A Framework for Algorithm Evaluation and Clinical Application Prototyping using ITK

Jan Rexilius[1], Wolf Spindler[1], Julien Jomier[2], Matthias König[1],
Horst K. Hahn[1], Florian Link[1], and Heinz-Otto Peitgen[1]

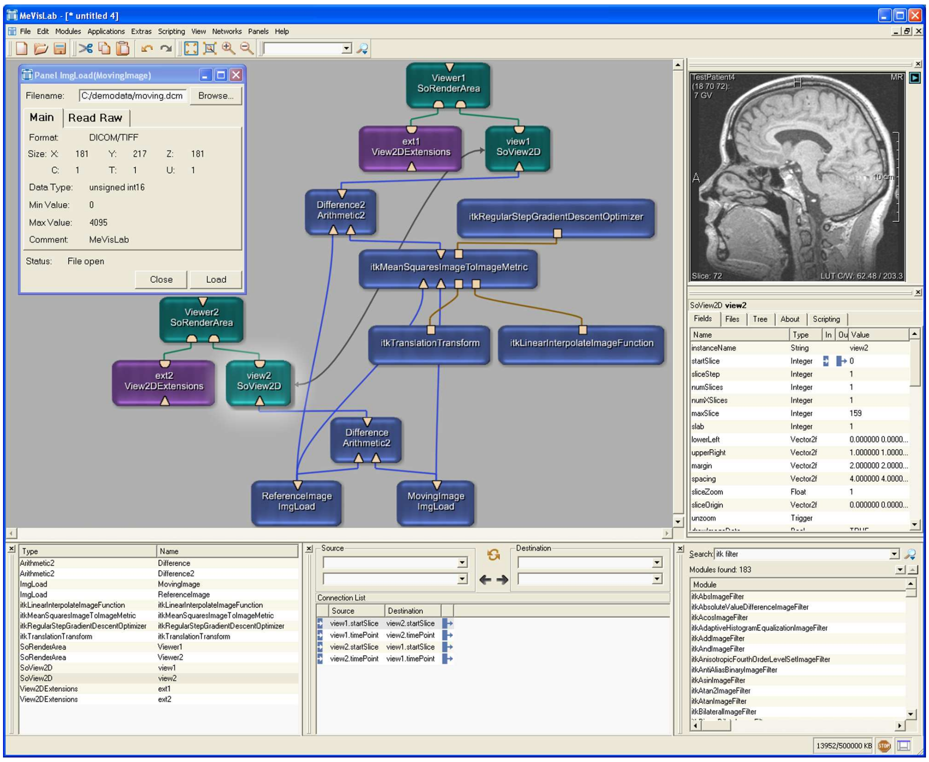
[1]MeVis, Center for Medical Diagnostic Systems and Visualization, Bremen, Germany
[2]CADDLab, University of North Carolina, Department of Radiology, USA
email: rexilius@mevis.de

Abstract. The usage of image processing has become an important and continuous growing field in medical imaging. Although a large amount of algorithms is available, only few are routinely applied within applications for clinical diagnosis support and treatment planning. New tools are required that allow for efficient evaluation and comparison of algorithms as well as the generation of dedicated user interfaces with sophisticated visualization for clinical applications. We propose a generic integration of ITK algorithms into the visual programming and rapid prototyping platform MeVisLab. One of the main limitation of ITK is its lack of visualization and interaction functionality through a dedicated development platform. MeVisLab provides a modular visual programming interface with a comprehensive suite of image processing and visualization tools. The linkage of ITK algorithms to MeVisLab is based on a two step process that consists of a generic XML description for each ITK algorithm and an automatic code generator to create new MeVisLab modules, which will be made publicly available within the free version of MeVisLab. Each module can handle different image types and data dimensions dynamically at runtime. Example applications using registration, segmentation, and visualization modules are presented in order to show the capabilities of the described integration concept.

1 Introduction

Medical image analysis and visualization are rapidly growing fields that have gained importance in several medical disciplines. With an ever growing amount of data acquired by medical imaging devices, the variety and complexity of post-processing algorithms and the need for an interactive handling is widely increasing. Therefore, a common software framework containing a large set of standard as well as advanced algorithms is an indispensable tool. The Insight Segmentation and Registration Toolkit (ITK) [1], with the first official public release in 2002, has since then become one of the most popular libraries for the analysis of medical image data, and is already being used in a wide range of different applications (see e.g. [2]).

An important requirement for the development of novel image analysis techniques, their evaluation, and for their routinely applicability in a clinical setting is a user-friendly handling through a dedicated user interface. Typically, this addresses several generic problems irrespective of the specific application, including the efficient support of image import and export using medical image format standards such as DICOM, multi-dimensional visualization, user interaction, and image analysis. All of these tasks usually take a substantial amount of development time before any usable application can be delivered to practitioners or specialists. New tools are required that allow for an efficient evaluation and comparison of algorithms, and the generation of sophisticated user interfaces for applications used in clinical diagnosis support and treatment planning. Today, no platform is available that satisfactorily combines the algorithmic strengths of ITK with a modular visual programming interface plus a rapid application prototyping environment.



Filename: C:/demodata/moving.dcm

Format: DICOM/TIFF

Size: X: 181 Y: 217 Z: 181
C: 1 T: 1 U: 1

Data Type: unsigned int16

Min Value: 0

Max Value: 4095

Comment: MeVisLab

Status: File open

Type	Name
Arithmetic2	Difference
Arithmetic2	Difference2
ImgLoad	MovingImage
ImgLoad	ReferenceImage
itkLinearInterpolateImageFunction	itkLinearInterpolateImageFunction
itkMeanSquaresImageToImageMetric	itkMeanSquaresImageToImageMetric
itkRegularStepGradientDescentOptimizer	itkRegularStepGradientDescentOptimizer
itkTranslationTransform	itkTranslationTransform
SoRenderArea	Viewer1
SoRenderArea	Viewer2
SoView2D	view1
SoView2D	view2
View2DExtensions	ext1
View2DExtensions	ext2

Source	Destination
view1.startSlice	view2.startSlice
view1.timePoint	view2.timePoint
view2.startSlice	view1.startSlice
view2.timePoint	view1.timePoint

SoView2D **view2**

Name	Type	In	Ou	Value
instanceName	String			view2
startSlice	Integer			0
sliceStep	Integer			1
numSlices	Integer			1
numXSlices	Integer			1
maxSlice	Integer			159
slab	Integer			1
lowerLeft	Vector2f			0.000000 0.0000...
upperRight	Vector2f			1.000000 1.0000...
margin	Vector2f			2.000000 2.0000...
spacing	Vector2f			4.000000 4.0000...
sliceZoom	Float			1
sliceOrigin	Vector2f			0.000000 0.0000...
unzoom	Trigger			

Search: itk filter

Modules found: 183

Module
itkAbsImageFilter
itkAbsoluteValueDifferenceImageFilter
itkAcosImageFilter
itkAdaptiveHistogramEqualizationImageFilter
itkAddImageFilter
itkAndImageFilter
itkAnisotropicFourthOrderLevelSetImageFilter
itkAntiAliasBinaryImageFilter
itkAsinImageFilter
itkAtan2ImageFilter
itkAtanImageFilter
itkBilateralImageFilter

Fig. 1. Development platform MeVisLab [3] with an example network that combines ITK and MeVisLab modules. Different colors denote image processing, visualization, and macro modules. Several inspectors such as a module search interface and detailed module and connection inspectors are shown. Furthermore, the module panel of a basic load operator is shown.

A number of dedicated software platforms for medical and scientific image analysis such as Analyze [4], SCIRun [5], VolView [6], or [7] are available that integrate ITK to some extent within their scope of supply. However, a full or generic integration of ITK as well as a simple and user-friendly or even automatic wrapping of ITK functionality is still missing. Thus, an update to a new ITK version that contains interface changes or new algorithms might be tedious.

Besides several software platforms, a number of medical image processing libraries, such as the Medical Imaging ToolKit or the Medical Imaging Interaction Toolkit (MITK), are available from different research groups [8, 9]. They combine features of ITK and a visualization library such as VTK [10] in a unified framework. Since any graphical user interface is usually missing within these tools, a long period of vocational training is required for the user. Moreover, the usage of these libraries requires advanced C++ development skills and in-depth knowledge of ITK, the used visualization library, and of the actual wrapping library.

The aim of this paper is to introduce a seamless integration of ITK into the flexible research and development platform MeVisLab [3]. To this end, we propose a code generator that automatically creates MeVisLab modules from ITK algorithms such that they are accessible per drag and drop for visual programming. Most ITK algorithms are incorporated into the software, including image to image filters, the flexible and extendable framework for image to image registration, and level set methods. The capability of our integration is shown by means of example applications.

2 Methods

2.1 The research and prototyping platform MeVisLab

MeVisLab combines a number of design concepts for rapid prototyping. The multiplatform software (MS Windows, Linux) provides a comprehensive suite of general image processing and visualization tools for multi-dimensional medical images as well as advanced methods for specific organs and clinical tasks.

Image Processing. An integral part of MeVisLab is the object-oriented MeVis Image Processing Library (ML), that provides a general framework for image processing. Each algorithm is represented as a self-descriptive module inside the development environment. These functional units can be arbitrarily combined to form a network via an intuitive graphical user interface (see Fig. 1 and 2 (a) for examples). Image processing is done in a strictly request-driven way using sophisticated paging, caching, and multithreading strategies. Various concepts for local, global, and kernel image processing are supported, as well as special assistance for global access to large images. Currently, more than 200 image processing modules are available, including image filtering, segmentation, and statistical analysis.

Visualization. For visualization and interactive graphics programming, the open source Open Inventor 3D visualization toolkit [11] is fully integrated into MeVisLab. Based on a 3D scene graph, Open Inventor includes a rich set of objects such as geometrical primitives, polygons, text, materials, cameras, lights, trackballs, handle boxes, 3D viewers, and editors. Furthermore, additional functionality has been added, including state-of-the-art volume rendering using 2D and 3D textures, customizable 2D and 3D viewers and annotations, and advanced MPR techniques.

Extensibility and Application Development. New image processing algorithms and visualization tools can be easily integrated as new C++ modules using a standardized software interface. Module wizards as well as an interface for customized wizards add further usability. Macro modules that allow for a hierarchical encapsulation of networks can be used to easily reuse available developments. Efficient design of graphical user interfaces can be achieved by an abstract, hierarchical module definition language (MDL), hiding the complexity of the underlying module network to the end user (Fig. 2). Dynamic functionality on both the network and the user interface level can be added using JavaScript 2.0 or Python. MeVisLab can be fully integrated into the radiological workflow (PACS and DICOM integration). See [3] and the provided documentation therein for further details.

2.2 The Insight Segmentation and Registration Toolkit (ITK)

The open source and cross-platform toolkit ITK is an object-oriented library, providing an extensive collection of C++ classes for image segmentation and registration of medical image data. An automated wrapping process for other programming languages such as Tcl and Python is currently under development. One of the core concepts in ITK is generic programming using the template programming mechanism, which provides both developer and user with great flexibility. Similar to MeVisLab, a page-based processing of a data pipeline is used. Due to the strict partition between algorithms and user interface, no support for visualization or graphical user interface design is available within ITK. See [1] and [12] for further details.

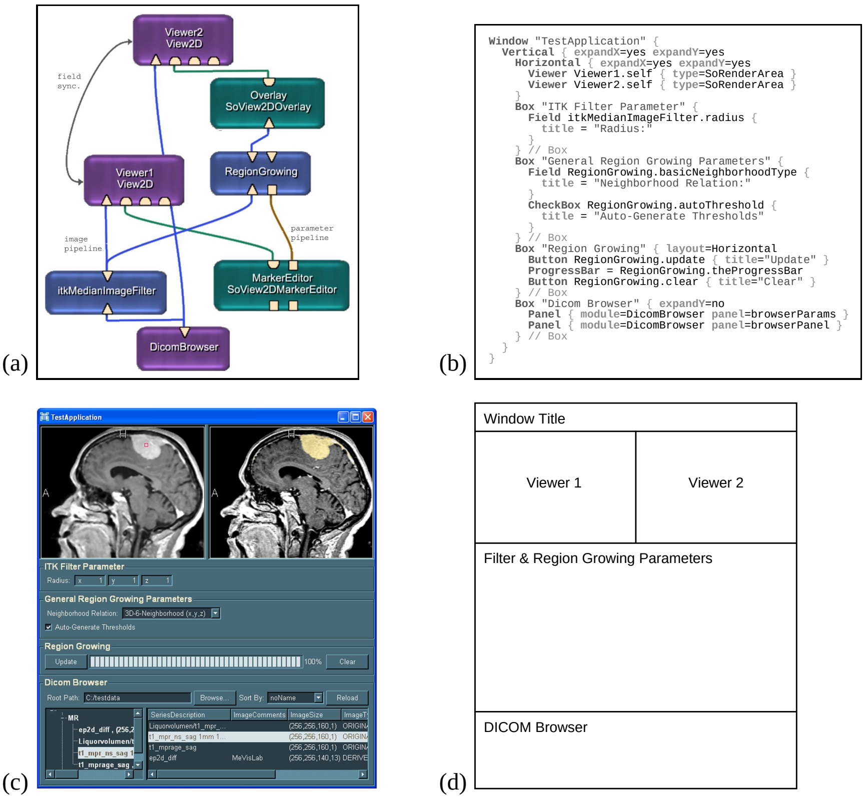
2.3 Integrating ITK into MeVisLab

We propose a framework that integrates the advanced image analysis techniques of ITK into the MeVisLab programming environment. Using the original class name as module name, each ITK algorithm can be clearly distinguished from other MeVisLab modules by its itk-prefix. The generation of a new MeVisLab module from an ITK filter consists of two steps.

1. XML description of ITK algorithms. In a first step, a detailed description of the targeted ITK algorithm is automatically generated and saved into an XML structure. The current coding styles suggested by the ITK project [1] are used for a contextual parsing of files. This is especially difficult, since not all ITK algorithms strictly follow these guidelines yet. Various types of algorithms are integrated such as image filters, optimizers, or interpolators. The structure can be largely divided into a basic class description and a parameter description section.

(i) General algorithm description:
- Number and type of input/output image
- Algorithm type, e.g. image filter, interpolation, metrics, or optimization
- Data type limitations, e.g. only integer or float
- Superclass hierarchy

(ii) Parameter description:
- Names and get/set methods
- Types and default values



```
Window "TestApplication" {
  Vertical { expandX=yes expandY=yes
    Horizontal { expandX=yes expandY=yes
      Viewer Viewer1.self { type=SoRenderArea }
      Viewer Viewer2.self { type=SoRenderArea }
    }
    Box "ITK Filter Parameter" {
      Field itkMedianImageFilter.radius {
        title = "Radius:"
      }
    } // Box
    Box "General Region Growing Parameters" {
      Field RegionGrowing.basicNeighborhoodType {
        title = "Neighborhood Relation:"
      }
      CheckBox RegionGrowing.autoThreshold {
        title = "Auto-Generate Thresholds"
      }
    } // Box
    Box "Region Growing" { layout=Horizontal
      Button RegionGrowing.update { title="Update" }
      ProgressBar = RegionGrowing.theProgressBar
      Button RegionGrowing.clear { title="Clear" }
    } // Box
    Box "Dicom Browser" { expandY=no
      Panel { module=DicomBrowser panel=browserParams }
      Panel { module=DicomBrowser panel=browserPanel }
    } // Box
  }
}
```

ITK Filter Parameter

Radius: x 1 y 1 z 1

General Region Growing Parameters

Neighborhood Relation: 3D-6-Neighborhood (x,y,z)

☑ Auto-Generate Thresholds

Region Growing

Update 100% Clear

Dicom Browser

Root Path: C:/testdata Browse... Sort By: noName Reload

MR
ep2d_diff , (256,2
Liquorvolumen/t
t1_mpr_ns_sag 1
t1_mprage_sag ,

SeriesDescription	ImageComments	ImageSize	ImageT
Liquorvolumen/t1_mpr_...		(256,256,160,1)	ORIGIN
t1_mpr_ns_sag 1mm 1...		(256,256,160,1)	ORIGIN
t1_mprage_sag		(256,256,160,1)	ORIGIN
ep2d_diff	MeVisLab	(256,256,140,13)	DERIVE

Fig. 2. Example application that combines both ITK and MeVisLab algorithms. The application uses a seeded region growing algorithm to generate a tumor segmentation. Input image data can be selected via a DICOM browser tool. (a): Module network and (b) complete description of user interface via MDL. (c): Screenshot of corresponding graphical user interface and (d) its schematic representation.

Due to the generic programming approach in ITK, it was important to extend the search for a correct parameter type or missing parameters to all superclasses of the class hierarchy. Furthermore, a dedicated analysis of the methods of a class has been implemented in order to find all available parameters and their default values. The resulting XML description is independent of the image processing platform, and could be used in other systems as well.

2. Module generation. In a second step, the XML based algorithm description is used as input of an automatic code generation process. Depending on the type and implementation of an ITK algorithm, various paging and caching strategies can be used. Since images in ITK are templated over the data type and the spatial dimension, code instances have to be generated separately for each type and dimension. Similar concepts are also established for other data objects of the ITK project. Thus, an important feature of the developed integration process is the coverage of different available image types and data dimensions within a single generated MeVisLab module. Our current implementation includes 8, 16, and 32 bit scalar data types (signed/unsigned, float, double) as well as complex data structures such as vectors. Image dimensions from 2D to 6D are supported.

Each resulting module will appear as a regular, self-descriptive module in MeVisLab, containing a number of input and output fields as well as further parameter fields. This allows the utilization inside the visual programming environment without limitations. Thus, a user can easily select and combine ITK and MeVisLab modules and interactively change parameter settings, while the correct precompiled ITK algorithm instance is dynamically selected at runtime. This adds new flexibility to ITK on the one hand and greatly extends the MeVisLab functionality by a set of new algorithms on the other hand.

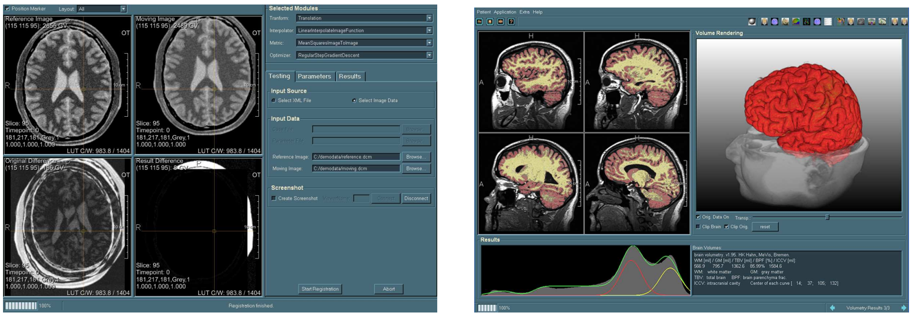
Besides image to image filters, further components such as the generic registration framework, level set algorithms and read/write methods of ITK are integrated. Additionally, exception handling, status, and tracing functionality of ITK is fully linked to MeVisLab. An example network combining a set of different ITK registration components is shown in Fig. 1.

3 Results

We successfully integrated most of the ITK toolkit functionality into the MeVisLab environment in a generic fashion. At the moment about 180 modules from ITK algorithms are available, ca. 50 of them as part of the registration framework (transforms, interpolators, metrics, optimizers). Due to a visual programming interface, no specific ITK programming expertise is required to use these algorithms. Each algorithm is available as a module via an intuitive user interface inside the software development environment. Modules for image processing and visualization can be combined with ITK algorithms to form a network via simple field connections. Parameter fields can be synchronized by drag and drop as well, e.g., in order to synchronize slicing through original and segmented datasets, as shown in Fig. 2 (a). Consequently, algorithms based on examples described in the ITK Software Guide [1] can be analyzed by selecting and connecting the required modules along with an appropriate visualization functionality from the MeVisLab module database. Networks for more than 80 algorithms have already been added, and a direct link to the online web pages of each ITK algorithm has been established.

In order to show the potential of our approach, Fig. 2 shows an exemplary application that incorporates algorithms of both ITK and MeVisLab. Different levels of detail (C++, network, and user interface level) are addressed, allowing for an efficient encapsulation of components. Fig. 2 (c) shows the graphical user interface of a segmentation application, based on the module network shown in Fig. 2 (a). The corresponding user interface description is given in Fig. 2 (b).

Besides an utilization of ITK within a visual programming environment, a further objective of this work was to provide a tool for the evaluation of algorithms, and for the development of application prototypes that can be integrated into a clinical workflow, respectively. To this end, two different applications are subsequently presented.



Tranform: Translation
Interpolator: LinearInterpolateImageFunction
Metric: MeanSquaresImageToImage
Optimizer: RegularStepGradientDescent

○ Select XML File ● Select Image Data

Reference Image: C:/demodata/reference.dcm
Moving Image: C:/demodata/moving.dcm

□ Create Screenshot

Registration finished.

☑ Orig. Data On
□ Clip Brain ☑ Clip Orig.

```
brain volumetry. v1.95. HK Hahn, MeVis, Bremen.
WM [ml] / GM [ml] / TBV [ml] / BPF [%] / ICCV [ml]:
566.9     795.7     1362.6    85.99%   1584.6
WM:   white matter              GM:  gray matter
TBV:  total brain    BPF: brain parenchyma frac.
ICCV: intracranial cavity      Center of each curve [  14;   37;   105;   132]
```

Volumetry Results 3/3

Fig. 3. Combining ITK and MeVisLab for the generation of flexible test environments and rapid application prototyping. (LEFT) Test environment for evaluation of ITK image registration algorithms. The data were obtained from the BrainWeb project [13]. (RIGHT) Application prototype for semi-automatic brain volume quantification used for diagnosis support and therapy monitoring.

3.1 Test environment for algorithm evaluation

The ITK toolkit contains a continuous growing set of various image processing algorithms. One of its exceptional features is a general framework for image to image registration, that provides a flexible combination of transforms, interpolators, metrics, and optimizers, depending on the analyzed problem [1, 12]. Fig. 1 shows the realization of this registration suite in MeVisLab.

Besides a general registration framework, the support of a comparison between combinations of different techniques within a dedicated software tool is also desirable. The utilization of a visual programming and rapid prototyping platform provides a powerful way to generate such a test environment as shown in Fig. 3 (left).

3.2 Rapid application prototyping

Modern medical workstations offer a wide range of tools for manipulating and visualizing images in two, three and four dimensions. However, customized solutions are often required, and new algorithms for medical image analysis need a systematic evaluation within a clinical workflow. Therefore, a rapid prototyping environment that allows for various levels of detail depending on the purpose of the prototype is a valuable tool. Fig. 3 (right) shows an example of an application prototype for semi-automatic brain volume quantification developed with MeVisLab. New image data can be imported into the system via a configurable DICOM browser interface. A structured reporting functionality is available for a convenient case management of the different application steps.

4 Discussion and Conclusion

In this work, a generic integration of the open source toolkit ITK into a flexible visual programming platform is proposed, suited for the efficient development of test environments and rapid prototyping of clinically valuable applications. Based on an extensive

XML description, a sophisticated linkage of ITK algorithms to the platform is generated. Our approach allows for an arbitrary combination of modules and various data types without any recompilation, and the ability of efficient graphical user interface generation. Different parameter settings can be interactively explored. Algorithms from upcoming ITK release versions can be easily integrated.

Future work will focus on an automatic integration of all not yet supported ITK algorithms. We further plan to integrate VTK into MeVisLab, using a similar approach for module generation as presented in this work for ITK. Our integration will be made publicly available as a package of the free version of MeVisLab.

References

1. Ibanez, L., Schroeder, W., Ng, L., Cates, J.: The ITK Software Guide. Kitware Inc. (2003)
2. Guyon, J.P., Foskey, M., Kim, J., Firat, Z., Davis, B., Aylward, S.: VETOT, Volume Estimation and Tracking Over Time: Framework and Validation. In: MICCAI. (2003) 142–149
3. MeVisLab 1.2, homepage at: (http://www.mevislab.de)
4. Augustine, K., Holmes, D., Robb, R.: ITK and Analyze: A synergistic integration. In: SPIE, Medical Imaging. (2004) 6–15
5. Johnson, C.R., MacLeod, R.S., Parker, S.G., Weinstein, D.M.: Biomedical Computing and Visualization Software Environments. Communications of the ACM **47** (2004) 64–71
6. VolView homepage at Kitware: (http://www.kitware.com/products/volview.html)
7. Leemput, K., Hämäläinen, J.: A Cross-Platform Software Framework for Medical Image Processing. In: MICCAI. (2004) 1091–1092
8. Wolf, I., Vetter, M., Wegner, I., et al: The medical imaging interaction toolkit (MITK): a toolkit facilitating the creation of interactive software by extending VTK and ITK. In: SPIE, Medical Imaging. (2004) 16–27
9. Zhao, M., Tian, J., Zhu, X., et al: The Design and Implementation of a C++ Toolkit for Integrated Medical Image Processing and Analysis. In: SPIE, Medical Imaging. (2004) 39–47
10. Schroeder, W., Martin, K., Lorensen, B.: The Visualization Toolkit, 3rd edition. Kitware Inc. (2003)
11. Wernecke, J.: The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2. Addison-Wesley Professional (1994)
12. Yoo, T.S.: Insight into Images: Principles and Practice for Segmentation, Registration, and Image Analysis. A K Peters, Ltd. (2004)
13. Collins, D., Zijdenbos, A., Kollokian, V., et al.: Design and Construction of a Realistic Digital Brain Phantom. IEEE TMI **17** (1998) 463–468