Improving performance of morphological reconstruction

Release 0.00

Richard Beare

February 15, 2006

Department of Medicine, Monash University, Australia

Abstract

Morphological reconstruction may be implemented in a number of different ways. ITK has an iterative method and a non iterative method. This article compares the performance of another non iterative method and finds a significant improvement.

Contents

1 Introduction

Morphological reconstruction by erosion or dilation, also referred to as geodesic erosion or dilation, operate on a marker image and a mask image. In the case of dilation the marker image is dilated by an elementary structuring element and the pixelwise minimum of the result and the mask is computed. This procedure can be run for a fixed number of iterations or until convergence.

The original implementation of morhpological reconstruction in ITK found in **itkGrayscaleGeodesicDilateImageFilter** and **itkGrayscaleGeodesicErodeImageFilter** provides a direct implementation of the approach outlined above. Unfortunately this approach is very time consuming when the aim is to run until convergence.

A faster, non iterative implementation, has been provided in **itkReconstructionByDilationImageFilter** and **itkReconstructionByErosionImageFilter**. These filters implement the algorithm described in [1]. Actually the algorithm is slightly modified to support non integer pixel types by using a map instead of an array and a set to store indexes, so a "pure" implementation of the Robinson algorithm may perform better than the modified version. Copies of these filters, with **Robinson** appended to the filename, are included in the archive associated with this article.

This report investigates the performance of another algorithm described in [2] and demonstrates another significant improvement in performance.

The files in the archive associated with this article can be used to replace files of the same name in the itk hierarchy.

2 Hybrid algorithm

The hybrid algorithm for morphological reconstruction described in [2] is:

- I : mask image
- J : marker image, defined on domain D_I
- scane D_I in raster order:
  - Let p be the current pixel;
  - $J(p) = (max\{J(q), q \in N_G^+(p) \cup \{p\}\}) \wedge I(p)$
- scane D_I in anti-raster order:
  - Let p be the current pixel;
  - $J(p) = (max\{J(q), q \in N_G^-(p) \cup \{p\}\}) \wedge I(p)$
  - If there exist $q \in N_G^-(p)$ such that $J(q) < J(p)$ and $J(q) < I(q)$ then *fifo_add(q)*
- Propagation step: While (!fifo_empty)
  - p = fifo_first()
  - For every pixel $q \in N_G(p)$:
    - If $J(q) < J(p)$ and $I(q) \neq J(q)$ then
      $J(q) = \min\{J(p), I(q)\}$
      fifo_add(q)

where $N_G(p)$ denotes the neighborhood of p on a grid G, with N^+ and N^- denoting the set of neighbors reached before and after p during a raster scan. $\wedge$ is the pointwise minimum operator.

3 Implementation details

The implementation of this algorithm is provided by **itkReconstructionImageFilter**, which takes functor template arguments to provide either erosion or dilation reconstruction filters. The reconstruction by erosion filter is **MorphologicalReconstructionByErosionImageFilter** and the reconstruction by dilation filter is **MorphologicalReconstructionByDilationImageFilter**. These filters are verified by comparison with the existing implementation using a series of tests in which regional extrema are supressed. The tests use **itkHMinimaImageFilter** and **itkHMaximaImageFilter** which are versions of **itkHMinimaImageFilter** and **itkHMaximaImageFilter** that have been modified to use the new reconstruction filter. The old versions are available in the archive as **itkHMinimaImageFilterRobinson** etc.

A typical requirement is that the mask must always be greater than or equal to the marker image. In the new filter this restriction is enforced implicitly.

3.1 Optimization

Profiling of the initial implementation suggested that the boundary checks were contributing a significant overhead, as would be expected. An attempt was made to use the face calculator to improve performance without increasing the size of the image (these modifications are present in the code and may be activated by defining the symbole “FACE” in the header file). Unfortunately the algorithm is strongly dependent on visit order and operating on face and body regions changes the visit ordering and produces incorrect results. This strategy may be feasible with much more careful analysis of the region borders.

A much simpler optimization is to pad the image. This option can be activated by defining the “COPY” macro in the header file and provides a considerable improvement in execution speed at the cost of greater memory use. Someone should probably check my use of the pad and crop filters to make sure that the mini-pipeline conforms to standards.

4 Performance tests

The first column indicates the connectivity - 0 is face connected, 1 is fully connected.

2D image without optimization:

```
> ./perf2D  images/cthead1.png
#F      Robinson        Vincent
0       0.1324          0.0218
1       0.1427          0.0325
```

3D image without optimization:

```
> ./perf3D  images/ESCells.hdr
#F      Robinson        Vincent
0       19.931          3.867
1       27.534          9.444
```

2D images with edge padding:

```
> ./perf2D images/ctheadl.pn g
#F Robinson Vincent
0 0.1355 0.0184
1 0.1474 0.0223
```

3D image with edge padding:

```
> ./perf3D images/ESCells.hdr
#F Robinson Vincent
0 20.135 2.736
1 27.765 5.071
```

5 Memory requirements

The Vincent algorithm is fairly efficient with respect to memory requirements. Most of the work is done directly on the output image, with additional memory resources being consumed by the FIFO. The size of the FIFO is data dependent, but should never have more elements than the original image.

The optimization approach employing padding makes padded copies of both of the input images, which effectively doubles the memory requirements:

Unoptimized : mask, marker, output, FIFO.

optimized : mask, marker, padded mask, padded marker, output, FIFO.

6 Sample code

The following code is from **test2Dmax.cxx** and compares the operation of the two implementations:

```
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"

#include "itkHMaximaImageFilter Robinson.h"
#include "itkHMaximaImageFilter.h"
#include "itkSimpleFilterWatcher.h"

int main(int, char * argv[])
{

  const int dim = 2;
  typedef unsigned char PType;
  typedef itk::Image< PType, dim > IType;

  // read the input image
  typedef itk::ImageFileReader< IType > ReaderType;
  ReaderType::Pointer reader = ReaderType::New();
  reader->SetFileName( argv[1] );
  reader->Update();
```

```
  int height = atoi(argv[4]);
  int F = atoi(argv[5]);
  typedef itk::HMaximaImageF ilter< IType, IType > ReconType;
  ReconType::Pointer recon = ReconType::New();
  itk::SimpleFilterWat cher watcher(recon, "recon");
  recon->SetInput( reader->GetOutput() );
  recon->SetHeight(hei ght );

  typedef itk::HMaximaImageF ilterR cbi nso n< IType, IType > OrigReconType;
  OrigReconType::Point er origrecon = OrigReconType::New();
  origrecon->SetInput( reader->GetOutput() );
  origrecon->SetHeight (he igh t);
  itk::SimpleFilterWat cher watcher2(origrecon, "origrecon");

  origrecon->SetFullyC onn ect ed( F);
  recon->SetFullyConne cte d(F );

  origrecon->Update();

  recon->Update();

  typedef itk::ImageFileWrit er< IType > WriterType;
  WriterType::Pointer writer = WriterType::New();
  writer->SetFileName( argv[2] );

  writer->SetInput(rec on->GetOu tpu t() );
  writer->Update();

  writer->SetFileName( arg v[3 ]);
  writer->SetInput(ori gre con->GetO utp ut( ));
  writer->Update();


  return 0;
}
```

7 Conclusion

The new implemenation of the Vincent algorithm is between 5 and 8 times faster than the Robinson algorithm, depending on the choice of connectivity and image padding optimization. Optimization by padding the boundary produced a significant saving for the 3D dataset used in testing. The saving was not as significant for 2D data.

References

[1] K. Robinson and P. Whelan. Efficient morphological * reconstruction: A downhill filter. *Pattern Recognition Letters*, 25(15):1759–1767, 2004. 1

[2] L. Vincent. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, 1993. 1, 2