Bioimage informatics

**ABSTRACT**

**Summary:** ImgLib2 is an open-source Java library for n-dimensional data representation and manipulation with focus on image processing. It aims at minimizing code duplication by cleanly separating pixel-algebra, data access and data representation in memory. Algorithms can be implemented for classes of pixel types and generic access patterns by which they become independent of the specific dimensionality, pixel type and data representation. ImgLib2 illustrates that an elegant high-level programming interface can be achieved without sacrificing performance. It provides efficient implementations of common data types, storage layouts and algorithms. It is the data model underlying ImageJ2, the KNIME Image Processing toolbox and an increasing number of Fiji-Plugins.

**Availability:** ImgLib2 is licensed under BSD. Documentation and source code are available at http://imglib2.net and in a public repository at https://github.com/image/ImgLib2.

**Supplementary Information:** Supplementary data are available at Bioinformatics Online.

**Contact:** saalfeld@mpi-cbg.de

Received on July 6, 2012; revised on August 21, 2012; accepted on July 28, 2012

1 INTRODUCTION

Many algorithmic concepts from computer vision and image processing are applicable to the analysis of biological image data. However, re-using existing code is often difficult because it is implemented for a specific data type, limited image size or fixed number of dimensions, e.g. small 2d grayscale images. Biological imaging techniques generate images of varying dimensionality and a multitude of sample types (e.g. wavelength, frequency spectra, diffusion tensors) with varying precision. Improvements in imaging speed and resolution result in gigantic datasets that require well-designed strategies for data handling (e.g. tiled or compressed storage, streaming access). Writing code that is re-usable across many combinations of dimensionality, sample type and storage strategy is challenging and requires an appropriate abstraction layer.

We present ImgLib2, an open-source image processing framework that achieves code re-usability through a generic interface architecture that abstracts from dimensionality, sample type and storage strategy. It is highly extensible, providing developers with great flexibility in adding new sample types and image representations that will seamlessly work with existing algorithms, and vice versa. ImgLib2 shares basic concepts with the C++ frameworks ITK (Yoo et al., 2002) and Vigra (Koethe, 2000) for n-dimensional, generic image processing. It is the first framework that introduces generic programming to the Java image processing community (Preibisch et al., 2010). We chose Java for its simplicity and wide acceptance among biological researchers due to the popular image processing toolbox ImageJ (Rasband, 2012).

2 ARCHITECTURE

The ImgLib2 core design is based on three main concepts: Accessibles (i.e. images), Accessors and Types. We define an image as any mapping from a subset of n-dimensional Euclidean coordinate space to a generic pixel value type. Image properties are expressed by Accessible interfaces: coordinates can be either integer or real-valued, the coordinate domain can be either bounded or infinite, the image may support random access at arbitrary coordinates and/or iteration of all samples. Consider a conventional pixel image. It comprises samples of a specific value type in bounded n-dimensional space, arranged on an integer grid and is both random-accessible (at arbitrary integer coordinates) and iterable. Importantly, ImgLib2 supports concepts beyond the conventional pixel image, e.g. infinite, procedurally generated images or continuous images interpolated from sparsely sampled data.

Access to sample (pixel) values and coordinates is provided by Accessor interfaces. These exist in variants for integer and real coordinates, as well as iterating and random access. For iterating accessors, iteration order is subject to implementation, specialized for each memory layout to minimize access time.

Accessors provide value access via Types. ImgLib2 has a hierarchy of Type interfaces that describe algebraic properties of families of concrete types. Examples are Comparable types or NumericTypes that support basic arithmetic operations (+, −, *, /).

Access patterns and type properties allow fine-grained specification of algorithmic requirements. An algorithm that is built using appropriate interfaces applies to any specific image implementing those interfaces. Re-usability of algorithms is maximized by specifying them for the minimal set of required properties. Consider, for example, summing all pixel values in an image. This can be implemented in two lines of Java code for, e.g. a gray-level image stored as a byte[] array. However, it has to be re-implemented, over and over, for every combination of data...
implementations using image factories. This allows performance tuning for specific datasets without any modification to the algorithm implementation.

We compared the performance of ImgLib2 generic code and special purpose (fixed dimensionality and value type) implementations for Java primitive type arrays and ImageJ (Supplementary Table S1). For simple per-pixel operations, generic ImgLib2 code achieves 100% of the performance of special purpose implementations using native arrays. For a more complex operation involving an inner loop over the unknown number of dimensions, the ImgLib2 code was on average 1.6x slower than native arrays (1.5x slower than ImageJ). We consider this a reasonable abstraction penalty as the ImgLib2 code supports any dimensionality, image and value type. In contrast, native arrays and ImageJ images require specialized implementations for each supported dimensionality and value type. For the cases tested in our benchmark, this amounts to an order of magnitude increase in lines of code. Even so, only ImgLib2 is able to handle all test cases due to dimensionality and image size limits of both ImageJ and primitive type arrays.

ImgLib2 permits virtualization of sample access. We use this for accessors that perform on-the-fly coordinate and value transformations without copying the underlying data. The Views framework creates accessibles that provide coordinate-transforming accessors. Integer coordinate transformations include slicing, windowing, axes permutations and 90° rotations. Consecutive transformations are reduced and simplified, yielding accessors with optimal performance. For real coordinates we consider this a reasonable abstraction penalty as the ImgLib2 framework realizes a transparent transformation of values. For instance, a FloatType image can be addressed as a ListImg, an n-dimensional affine transformation. Interpolating and rasterizing views convert between discrete and continuous coordinate spaces. Finally, some algorithms (e.g. convolution) require access to pixels outside of the image which are usually created by padding or mirroring. This is achieved by extending views, whose accessors generate outside values on demand. Note, that views may be cascaded and act both as input and output for pixel processing. Similarly, the Converters framework realizes transparent transformation of values. For instance, a FloatType image can be addressed as ByteType using an arbitrary mapping function.

ImgLib2 uses Bio-Formats (Linkert et al., 2010) to read and write a large number of image file formats. Interoperability with
ImgLib2 is an open-source image processing framework that increases code re-usability by promoting generic implementations. It provides an abstraction layer that focuses on flexible and efficient image storage and access. The core paradigm is a clean separation of pixel algebra (how sample values are manipulated), data access (how sample coordinates are traversed) and data representation (how the samples are stored, laid out in memory or paged to disc). ImgLib2 relies on virtual access to both sample values and coordinates, facilitating parallelizability and extensibility.

ImgLib2 aims to connect software projects through an interface design that is easily adapted to existing data structures. ImgLib2 is the first image processing library available for Java that combines a flexible high-level programming interface with optimal performance. It enables developers of bioimage analysis software to focus on the design of complex algorithms instead of data management. Conversely, software engineers can develop efficient infrastructure without interfering with algorithm design. This becomes particularly interesting in the emerging field of bioimage informatics that is coping with the enormous amount of n-dimensional image data generated by recent developments in microscopy. Consequently, ImgLib2 is already being used by several high-profile projects of the Java bioimaging community (Berthold et al., 2009; Rueden et al., 2010; Schindelin et al., 2012). It is easily integrated into other projects providing an ideal basis for sharing interoperable, generic algorithms.

ACKNOWLEDGEMENTS

ImgLib2 and ImgLib have been supported by Fiji Hackathons at the MPI-CBG, EMBL and LOCI. We gratefully thank all developers and users that contributed, notably C. Rueden, B. DeZonia, C. Dietz, M. Horn, L. Kamentsky, A. Cardona, J. Schindelin, G. Harris, L. Lindsey, M. Longair, J.-Y. Tinevez, N. Perry, J. Funke and S. Jaensch.

Funding: T.P., P.T. and S.S. were funded by MPI-CBG, S.P. was funded by MPI-CBG and HHMI.

Conflict of Interest: none declared.

REFERENCES