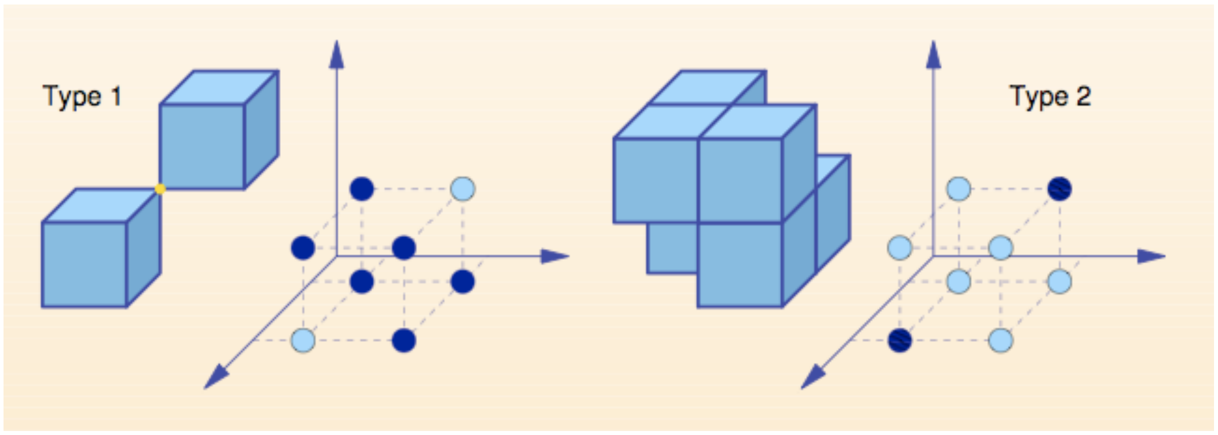
Figure 6: $C1$ configurations are repaired by changing one or both of its background points.

```
  typedef itk::Image<PixelType, 2> Image2DType;

  typedef itk::ImageFileReader<Image2DType> Reader2DType;
  Reader2DType::Pointer reader2D = Reader2DType::New();
  reader2D->SetFileName( argv[1] );
  reader2D->Update();

  try
    {
    typedef itk::WellComposedImageFilter<Image2DType> Filter2DType;
    Filter2DType::Pointer filter2D = Filter2DType::New();
    filter2D->SetInput( reader2D->GetOutput() );
    filter2D->SetFullInvariance( true );
    filter2D->SetTotalNumberOfLabels( atoi( argv[3] ) );

    typedef itk::ImageFileWriter<Image2DType> Writer2DType;
    Writer2DType::Pointer writer2D = Writer2DType::New();
    writer2D->SetInput( filter2D->GetOutput() );
    writer2D->SetFileName( argv[2] );
    writer2D->Update();
    }
  catch (itk::ExceptionObject & err)
    {
    std::cout << "ExceptionObject caught !" << std::endl;
    std::cout << err << std::endl;
    return EXIT_FAILURE;
    }
}

int itkWellComposedImageFilterTest3D( int argc, char **argv )
{
  typedef int                      PixelType;
  typedef itk::Image<PixelType, 3> Image3DType;
```

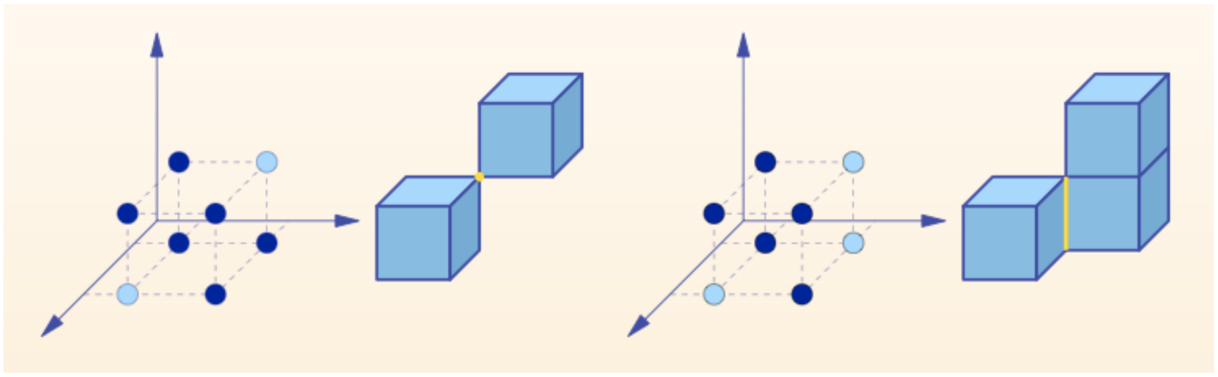
Aside from the full invariance option (explained below), the API for the 3D filter is identical.

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.

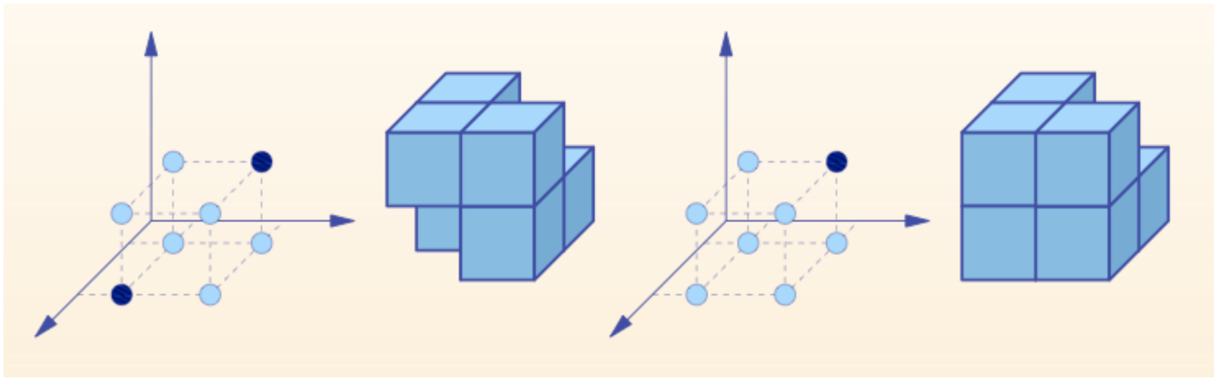
```
    writer2D->SetFileName( argv[2] );
```



(a)

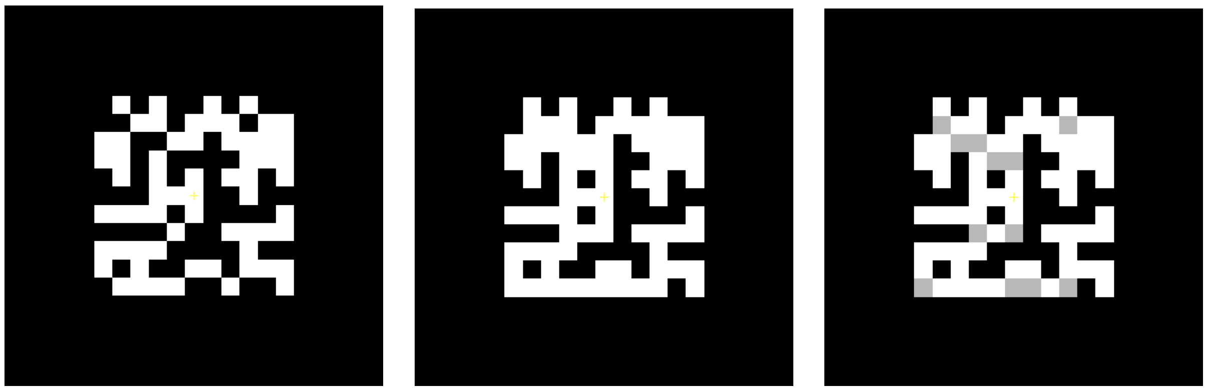


(b)



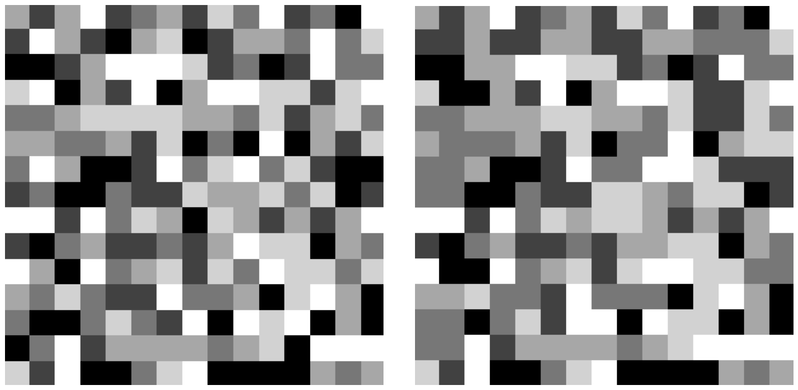
(c)

Figure 7: (a) $C2$ configurations manifest themselves in two types. (b) Repairing Type 1 configurations introduces a new $C1$ configuration. (c) Repairing Type 2 configurations involves changing one or both implicit background voxels to foreground voxels.



(a) (b) (c)

Figure 8: Results from our 2-D filter on a binary image. An ill-composed random image is given in (a). After repairing the image with the algorithm, the image in (b) 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.



(a) (b)

Figure 9: Results from our 2-D filter on a 6-label image. An ill-composed random image is given in (a). After repairing the image with the algorithm, the image is now well-composed.

4 Results

4.1 2-D

2-D results of our algorithm are illustrated in Figure 8 and Figure 9. 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 8(a)). Notice that in the resulting well-composed image given in Figure 8(b) there are no 8-connected neighboring foreground or background pixels. The image in Figure 8(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. The results for a 6-label random image are given in Figure 9.

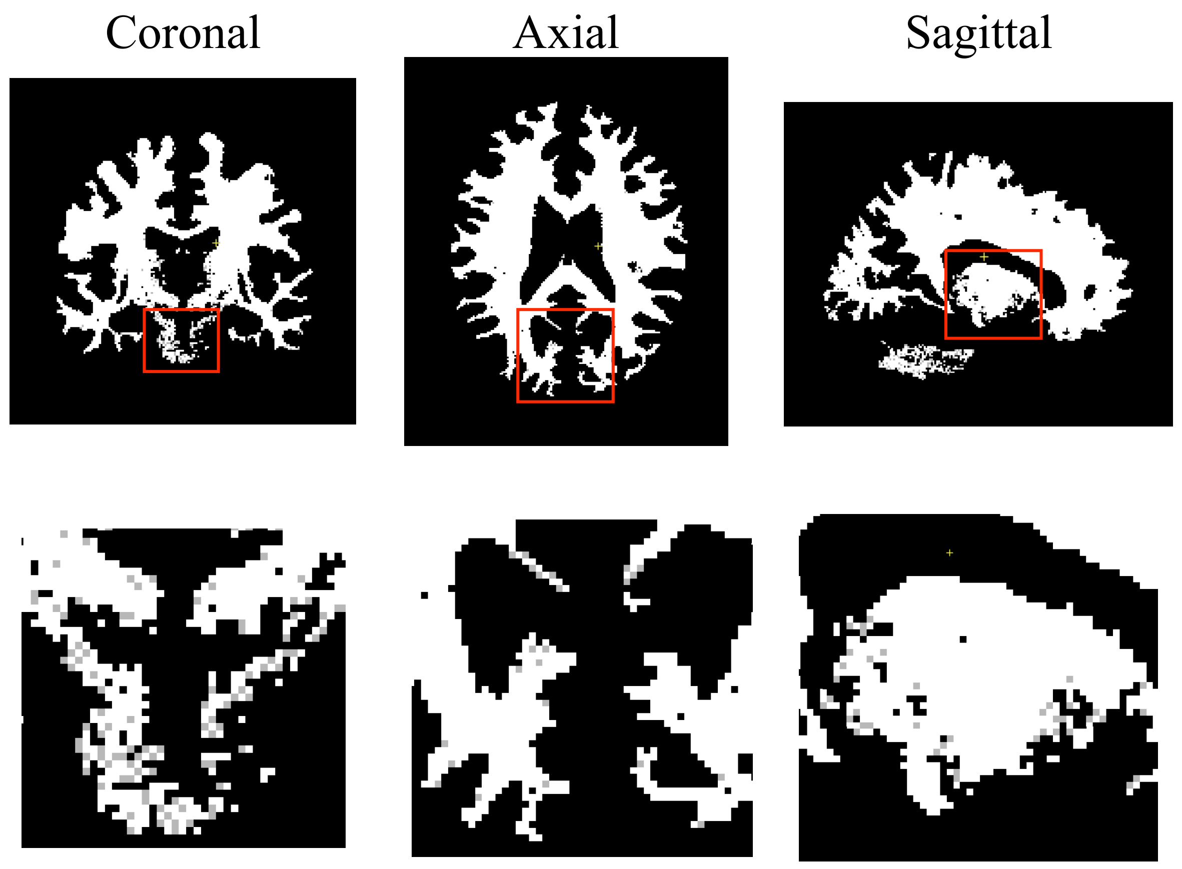


Figure 10: Results from our 3-D filter on a binary image. An ill-composed 3-D segmentation image of white matter is repaired. We illustrate the repairs made by rendering the changed voxels (from the background value to the foreground value) in gray.

4.2 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 10. 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. Multilabel results are provided in Figure 11.

References

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

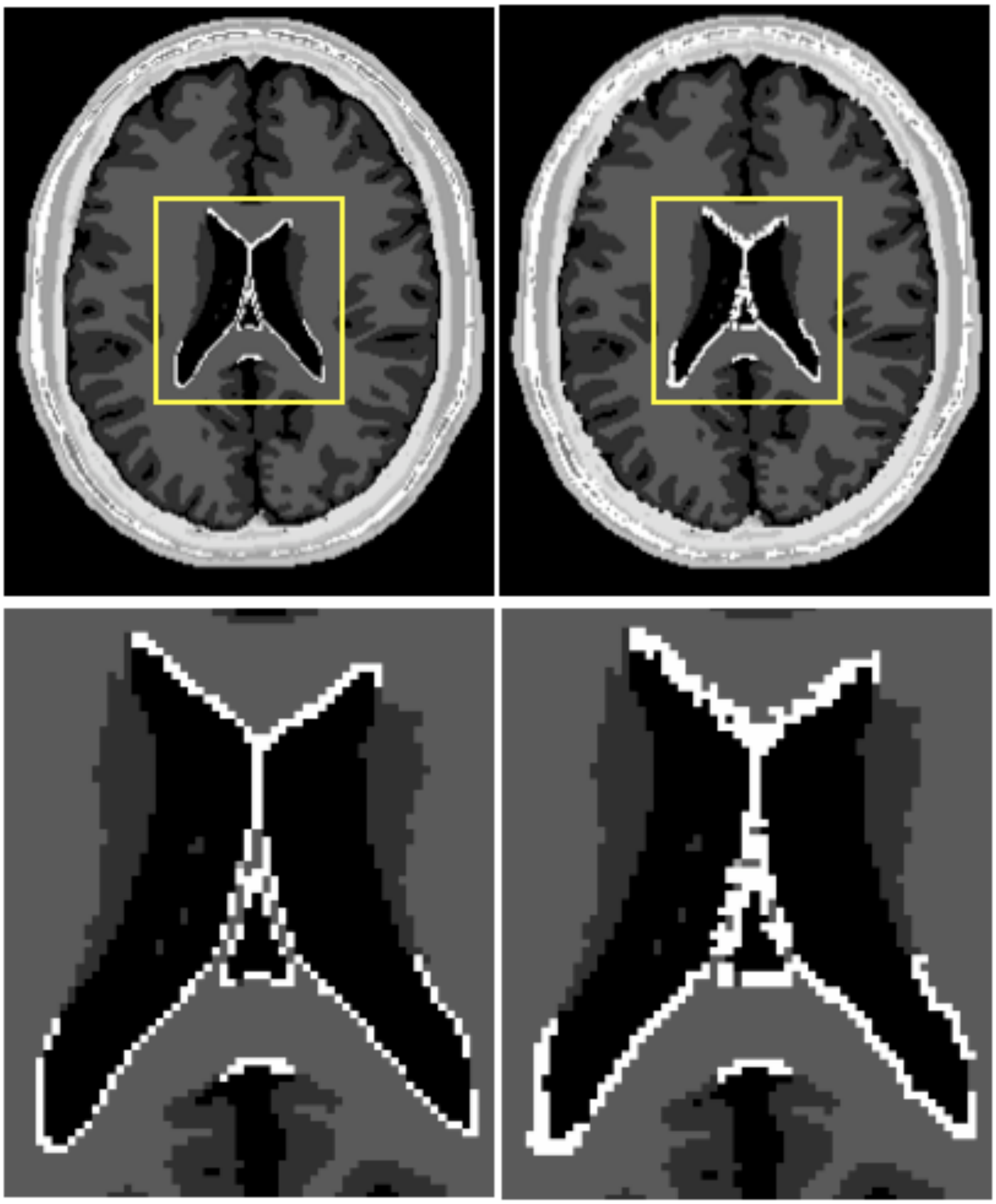


Figure 11: Results from our 3-D filter on a 10-label image. A segmented image labeling the various tissues in the brain. The top left is an axial slice of the original 3-D image whereas the top right is the same slice after topological correction. The bottom images provides a close-up view of the ventricles (uncorrected—left and corrected—right) corresponding to the yellow regions in the top images.

[2] L. J. Latecki, *Discrete Representation of Spatial Objects in Computer Vision*, Kluwer Academic Publishers, 1998, pp. 66-67. 2, 2.1, 3

[3] A. Gross and L. J. Latecki, “Digitizations preserving topological and differential geometric properties”, *Computer Vision and Image Understanding*, 62(3):370-381, 1995. 1.1

[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*, 29(1):126-140. 1.1

[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, 2, 2.2

[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

[7] M. Siqueira, L. J. Latecki, J. Gallier, N. Tustison, J. Gee, “Topological Repairing of 3-D Digital Images”, *Journal of Mathematical Imaging and Vision*, submitted. 2

[8] N. J. Tustison, M. Siqueira, and J. C. Gee, “Well-Composed Image Filters for Repairing 2-D and 3-D Binary Images”, http://hdl.handle.net/1926/305, *Insight Journal*, Aug, 2006. (document)