# Image projections along an axe

Gaëtan Lehmann

January 27, 2006

Unité de Biologie du Développement et de la Reproduction, Institut National de la Recherche Agronomique, 78350 Jouy-en-Josas, France

**Abstract**

Image projection is a very common task in image analysis to reduce the dimension of an image. A base filter is provided with some specialized filters which implement different projection methods.

## 1 Introduction

ITK already provide a class to project an image along an axe: `itk::AccumulateImageFilter`. However, this class is limited to sum projection and mean projection.

## 2 Implementation

The base class, ProjectionImageFilter, is mainly a copy of AccumulateImageFilter, with some enhancement:

- projection is delegated to a functor
- report its progress

Different projection types are provided, with specialized classes:

- maximum projection (MaximumProjectionImageFilter)
- minimum projection (MinimumProjectionImageFilter)
- mean projection (MeanProjectionImageFilter)
- median projection (MedianProjectionImageFilter)
- sum projection (SumProjectionImageFilter)

Other projections types can easily be created just by providing a new functor.

AccumulateImageFilter is not modifiable to implement functor support without breaking the API.

## 3 Performance

A timing test comparing performance on a $371 \times 371 \times 34$ image showed that the results are similar with AccumulateImageFilter and the equivalent ProjectionImageFilter, and that the new projection types have similar performance. The results achieved on an Athlon 64 Processor 2800+ (1802 MHz) with 512Kb cache, 512 Mb of RAM and gcc 4.0.2 are shown in Table 1.

| Filter | execution time |
|---|---|
| MaximumProjectionImageFilter | 0.652 s |
| MinimumProjectionImageFilter | 0.614 s |
| SumProjectionImageFilter | 0.614 s |
| MeanProjectionImageFilter | 0.619 s |
| MedianProjectionImageFilter | 0.831 s |
| AccumulateImageFilter, sum | 0.615 s |
| AccumulateImageFilter, mean | 0.617 s |

Table 1: Execution time.

## 4 Examples

Figure 1, Figure 2, Figure 3, Figure 4, Figure 5 and Figure 6 show the result of different projection on different axes. The result of SumProjectionImageFilter in similar to the result of MeanProjectionImageFilter, but with a higher range of values, and so is not shown here.

Figure 1: MaximumProjectionImageFilter, projection along axe x.

Figure 2: MaximumProjectionImageFilter, projection along axe y.

## 5 Sample code

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

#include "itkMaximumProjectionImageFilter.h"
#include "itkExtractImageFilter.h"


int main(int, char * argv[])
{
  int axe = atoi(argv[1]);
```

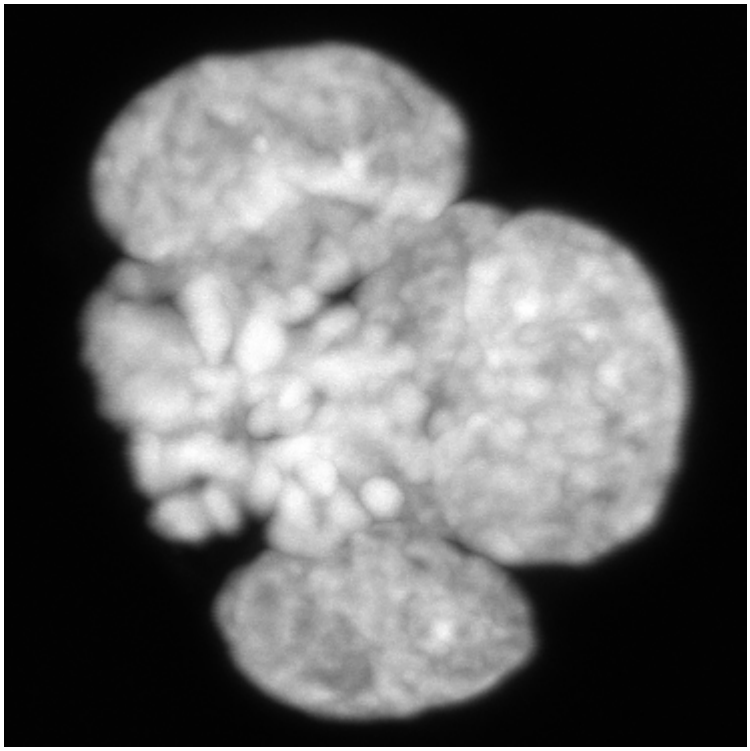

Figure 3: MaximumProjectionImageFilter, projection along axe z.

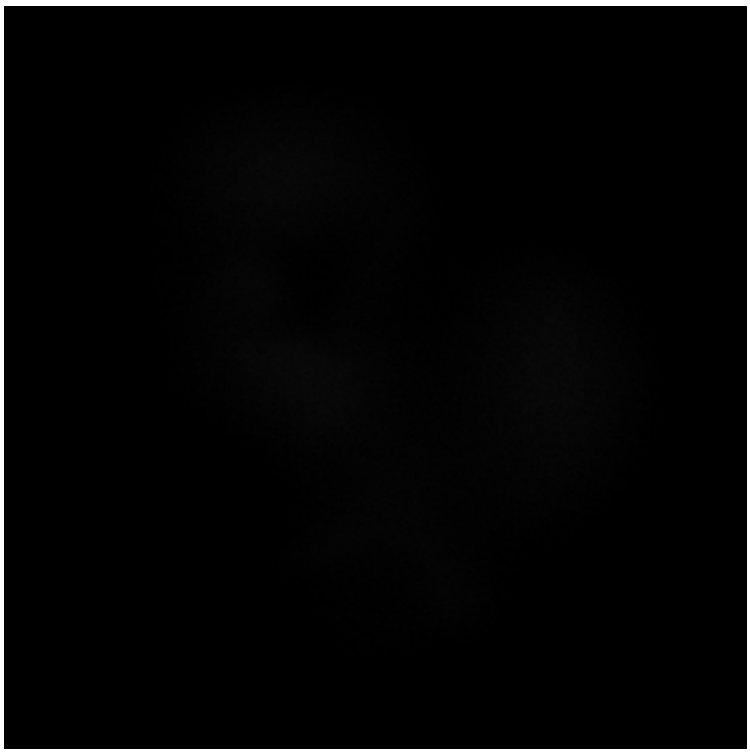

Figure 4: MinimumProjectionImageFilter, projection along axe z.

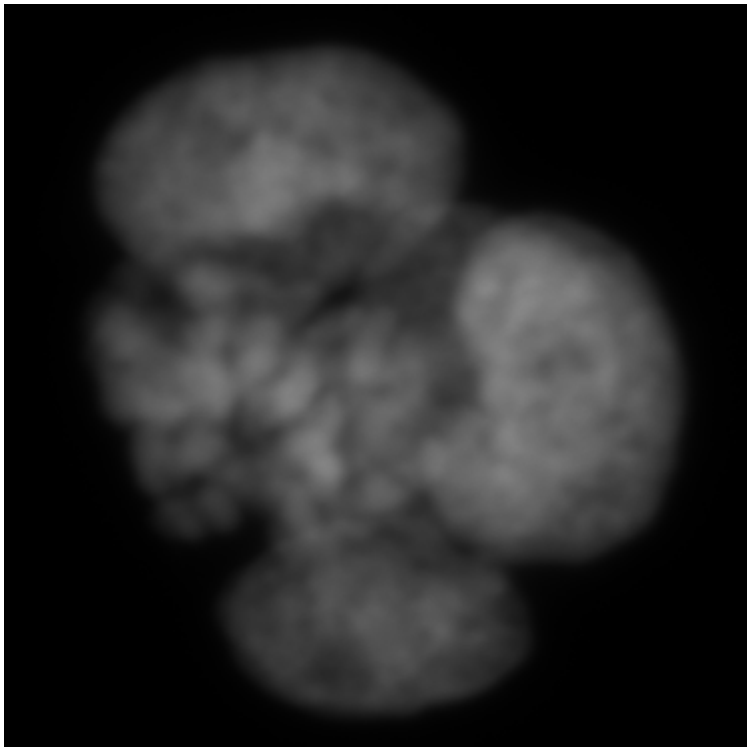

Figure 5: MeanProjectionImageFilter, projection along axe z.

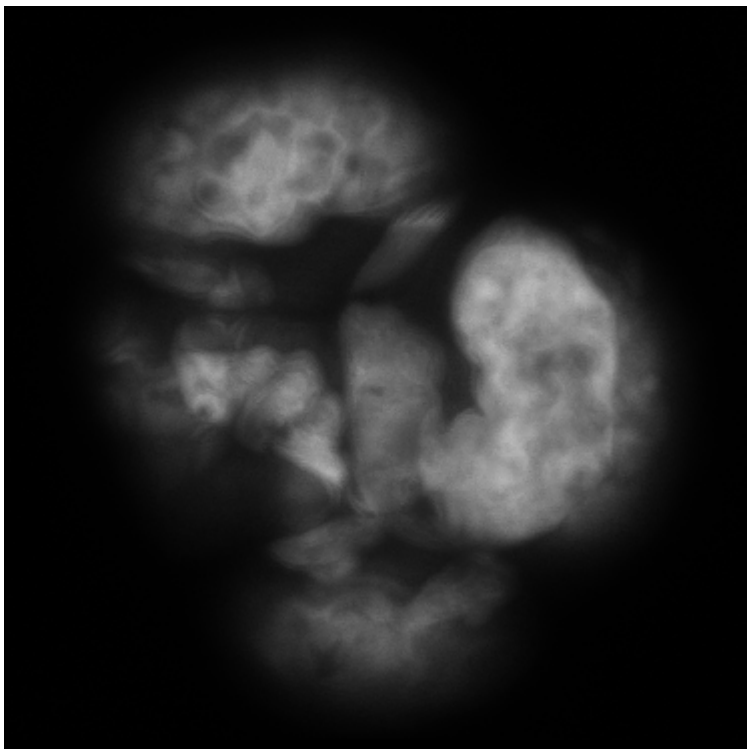

Figure 6: MedianProjectionImageFilter, projection along axe z

```
  typedef unsigned char PType;
  typedef itk::Image< PType, 3 > IType;
  typedef itk::Image< PType, 2 > IType2;

  typedef itk::ImageFileReader< IType > ReaderType;
  ReaderType::Pointer reader = ReaderType::New();
  reader->SetFileName( argv[2] );

  typedef itk::MaximumProjectionImageFilter< IType, IType > FilterType;
  FilterType::Pointer filter = FilterType::New();
  filter->SetInput( reader->GetOutput() );
  filter->SetAxe( axe );

  itk::SimpleFilterWatcher watcher(filter, "filter");

  filter->Update();

  IType::SizeType inputSize = filter->GetOutput()->GetLargestPossibleRegion().GetSize();

  typedef itk::ExtractImageFilter< IType, IType2 > ExtractType;
  ExtractType::Pointer extract = ExtractType::New();
  extract->SetInput( filter->GetOutput() );

  IType::SizeType size;
  for(int i=0; i<=3; i++) {
    if(i == axe) {
      size[i] = 0;
    } else {
      size[i] = inputSize[i];
     }
  }
  IType::IndexType idx;
  idx.Fill(0);

  IType::RegionType region;
  region.SetSize( size );
  region.SetIndex( idx );
  extract->SetExtractionRegion( region );

  typedef itk::ImageFileWriter< IType2 > WriterType;
  WriterType::Pointer writer = WriterType::New();
  writer->SetInput( extract->GetOutput() );
  writer->SetFileName( argv[3] );
  writer->Update();

  return 0;
}
```

## 6 Conclusion

The new classes give more feature and more flexibility with equivalent performance than the current available filter, but loss the compatibility with the previous filter. I propose to mark AccumulateImageFilter as deprecated in the next release of ITK to incite user to the new class.

## 7 Acknowledgments

We thank Dr Pierre Adenot and MIMA2 confocal facilities (http://mima2.jouy.inra.fr) for providing image samples.

## References

[1] L. Ibanez and W. Schroeder. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf, 2003.