RLEImage: run-length encoded memory compression scheme for an itk::Image

Release 1.0

Dženan Zukić[1], Matthew McCormick[2], Guido Gerig[3] and Paul Yushkevich[4]

August 19, 2016

[1]Kitware Inc., dzenan.zukic@kitware.com
[2]Kitware Inc., matt.mccormick@kitware.com
[3]NYU Tandon School of Engineering, Department of Computer Science and Engineering, gerig@nyu.edu
[4]Penn Image Computing & Science Laboratory, Department of Radiology, University of Pennsylvania, pauly2@mail.med.upenn.edu

Abstract

This document describes a new class, `itk::RLEImage`, which uses run-length encoding to reduce the memory needed for storage of label maps. This class is accompanied by all the iterators to make it a drop-in replacement for `itk::Image`. By changing the image typedef to `itk::RLEImage`, many ITK image processing algorithms build without modification and with minimal performance overhead. However, it is not possible if the user code uses `GetBufferPointer()` or otherwise assumes a linear pixel layout.

This class is implemented to reduce the memory use of ITK-SNAP (`www.itksnap.org`), so ITK-SNAP is the base for measuring the quantitative results.

The class, accompanying iterator specializations, automated regression tests, and test data are all packaged as an ITK remote module `https://github.com/KitwareMedical/ITKRLEImage`.

Latest version available at the Insight Journal [`http://hdl.handle.net/10380/3562`]

Contents

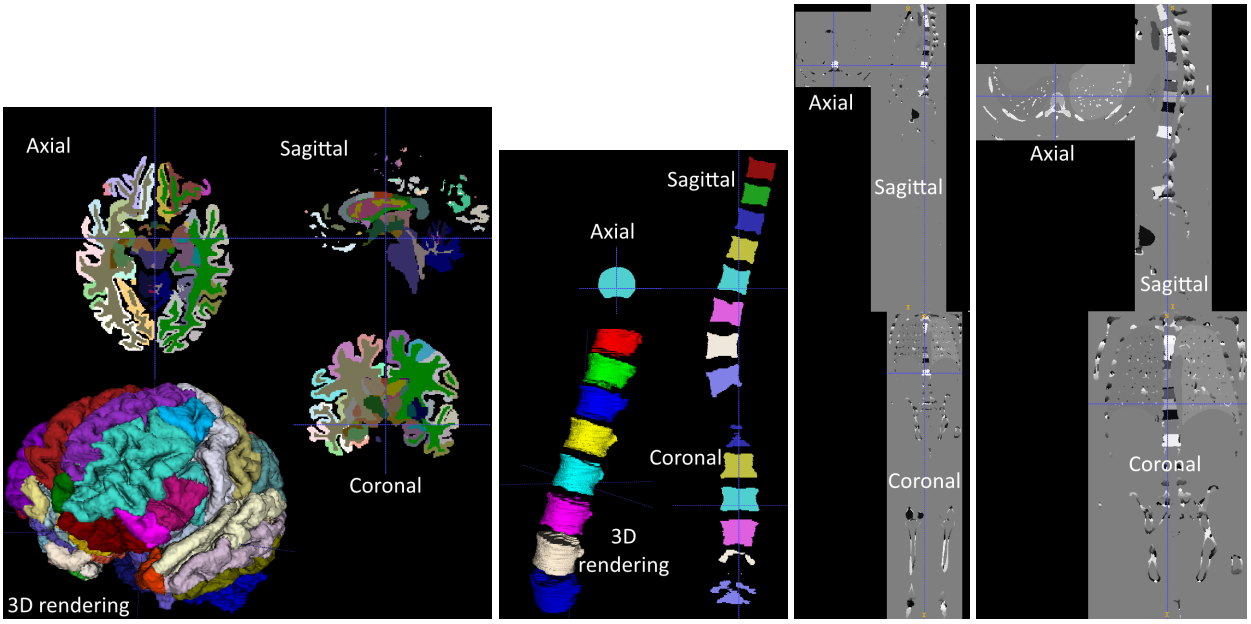

Figure 1: When there are pixels with the same value next to each other, it is an opportunity for compression. Left to right: brainParc (parcellation of a brain), vb-seg (segmentation of vertebral bodies), wb-seg (segmentation of a whole-body CT using watersheds algorithm), wb-crop (its shoulder-to-hips cropped version).

1 Introduction

Run-length encoding is used in many image compression applications [2]. Images with groups of pixels with the same value are best suited for such compression, and label maps (Fig. 1) are one such example.

Although `itk::LabelMap` [1] is well suited for computing statistics on a static label map (label map which will not change during the program execution), it uses a data structure which is not very suitable for either slice extraction (slicing) or pixel modification (editing). We therefore propose the new `itk::RLEImage` which addresses this issue by using a simpler data structure. That makes it not just more suited for on-the-fly editing, but also significantly faster for slicing (Tab. 2).

2 Data structure

n-dimensional RLEImage is implemented as $n-1$-dimensional `itk::Image` of run-length lines. A run-length line is implemented as `std::vector` of run-length segments. A run-length segment is a pair consisting of pixel value and its repetition count. A run-length line runs along the fastest chaning index, i.e. X-axis. C++ code:

```
typedef std::pair< CounterType, PixelType > RLSegment;
typedef std::vector< RLSegment > RLLine;
typedef typename itk::Image< RLLine, VImageDimension - 1 > BufferType;
typename BufferType::Pointer m_Buffer;
```

This data structure is quite simple and very similar to uncompressed image. Once a pixel at a given index is changed, there is not much searching needed or complex data structure to maintain like in `itk::LabelMap`.

This data structure is very convenient for slice extraction (slicing) across all axes except the first one: uncompressing a run-length line is about as fast as copying an already uncompressed row of pixels.

Unlike `itk::LabelMap`, all the standard iterators are specialized for `itk::RLEImage`: const/-writable × withIndex/without explicit index × plain/region/scanline. In addition to this, `itk::RegionOfInterestImageFilter` is specialized for Image→RLEImage, RLEImage→Image and RLEImage→RLEImage cases. If the RegionOfInterest is set to LargestPossibleRegion, the class effectively does only conversion between standard `itk::Image` and `itk::RLEImage`. If RegionOfInterest is smaller than LargestPossibleRegion, it really behaves as a classic region of interest filter. For RLEImage→RLEImage case, the `itk::RegionOfInterestImageFilter` can behave as a caster for template types for repetition counter and pixel values, e.g. cast `itk::RLEImage<short,short>` into `itk::RLEImage<long,char>`.

3 ITK-SNAP Performance measurements

ITK-SNAP [3] is a software application used to segment structures in 3D medical images. The authors' vision was to create a tool that would be dedicated to a specific function, segmentation, and would be easy to use and learn. ITK-SNAP is free, open-source, and multi-platform.

Since version 3.4, snap uses RLEImage for storing and editing segmentation layer. The interaction speed was not noticeable degraded, but memory consumption has significantly reduced (Tab 1).

	Image Size			**Million**	**Label**	**Commit Size [MB]**		
Image	X	Y	Z	**voxels**	**count**	Dense	RLE	**Difference**
None						154	165	**+7%**
vb-seg	640	633	299	116	9	731	508	**-31%**
wb-seg	512	512	1559	390	58672	3058	2267	**-26%**

Table 1: Memory consumption before addition of RLEImage (Dense) and after (RLE). From the first row (without any image loaded) it is clear that program itself takes more memory with version **3.4** than **3.2**. This is influenced by many number of factors, not just addition of RLE compression.

Scrolling through slices along different axes causes extraction of slices from the 3D image, called 'slicing'. Slicing is the most frequently executed operation, and its performance is very important. To take full advantage of `itk::RLEImage`, a slicing filter was specialized for it in ITK-SNAP. Comparison of slicing speed across the principal axes for different input images is given in Tab. 2.

These results show that slicing across the run-length lines is about twice slower than through uncompressed image. However, slicing along run-length lines is faster than the algorithm ITK-SNAP employed before, and not significantly slower than 'pure ITK' RoI filter. Additionally, it is a few times faster than slicing through `itk::LabelMap`.

4 Acknowledgments

This work is supported by NIH grant R01 EB014346, 'Continued development and maintenance of the ITK-SNAP 3D image segmentation software'.

Image		brainParc	vb-seg	wb-seg
Label count		114	9	58672
	X	256	640	512
Dimensions	Y	256	633	512
	Z	256	299	1559
		Slice extraction duration [ms]		
	X	51.0	105.9	993.6
itk::LabelMap	Y	39.2	71.5	912.1
	Z	39.0	94.8	874.7
	X	11.7	30.0	154.2
SNAP+RLE	Y	1.1	2.9	15.4
	Z	1.0	5.6	8.1
	X	5.2	22.5	73.9
SNAP-noRLE	Y	3.6	10.4	43.4
	Z	3.6	21.8	14.1
	X	10.3	26.3	79.1
pureITK	Y	0.5	0.6	1.4
	Z	0.4	0.7	0.6

Table 2: Slicing performance of `LabelMap` using `ChangeRegionLabelMapFilter`, ITK-SNAP with `itk::RLEImage`, ITK-SNAP with `Image`, and `Image` using `RegionOfInterestImageFilter`. All durations are in milliseconds. **SNAP-noRLE** was single-threaded, all the others used 4 processor cores.

References

[1] Gaëtan Lehmann. Label object representation and manipulation with ITK. *Insight Journal*, 08 2007. 1

[2] Wikipedia. Run-length encoding — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Run-length%20encoding&oldid=729616046, 2016. [Online; accessed 29-July-2016]. 1

[3] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006. 3