An efficient and generic extension to ITK to process arbitrary shaped regions of interest

Jean-Baptiste Fasquel*, Vincent Agnus, Julien Lamy
IRCAD, 1 Place de l’Hôpital, F-67091 Strasbourg, France

ABSTRACT

The paper describes a software method to extend ITK (Insight Toolkit, supported by the National Library of Medicine), leading to ITK++. This method, which is based on the extension of the iterator design pattern, allows the processing of regions of interest with arbitrary shapes, without modifying the existing ITK code. We experimentally evaluate this work by considering the practical case of the liver vessel segmentation from CT-scan images, where it is pertinent to constrain processings to the liver area. Experimental results clearly prove the interest of this work: for instance, the anisotropic filtering of this area is performed in only 16 s with our proposed solution, while it takes 52 s using the native ITK framework.

A major advantage of this method is that only add-ons are performed: this facilitates the further evaluation of ITK++ while preserving the native ITK framework.

© 2005 Elsevier Ireland Ltd. All rights reserved.

1. Introduction

Nowadays, in modern medicine, there is a growing need for medical image analysis systems and softwares, to assist practitioners: computer-aided diagnosis, surgical planning, simulation, robotically assisted surgical intervention, etc. An important element of such systems and softwares is the image processing library which is expected to provide efficient algorithms allowing to automate and to improve medical data analysis. Therefore, it appears crucial to correctly design such a library to easily integrate new functionalities. This is furthermore required in this research field which is in permanent evolution and which involves a large community of scientists as many new efficient algorithms or pertinent alternatives are regularly proposed and published. To facilitate the integration of these developments, the code of image processing libraries must be generic enough to be easily reused and improved. For example, it appears highly convenient to write the code of an algorithm only once and to be able to use it whatever the spatial dimensionality of images (e.g. mammograms are 2D images, CT-scans are 3D images) or the analyzed image area (i.e. region of interest). Runtime performance is an additional constraint which must also be taken into account so that resulting systems are usable: fast diagnosis and patient modeling (e.g. for surgical planning in emergency), fast medical data analysis and registration for robot guidance during a surgical act (which can also include a system for heart beat or breathing compensation), etc.

The literature dedicated to the methodology for designing an image processing library seems to be particularly sparse. To our knowledge, only Köthe [1] recently published a detailed study dedicated to generic programming for image processing. However, it does not consider the genericity with respect to the region of interest. Among existing image processing libraries, the ITK library (Insight Segmentation and Registration Toolkit [2]), supported by the National Library of Medicine, appears to be a rich and well-designed framework because it provides a large and increasing number of sophisticated algorithms which can be applied, without code rewriting, whatever the spatial dimensionality and intensity (e.g. scalar or multi-
3. Extending ITK

Hereafter, we review fundamentals of ITK and present our contribution. The programming language considered is C++, and it is assumed that concepts such as design patterns [8], generic programming [9] and template mechanism [10] are known.

3.1. Fundamentals of ITK

ITK image processing algorithms are implemented in classes which are generally templated (i.e. parameterized) with respect to the input and the output image types. For example, the RecursiveGaussianImageFilter<TInputImage, TOutputImage> smooths an image of type TInputImage and stores the results within an image of type TOutputImage. Most ITK images, ImagePointType, Dim, are also defined through two template parameters: the intensity (PointType) and the spatial dimensionality (Dim). Thus, algorithm code becomes reusable whatever the medical image (2D mammograms, 3D CT scanner, multimodal images, etc.) it must process is managed by ITK iterators. An iterator is a special kind of object (a design pattern [8]) widely used in object-oriented programming to scan a dataset and access its elements. In our context, ITK iterators are used to access image points and are templated over the image type: the genericity of algorithms with respect to the image intensity and spatial dimensionality is then guaranteed. Among available iterators, the most widely used in algorithms are both variants of ImageRegionIterator and NeighborhoodIterator, and are limited to N rectangular ROIs. Only the NeighborhoodIterator provides neighborhood access (i.e. allow accessing neighbors of a given image point, which is required for algorithms such as filtering). The main drawback of ITK, leading to its lack of genericity, is that algorithms are not parameterized with respect to the iterator type: for each algorithm, the iterator type is hard-coded, which is highly critical. Indeed, due to the explicit definition of the iterator type (e.g. at code level) in the core of algorithms, it is a priori impossible to modify the ROI type (rectangular region, arbitrary shape one, etc.) without rewriting the code of the algorithm, which would be a tedious task due to the huge number of currently implemented algorithms.

To overcome this limitation, we propose to extend and parameterize both iterators with a new pattern. This extension, leading to ITK++, is hereafter described.

3.2. Proposed improvement: extending iterators

The design we propose consists of embedding an entity within an ITK-like iterator: this entity being more abstract than a simple rectangular region. This entity, which we propose to name scanner, will control the iterator movement over image points.
Fig. 1 – Illustration of the proposed iterator extension in pseudo-code. The iterator manages image scanning: the private member \texttt{currentPosition} indicates the current image point under analysis. The proposed entity (the private member \texttt{scanner}) manages the jump to perform to access the next image point from the current one, by invoking the function \texttt{operator++}.

For rectangular ROIs, this entity will be associated with a rectangular box. For arbitrary shaped ROIs, such as organs, this entity will store the list of image points to scan. Thus, when getting to the next point, the iterator will query the scanner to determine the required memory jump (or image point jump). Pseudo-code reported by Fig. 1 illustrates the use of the scanner embedded within an iterator.

3.3. Integration within ITK: ITK++

In order to integrate the scanner within ITK, we propose to modify the iterator behaviour according to the image type, becoming itself parameterized by the scanner type. Technically, this involves the creation of a new image type (Section 3.3.1) and the specialization of some ITK iterators (Section 3.3.2).

3.3.1. A new image type
The first entity that is created is the class \texttt{ImageWithScanner\lt\texttt{TImage},\texttt{Tscanner}\gt} which inherits from the native ITK \texttt{Image}, therefore involving minimal code writing. It can be seen as a classical ITK \texttt{Image} with an additional attribute: a scanner, the type of which being defined by the template parameter \texttt{TScanner}. Thanks to such inheritance, the \texttt{ImageWithScanner\lt\texttt{TImage},\texttt{Tscanner}\gt} is ensured to be compatible with all filters designed for the native ITK \texttt{Image}.

3.3.2. ITK iterator specialization

The behaviour of the most widely used image iterators must be specialized (i.e. rewritten) for the new image type (i.e. \texttt{ImageWithScanner}), in order to let the underlying scanner control the iterator movement. The resulting iterators remain similar to the native ones (to ensure compatibility), except that their movement is then managed by the underlying scanner provided by the new image.

Fig. 2 illustrates how both new image type definition and iterator specialization can make native ITK algorithms generic with respect to the region of interest.

In order to preserve the ITK pipeline functionality, we need to take care of the information propagation when executing connected ITK algorithms (see[2]for details concerning the ITK pipeline). For example, in the case of a downsampling algorithm, the output user-defined requested region of interest must be appropriately upsampled to evaluate the buffer area which must be available at the pipeline input.

4. Evaluation of the proposed design in a practical use case

This section describes the evaluation of the extension of the ITK iterators. We first present the experimental protocol, the implementation of the scanning mode and then the results.

4.1. Experimental protocol

In order to evaluate to proposed design, we consider a use case dealing with the segmentation of liver vessels from a
Fig. 3 – Integration of the iterator extension within ITK. ITK algorithms must be used with the new image type (i.e. `ImageWithScanner`), which is generic with respect to the type of the integrated scanner: by rewriting the behaviour of underlying iterators, the scanner controls the movement. Computations are restricted to a region of interest. Template parameters `PointType` and `Dim` are respectively related to the image point type (e.g. scalar or multimodal images) and its spatial dimension (2D, 3D, 4D, etc.).

CT-scan image, which is an important application in medical image analysis [7]. This is required for surgical planning, where it appears convenient to localize possible liver tumors with respect to the vessel network. We assume that the liver has been previously segmented.

We consider a segmentation procedure based on a simple image smoothing followed by an intensity thresholding. This processing flow has been implemented using the pipeline mechanism provided by the native ITK framework. Three kinds of classical image smoothing algorithms have been considered: anisotropic diffusion, mean filtering and morphological dilation. This procedure, which is clearly too trivial to lead to an efficient and robust segmentation, is used for illustration purposes.

As the analysis is focused on internal liver structures, the procedure should be applied only to voxels belonging to the arbitrary shaped area related to the liver, thus avoiding the unnecessary processing of external voxels. Using the proposed design, this can easily be performed. Fig. 4 gives a graphical representation of the code which has to be written using ITK++: a classical ITK pipeline is built and the region of interest is specified by a mask.

Using the native ITK design, the best way is to process all voxels belonging to the bounding box of the liver and then to intersect the result with the liver mask, as shown in Fig. 5. The native pipeline involves more operations than with the new design (although it is, of course, not so critical in this really trivial example): the bounding box must be computed and the result must be intersected with the initial mask. This proves that, from the computer scientist point of view, the proposed contribution can make procedure implementation easier.

In order to achieve a pertinent comparison, we propose to evaluate, in this particular use case, the relevance of our contribution by comparing the previously described procedure for two scanning modes. The first scanning mode concerns rectangular regions. This mode is native in ITK and has been implemented with our design for comparison purposes. The second mode we consider is related to the scanning of arbitrary shaped regions of interest. The implementation of these two modes in the ITK++ framework is shortly presented in the next section.

The size of the considered image is $512 \times 512 \times 80$ voxels. The bounding box related to the liver was a rectangular area of size $200 \times 197 \times 77 = 3,003,800$ voxels. It has been measured that the liver was defined by only 887,134 voxels, which is much less than the voxels belonging to its bounding box (about 3.4 times less).

The 1.8.0 release of ITK has been considered. A Linux based operating system with the gcc compiler version 3.3.4 has been used. Code source compilations have been done in release mode, with `-O3` option (highest optimization level). The computer used for experiments has an Intel Pentium 4 CPU (2.4 GHz) and 1 GB of RAM.

### 4.2. Implementation of two scanning modes

The entry which has been created for rectangular regions of interest is the class `BoxScanner`, and is generic with respect
Fig. 4 – Example of processing flow with the proposed contribution (ITK++): detection of liver vessels assuming the liver is already segmented. The original image is smoothed and thresholded. Processings are constrained to the arbitrary shaped region of interest defined by the liver mask in order to optimize runtime and internal liver vessel detection.

Fig. 5 – Example of processing flow with the native ITK: detection of liver vessels assuming the liver is already segmented. The original image is smoothed and thresholded. Processings can only be constrained to the bounding box of the liver, involving useless computations. By finally intersecting the result with the liver mask, internal liver vessels can be recovered.
ties. This is illustrated in Fig. 6. This aspect, which must be manage than in case of rectangular ROIs due to concavi-
aries. Nevertheless, from a practical point of view, this can be overcome by considering a safety margin large enough with respect to the kernel size of the neighborhood operations to be applied.

4.3. Experimental results

Table 1 reports experimental runtimes we measured for the three different image smoothing algorithms previously enounced, within the previously defined processing flow. For comparison purposes, we first reported runtimes obtained when processing the bounding box of the liver area using both the native ITK design and our design (with the ShapedScanner). We observe that our implementation leads to faster processings: this probably results from slight differences in code writ-
ing and compilation optimization. For this reason, it appears more relevant to illustrate the interest of our approach by com-
paring runtimes resulting from the same design (i.e. com-
paring runtimes obtained with both the ShapedScanner and the BoxScanner). It appears that computations can be clearly faster with arbitrary shaped ROIs. In this use case, a process-
ing constrained to the liver area is about 3.5 faster (on average) than the processing of its bounding box. We point out that run-
times related to overall processing flow are not given because additional computations (i.e. image thresholding, offset com-
putations from the liver mask, intersecting mask and thresh-
olded image) have been evaluated to be negligible (less than 1 s) with respect to the slowest step (i.e. the image smoothing). Experimental results clearly prove the interest of our contribu-
tion.

5. Discussion

As illustrated by the experimental results, the use of arbitrary shaped ROIs can allow to spare non-negligible time (which increases with algorithm complexity and data volume), which can be helpful for fast patient modeling. Similarly, in roboti-
cally assisted surgery, it can be convenient to directly focus on data of interest (e.g. detection of specific markers) to reg-
ister robot and real patient using medical data: computation time can be dramatically reduced, which is of great interest for heart beating and breathing compensation for instance. It must be underlined that the technical implementation of a s scanner is not unique. For example, in the case of arbi-
trary shaped regions of interest, another alternative could be to consider the shape to be scanned as an image mask: the iterator scans the image by skipping points which are outside the mask. The most efficient implementation depends on the shape and on the size of the area to scan: as the first method is memory consuming (one point can be associated to several

![Fig. 6 – Illustration of the ambiguity of data extrapolation at boundaries of an arbitrary shaped region of interest (grey area). Digital values represent intensities and *?* ambiguous values. A classical extrapolation has been considered in this example: the intensity of a given outside point (white area) is the one of the closest image point belonging to the ROI. Extrapolation can become ambiguous for outside points having several closest image points belonging to the ROI (i.e. concavities).](image)

Table 1 - Runtimes measured in seconds for the three different ITK based smoothing algorithms and for both the proposed design and the native ITK design

<table>
<thead>
<tr>
<th>Smoothing algorithm</th>
<th>Anisotropic (s)</th>
<th>Dilation (s)</th>
<th>Mean (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native ITK</td>
<td>65</td>
<td>7.23</td>
<td>7.88</td>
</tr>
<tr>
<td>ITK++ with BoxScanner</td>
<td>52</td>
<td>5.65</td>
<td>6.66</td>
</tr>
<tr>
<td>ITK++ with ShapedScanner</td>
<td>16</td>
<td>1.62</td>
<td>1.80</td>
</tr>
</tbody>
</table>

For comparison purposes, with our design, the processing of this rectangular area is based on the use of a BoxScanner. The Shaped-
scanner corresponds to the processing constrained to the liver area. Morphological dilation and mean filtering are performed using a neighborhood of size $5 \times 5 \times 5$. |
coordinates plus a memory jump, it should be preferred for small but widespread shapes, such as areas associated to the liver vascular system in medical image analysis. In such a case, the number of image points is relatively small with respect to the spatial extent of the anatomical structure. Moreover, the scan of irrelevant image points is avoided. Thanks to genericity, the proposed design can easily manage both implementations by just writing both behaviours and invoking the most relevant one. Thanks to the proposed framework, it is possible to design a specific scanner which will be automatically applicable to ITK algorithms.

Another point to discuss concerns the problem of data extrapolation, which has been shortly introduced previously although it is not the topic of this paper. In our sense, it would be interesting to achieve a specific study of possible extrapolation techniques and their impact on runtime and processing efficiency. A relevant solution could be to consider a distance transform based extrapolation: the value of ambiguous points is computed according to the distance with respect to their closest neighbours belonging to the ROI. Technically, this could be implemented at the iterator/scanner level, as it is already done in the native ITK for rectangular ROIs. In our sense, this additional work could be easily integrated within the proposed contribution.

The last point we propose to discuss concerns the limit of validity of our contribution: the proposed solution works only with ITK filters which do not hard-code neither the image type (differing from ImageWithBuffer) nor an iterator type differing from those which have been specialized. When the first situation occurs, as it is for example the case with the ITK anisotropic diffusion we used, the filter must be adapted by specializing it for our proposed new image type (however, this is not a tedious task). When the second situation occurs (we observed that it is rarely the case), the iterator must be specialized to support the new image type.

In our opinion, the proposed solution has the advantage of allowing to further evaluate the practical interest of such genericity while preserving the native behaviour of ITK (i.e., both versions are available). Furthermore, from a practical point of view, it minimizes add-ons and does not involve any modification of the native library, thus ensuring limited bugs and a stable behaviour of the native version.

6. Conclusion

In this paper, we proposed a solution to improve the genericity of the powerful tool ITK in order to manage regions of interest. Therefore, we defined a new entity, that we named scanner, dedicated to the parameterization of the set of image points to be processed when applying an algorithm to a medical image. We implemented this design within ITK and proved the interest and potential of our approach. Processings can be restricted to any user-defined ROI without wasting time by considering unwanted parts of images. As underlined, this makes processings faster and more efficient, which is becoming nowadays crucial for many medical systems and especially those dealing with real-time processing (e.g., robot guidance, heart beat compensation, etc.). This work would be useful for medical image processing software/system users and developers in terms of runtime performances and segmentation accuracy. It must be pointed out that there are still some filters which must be adapted to support our add-ons. The next step is to validate our approach on more algorithms and complex processing flows, and to evaluate the impact (in terms of runtime) of extrapolators in the case of arbitrary shaped regions of interest.

Acknowledgements

First of all, we are very grateful to IRCADs computer science and robotics research and development team, and in particular G. Broder, J. Moreau, N. Papier and S. Thery for their pertinent remarks concerning programming techniques and methodology. We would also like to thank the French National League against Cancer, the Alsace Region, the French Research Ministry and the French Agency for Innovation for their financial support. Finally, we would like to thank all our medical partners, who have provided us medical images of patients, and in particular Pr. Afshin Gangi and Pr. Catherine Roy from the radiological department of the Civil Hospital in Strasbourg. We also thank the surgical team of Professor Maescaux for their help in our works, in particular Pr. Matter, Pr. Leroy, Dr. Vix.

References