Cardiac Interventional Guidance using Multimodal Data Processing and Visualisation: medInria as an Interoperability Platform

Florian Vichot, Hubert Cochet, Benoit Bleuzé, Nicolas Toussaint, Pierre Jaïs, Maxime Sermesant

Cardiac Interventional Guidance using Multimodal Data Processing and Visualisation: medInria as an Interoperability Platform

Florian Vichot[1], Hubert Cochet[2], Benoît Bleuzé[1], Nicolas Toussaint[1,3], Pierre Jaïs[2] and Maxime Sermesant[1]

August 14, 2012

[1]INRIA, 2004 route des lucioles, Sophia Antipolis, France
[2]CHU Haut Levêque, Bordeaux, France
[3]King's College London, UK
florian.vichot@inria.fr
http://med.inria.fr

Abstract

medInria is a free medical imaging software developed at Inria, which aims at providing clinicians with state-of-the-art algorithms dedicated to medical image processing and visualization. Efforts have been made to simplify the user interface, while keeping high-level algorithms. In this particular article, we will concentrate on its use in preoperative preparation for cardiac interventions, and how we handle the difficulties arising from the lack of standard format for data types such as meshes or fibers, the absence of a common programming interface for data processing algorithms, notably registration, and the issues of visualisation where display conventions would be beneficial.

Contents

1 Introduction

The ever-increasing presence of technology in the hospital means more and more data has to be visualised, analysed and shared. This situation has raised the need for developing standards between medical actors (equipment manufacturers, software providers, etc). Some success has been achieved with initiatives such as DICOM for medical imaging, or HL7 for medical records, but in other areas such as mesh or tensor data, there is still a lack of accepted standards. Moreover, it is not just data exchange that could benefit from the emergence of standards: the analysis of medical data often involves state-of-the-art algorithms, that are often only available in complex software and that require a rewrite everytime they need to be integrated into a new software, with each time a significant time investment, as well as a risk of errors, or of a loss in performance. Finally, the absence of clear conventions in terms of visualisation is also problematic, as for example there is no standard for the colors scale of lookup tables used to display attributes of mesh files, or the choice of color to represent directions in a tensor image. All of these make preoperative data analysis more complex for clinicians, and represent an increased time cost, as well as an increased risk of errors.

medInria tries to bridge these gaps in different fashions. In terms of data format, our approach is to support a broad range of formats for import and export, and to offer the possibility of exporting into a simple "composite" format that should be easily implemented by other software, while retaining some backward compatibility in software that will not support it. Regarding algorithm re-use, we propose the `Registration Programming Interface`[3], which is a generic programming interface for most registration algorithms. We now provide jointly a simple and intuitive graphical interface shared by all registration methods in medInria, that greatly simplifies the user experience while helping the author of a new registration method to focus on the algorithm, not on the user interface. Finally, on the subject of conventions in data visualisation, we discuss potential evolutions of medInria to cope with the varied ways of displaying the same datasets, to ease the work of clinicians in comparing them.

2 medInria Architecture

Nowadays, software packages for the visualisation and processing of medical images are numerous. So why yet another one? We felt that none addressed both the specific needs of clinicians, by providing an ergonomic, efficient interface, and the needs of researchers, by enabling easy addition of functionalities. While many medical imaging software packages exist, they are all too often designed only for one of these two applications. For example, they often provide a very low-level abstraction to algorithms, which requires some expertise from the users to achieve the desired results. Clinicians needs are simple yet very difficult to satisfy in a computer science point of view: they want ergonomic, reactive and intuitive software that offers real-time interactions with data.

medInria is built in C++ using proven components: for our visualisation needs, we use VTK[5], the Visualisation Toolkit, an open source, freely available visualisation framework designed for efficient data management and visualisation that is used by many other software, such as 3D Slicer[2] or GIMIAS[1]. The image processing is done using ITK[4], the Insight Segmentation and Registration Toolkit, another open source project. DICOM data is loaded using DCMTK[8]. Finally, we use the Qt library for the user interface and the system abstraction, allowing an ergonomic finish to medInria, as well as providing us with cross-platform compatibility.

The software uses a plugin architecture. This allows us to limit the dependencies on the core software, while enabling us to bring in new algorithms and components easily (Fig. 1), as well as shipping version of medInria with just the necessary functionalities to clinicians, to avoid feature bloat.

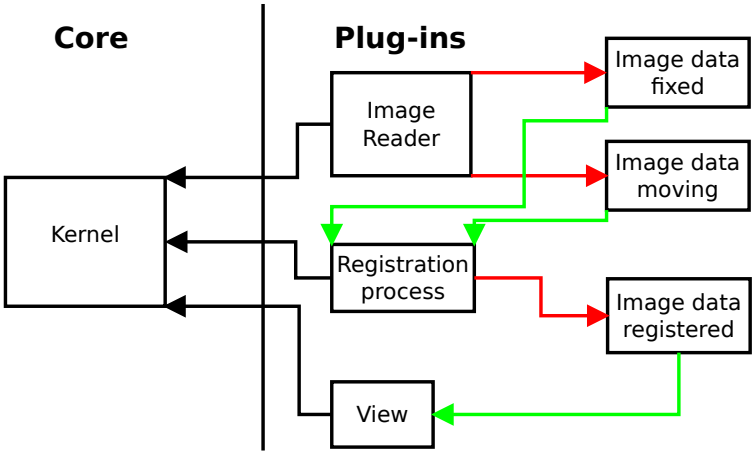

Figure 1: General architecture, here demonstrating an image reader, a view, and a registration plugin working together. (inputs are green, output red)

It is currently at its version 2.0.1, released in April 2012, that builds on the success of the previous versions[9] while also gaining in functionalities. We're currently working on developping an SDK that will allow third-parties to extend medInria via plugins, to support new data types, new visualisations, new algorithms or even add whole new workspaces. The license of the SDK will be very flexible, allowing either open-source or binary plugins. We plan to make it as simple as possible to write plugins, to encourage the research community to write their own.

medInria is freeware for non-commercial use, but we are considering eventually releasing it as open-source.

3 Data Formats

The first issue we have to tackle when we want to provide a software usable by clinicians during the pre-operative phase of an intervention, is to be able to import all the necessary data he will need to establish a diagnosis and identify the pathological area to ablate, for example. We then have to provide means to export the result of that preoperative process in the right format, so it can be loaded into the equipment used during the intervention. Importing medical images is mostly a solved problem: the existence of standards such as DICOM means that there are robust open-source implementations of image importers such as DCMTK that we can use, and we can also use ITK image readers for a large array of image formats.

There are still some rough edges though, for example, it is known that some equipments and software generate the orientation matrix of images in an incorrect manner, thus medInria comes with its own heuristics to try to identify those cases and fix them transparently for the user.

Clinicians also need to import other types of data beyond medical images, such as meshes (see Fig. 2) or tensor images. In the specific case of meshes, depending on their origin, they can come in very different formats. In our example of cardiac electrophysiology, they frequently come from Saint-Jude NavX system, BioSense CARTO, or CardioInsight, with various file formats. Currently, in research, the *de facto* standard for meshes is the VTK format.

Sadly, the fact that the VTK file format[6] is the *de facto* standard when it comes to meshes is not a satisfactory solution. Notably, this format has no support for metadata, which means we cannot attach patient

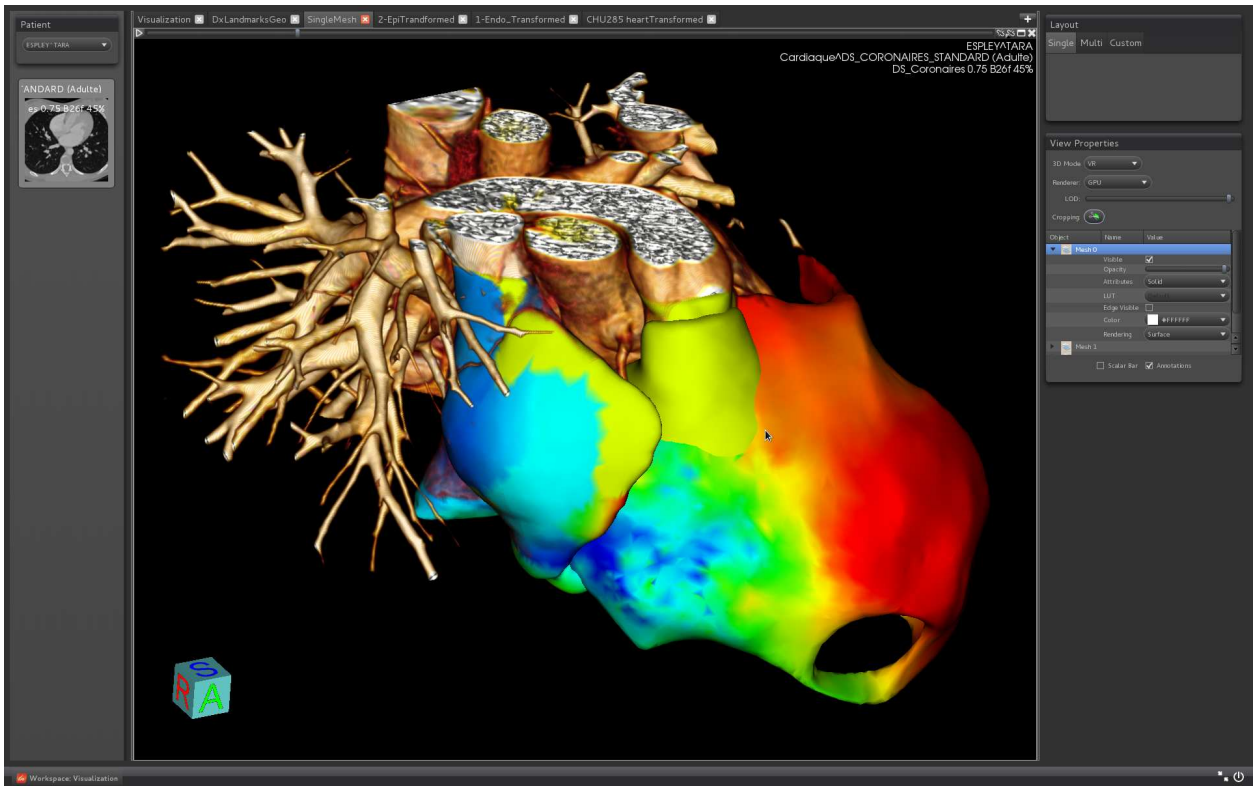

Figure 2: Combining data from different sources: a CT-scan image in volume rendering mode, an Ensite NavX mesh and a BioSense CARTO mesh. Such multimodal integration is not possible in the commercial platforms of each of these companies.

information to it. Metadata are a very important part of a patient's record, as it contains a breadth of information on the patient itself, which may come as useful during the diagnosis. It also does not support sequences, or annotations. This is why we introduced a new format, which we think is a good compromise between backward compatibility and extensibility.

This new format is based on the concept of "Composite dataset" that currently exists in medInria: similar to formats like the Open Document Format[7], we wrap a collection of files in a ZIP archive, and expose this to the user. It makes file sharing easier as everything is exposed as a single file, making it simpler to keep it all in one place.

Inside the ZIP archive, we store a sequence of meshes in a simple and backward compatible manner, that is as a series of VTK files, thus giving the possibility to users who wish to load those file into software that does not implement this format to unZIP the archive and load the file separately. We can then use regular text files stored inside the archive to implement functionalities like metadata or annotations. As the format in these text files is kept simple and human readable, their content can easily be imported, either by hand or through an ad-hoc importer, into other software.

4 Processing Algorithms

When we talk of standards we often think of data formats and protocols, but we should consider also the issue of sharing algorithm implementations amongst different software. Here we take the specific example

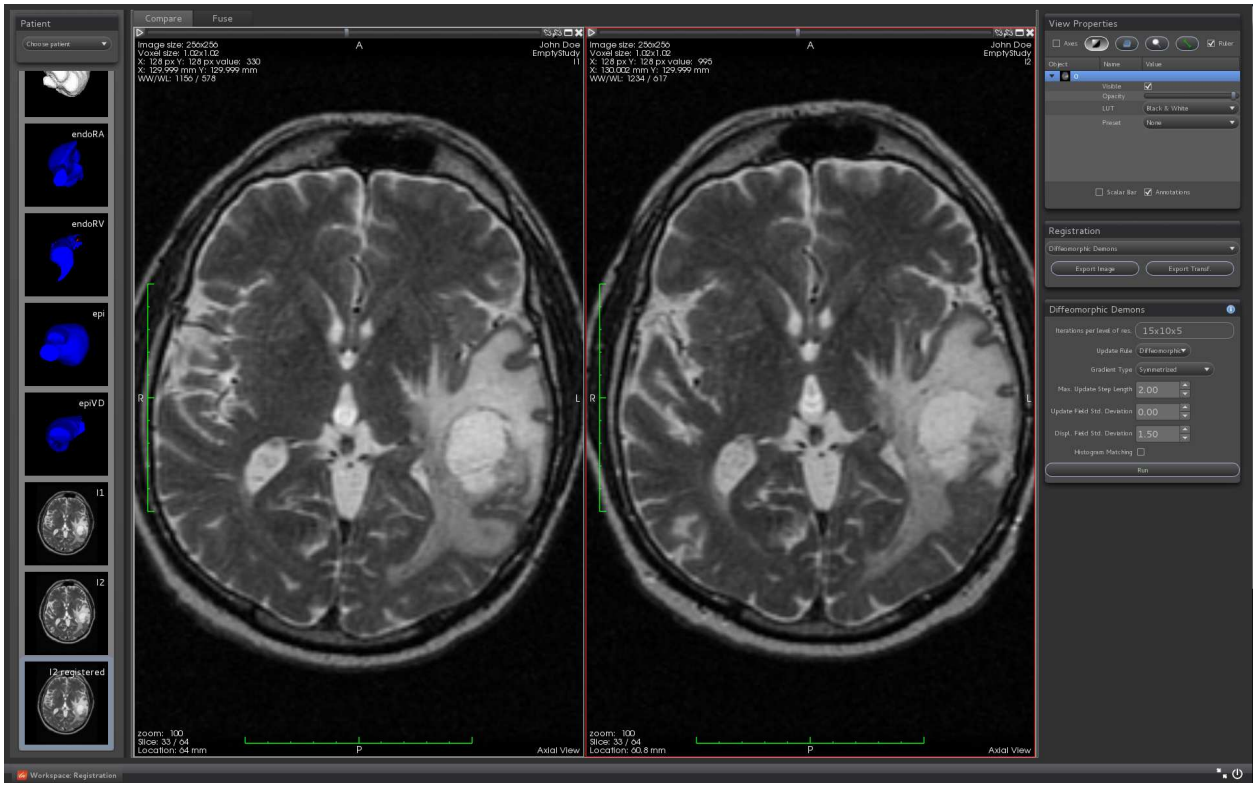

Figure 3: medInria v2.0.1: the registration workspace

of registration algorithms, but those considerations apply to most types of processing.

As clinicians often require a large amount of data to be able to make a diagnosis, different modalities of observation may be needed in order to gather information on the patient. These different modalities may have different origins, either because they were acquired at different times or in different spatial configurations, which justifies the need for algorithms which registers in the same spatial coordinates different datasets.

However, pairing the numerous image modalities and file formats around with the many registration algorithms is far from being straight forward. Indeed, each registration algorithm has its own input/output formats, set of parameters and visualisation. Thus, its use by a non-expert is challenging and the combination or comparison of such algorithms is difficult.

medInria has tried to address this concern by introducing an open source registration API which goal is to propose a simple and intuitive interface shared by all registration methods. This common interface greatly simplifies the user experience while helping the author of a new registration method to focus on the algorithm, not on the user interface. We call this interface RPI for `Registration Programming Interface`.

ITK also provides a registration framework and one may ask why RPI is not included into this framework. The ITK registration framework is too specific and does not allow to easily integrate some registration methods that may not be ITK based, and which do not use its registration loop. As a consequence, it has been decided to not base the registration API on this framework but to design a new framework simpler and more generic. It is still partly based on existing ITK objects though. We made that choice as we felt ITK being a powerful and popular open-source library, it was best not to reinvent already proven components.

RPI defines a C++ programming class which wraps the registration algorithm and provides the generic API.

This class gives a developer a strict but simple programming framework, that exposes only a few methods to override to connect the algorithm to the API. Its purpose is to provide a common interface which has been designed to cover most of the registration methods. It handles many different transformations : linear (both rigid and affine), displacement fields, and stationary velocity fields. The two images to register together are stored as itk::Image objects. This allows a wide range of image types and dimensions : pixel type can be either a scalar type (e.g. short, float, etc.) or can be a itk::Vector for diffusion tensor images. The resulting transformations are also stored as ITK objects.

The registration algorithm can be tuned using parameters that can be exposed by the developer via the API. For the moment, this mechanism is rather simple, so it does not lend itself to automatic GUI generation for example, as parameters have no semantic typing or metadata associated with them yet. This is an improvement currently being considered, allowing us to associate hints such as bounds, units, acceptable values etc to parameters.

The RPI effort is open-source, and its documentation and source are available online[3].

5 Data Visualisation

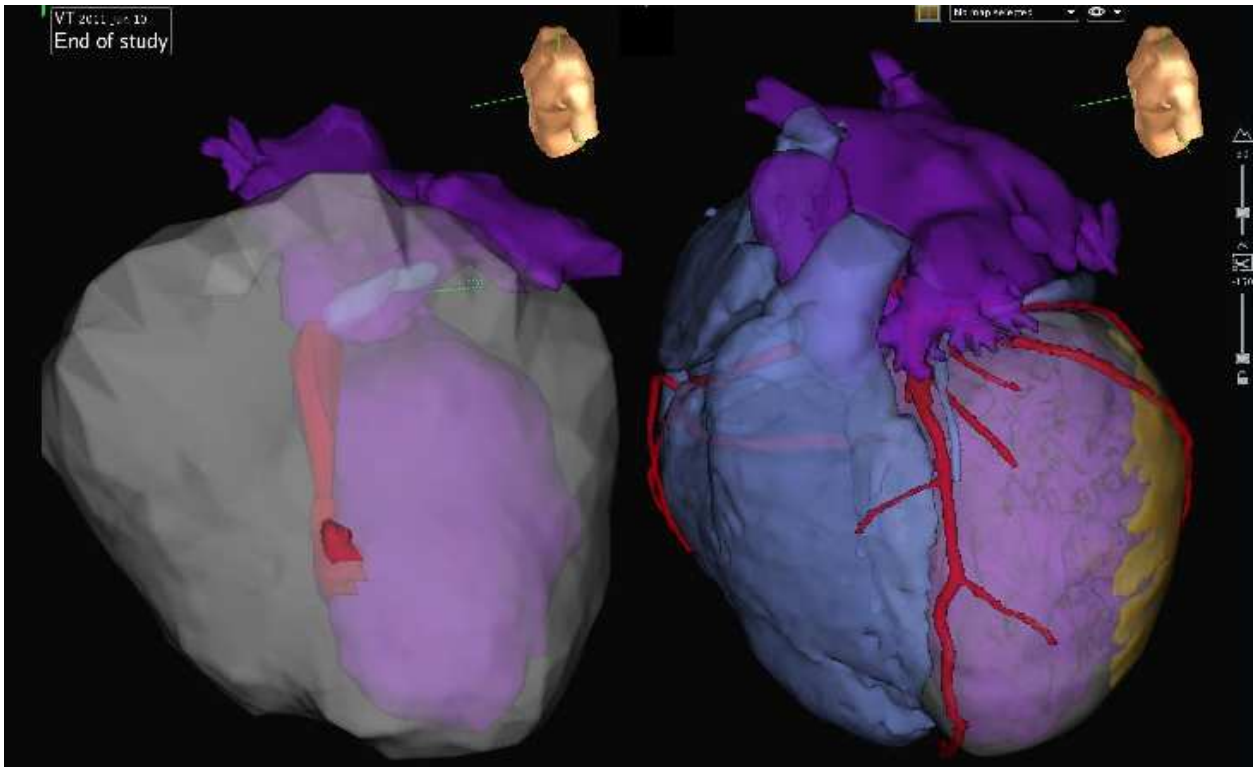

Figure 4: Two heart meshes. Left: a heart mesh from the NavX System, Right: a heart mesh segmented from images, exported to NavX with medInria

In order to visualise data during the interventions, cardiologists often use meshes representing the patient anatomy to guide the catheter navigation. However the tools provided to generate these meshes and control the way they look are often limited. We propose in medInria a tool using more advanced meshing techniques, as well as a way to easily set the colour, transparency,... In practice, clinicians then export the mesh in the

format of a given interventional platform, and load it in the cathlab. We present in Fig. 4 the meshes generated by the commercial platform on the left, and the one generated by medInria, once loaded into the platform on the right.

In the context of cardiac preoperative diagnosis, medInria is also often used by clinicians to study data such as activation maps or potential maps, to identify reentrant circuits that may cause tachycardia, for instance. To do so, they load mesh files with their associated attributes, and choose a color scale to map values into colors. The usual technique for applying a color map to a mesh's attribute is to compute its range, and "stretch" the values to cover the whole color scale. The issue with this approach is that to compare two meshes loaded in the same view, you need to have the same value mapped to the same color, so in medInria we make sure that the range is computed on the values of all the displayed meshes combined, to make sure the color maps applies the same to each. This also applies to sequences, where care has to be taken to compute the range over the whole sequence, and not over individual time steps.

The second issue with color scales, is that there is no actual standard set of color scales, nor a standard format to import / export them. In medInria, we have taken the practical approach of providing the set of color scales found in the most used imaging software, such as the one found in Osirix.

A similar problem arises concerning the display of diffusion tensors directions (Fig. 5) using color scales.

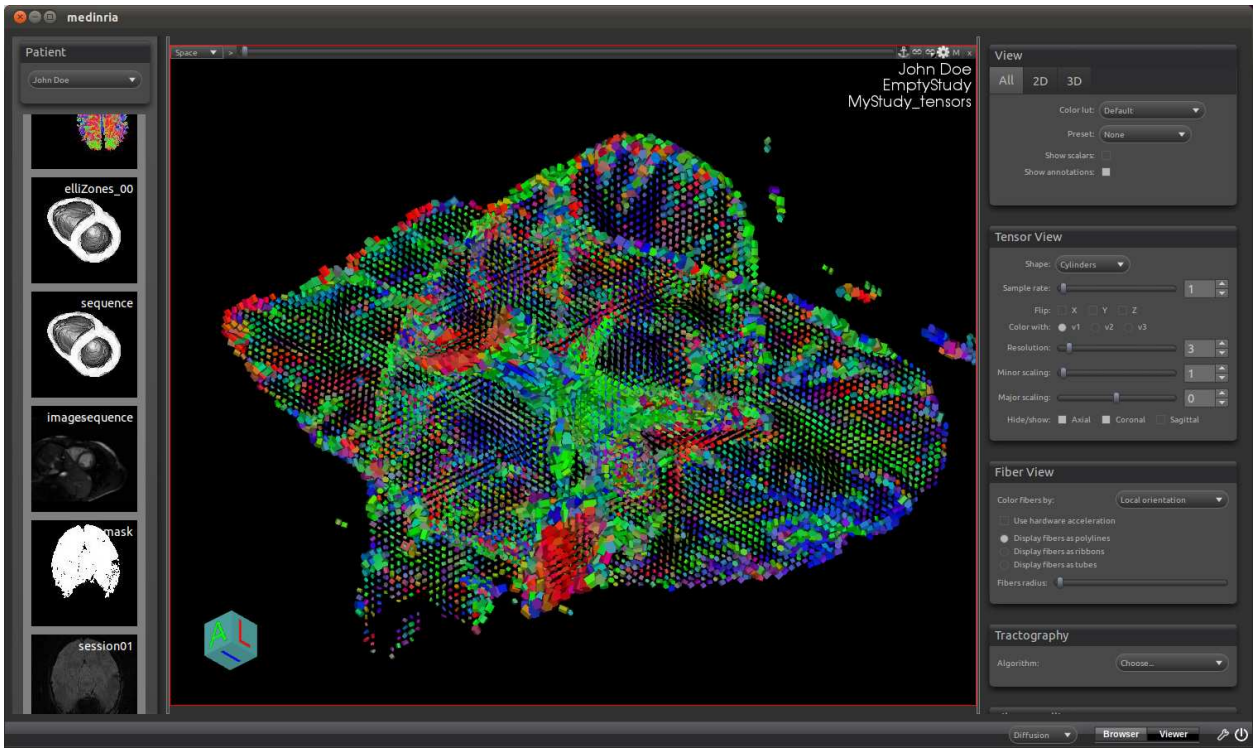

Figure 5: Tensors display with colors representing the direction of the diffusion

There is to our knowledge no standard as to which color should stand for which axis, so it's common that different software use different colors, and thus have very varying visualisation of the same dataset. Again, in this situation we try to take a practical approach, and we provide a way to select a color for each axis.

6 Conclusion

medInria strives to provide clinicians with a reactive, ergonomic and intuitive software, which finds itself at a crossroad between trying to provide access to state-of-the-art while also staying grounded on practical considerations. Thanks to our partnership with the CHU of Bordeaux, we get direct and concrete feedback on what make sense in a live environment, allowing us to refine our proposed solutions, as well as design new tools that fill an actual need of clinicians.

Given the interoperability issues between different equipments and the difficulty of interaction between competing companies, we believe such academic software is needed in order to achieve the integration of multimodal datasets coming from various equipments for diagnosis, therapy planning, and interventional guidance.

medInria is growing fast, and will remain free. Windows, Linux and Mac OS X versions can be downloaded at http://med.inria.fr.

References

[1] CISTIB. Graphical interface for medical image analysis and simulation. http://www.gimias.org/. 2

[2] Slicer Community. 3dslicer. http://www.slicer.org/. 2

[3] INRIA. Registration programming interface. http://www-sop.inria.fr/asclepios/software/RPI/. 1, 4

[4] KitWare. Insight toolkit. http://www.itk.org/. 2

[5] KitWare. Visualization toolkit. http://www.vtk.org/. 2

[6] KitWare. Vtk file formats. http://vtk.org/VTK/img/file-formats.pdf. 3

[7] OASIS. Open document format. https://www.oasis-open.org/committees/office/. 3

[8] OFFIS. Dicom toolkit. http://dicom.offis.de/. 2

[9] N. Toussaint, M. Sermesant, and P. Fillard. vtkinria3d: A vtk extension for spatiotemporal data synchronization, visualization and management. In *Proc. of Workshop on Open Source and Open Data for MICCAI*, Brisbane, Australia, October 2007. 2