A web interface for 3D visualization and interactive segmentation of medical images

Hector Jacinto, Razmig Kéchichian, Michel Desvignes, Rémy Prost, Sébastien Valette

A Web Interface for 3D Visualization and Interactive Segmentation of Medical Images

Hector Jacinto* OneFit Medical, SAS CREATIS
Razmig Kéchichian† CREATIS
Michel Desvignes‡ GIPSA-LAB
Rémy Prost§ CREATIS
Sébastien Valette¶ CREATIS

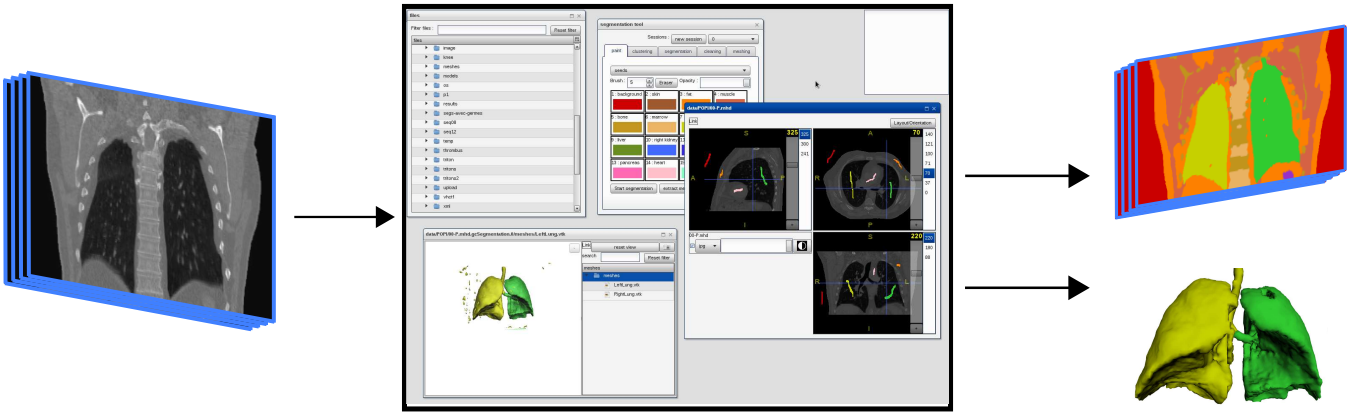


Figure 1: *Our framework enables visualization and processing of large medical images on modest computers. Here, a simple example of lung segmentation is shown.*

Abstract

We propose a web-accessible image visualization and processing framework well-suited for medical applications. Exploiting client-side HTML5 and WebGL technologies, our proposal allows the end-user to efficiently browse and visualize volumic images in an Out-Of-Core (OOC) manner, annotate and apply server-side image processing algorithms and interactively visualize 3D medical models. Server-side implementation is driven by a file-based, simple, robust and flexible Remote Procedure Call (RPC) scheme well suited for heterogeneous applications. We demonstrate the efficiency of our approach with both an interactive medical image segmentation and a 3D rendering of segmented anatomical structures. As a secondary contribution, we improve the segmentation algorithm with the introduction of user-defined anatomical priors.

*e-mail:hector.jacinto@creatis.insa-lyon.fr
†e-mail:razmig.kechichian@creatis.insa-lyon.fr
‡e-mail:michel.desvignes@gipsa-lab.grenoble-inp.fr
§e-mail:remy.prost@creatis.insa-lyon.fr
¶e-mail: sebastien.valette@creatis.insa-lyon.fr,
CREATIS: CNRS UMR5220; Inserm U630; INSA-Lyon; Université de Lyon, France
GIPSA-LAB : Grenoble-INPG, France

CR Categories: I.3.2 [Computer Graphics]: Graphic Systems—Remote systems** I.4.6 [Image Processing and Computer Vision]: Segmentation—Pixel classification; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations J.3 [Computer Applications]: Life and Medical Sciences—Health;

Keywords: HTML5, WebGL, web interface, clustering, segmentation, graph-cuts, meshing

1 Introduction

Technological advances have had a great impact on medical practice. Digital Computed Tomography (CT) Scanners, Magnetic Resonance Imaging (MRI) Devices and Ultrasound Sensors (US) have spread and are nowadays routinely used, which results in an increased volume of images to store, retrieve and visualize. Moreover, research in medical image processing is still ongoing, for which tools are developed for segmentation, detection, classification and visualization of such images.

The Body Browser [Blume et al. 2011] proved the concept of medical visualization to be achievable with HTML5. Such an approach can be extended to patient-specific medical data. Our goal is to allow an interactive segmentation of medical images i.e. extracting anatomical structures such as bones, lungs and other organs. The resulting models can be further used in various applications : orthopedics, simulations and statistic studies.

We focus on interactive techniques as automatic segmentation is still an open problem. Allowing the user to provide information and corrections to the segmentation algorithm improves success rates regarding different types of medical images. In this paper, we propose a framework aiming at bridging the gap between the end-user applications and ongoing research works.

Our framework is exposed via an HTML5-based web interface,

being able to efficiently visualize 3D datasets (3D images and polygonal meshes), and apply various processing applications on these datasets. Several advantages arise from using web-interfaces for medical image processing:

- with the increasing number of available medical data, the use of centralized repositories makes a lot of sense, and accessing the data via Internet comes as a natural extension, such as proposed in [Jomier et al. 2010]
- with an efficient application framework, researchers can instantly deploy their improvements on ongoing research, without the need for testers and end-users to update local programs.
- along with advances in image acquisition, medical datasets have become more and more demanding regarding Random Accessible Memory (RAM) requirements. Web interfaces allow to efficiently process large datasets on remote High Performance Computers (HPC) or Computing Grids [Glatard et al. 2012], while using only a low-end device for user interaction.

Based on these assumptions, we propose a framework streamlined for medical image processing. In section 2 we describe the core features of our proposal. Sections 3, 4 and 5 describe several key framework components, respectively out-of-core volume visualization, volume segmentation, surface meshing and visualization. These components are further used in an interactive segmentation application for which we present some results in section 6.

2 Framework outline

In order to provide maximal scalability to our approach, our design takes into account two key observations:

- On the client side, the performances of Web browsers have been significantly improved, and HTML5 allows powerful Graphical User Interfaces via the Canvas and WebGL Application Programming Interface (API). We aim at exploiting this processing power, to lighten the server side load.
- On the server side, the use of processing power and RAM should be reduced as much as possible, to allow a large number of concurrent user connections. This can be alleviated using an on-disk processing and content delivery strategy, in spirit with the OOC paradigm.

Our framework is based on the direct use of the server file system as a workspace for the end-user. The file system is exposed to the user via Asynchronous Javascript and XML (AJAX) calls. Actions such as file conversion, image segmentation, and mesh processing are performed via RPC, which simply consists in our case in launching executable programs on the server.

2.1 Graphical User Interface

Qooxdoo [1&1 Internet AG 2005-2012] is an Ajax application framework created to implement Rich Internet Applications (RIAs) without the need for additional client-side software installation. It allows programmers to build complex cross-browser applications using its class-based programming model. Qooxdoo was chosen to build the application for our interface mainly as it comes with several build-in graphic elements, clear and numerous application examples and a well constructed API reference. Appearance of graphic elements is set directly in the JavaScript code granting the possibility to incorporate style settings with the implementation of the functionalities of our interface. The graphic elements (such as windows and controllers) apart, Qooxdoo allowed us to create the classes controlling the mechanisms of the main modules such as the file browser, the visualization window and the tools widget.

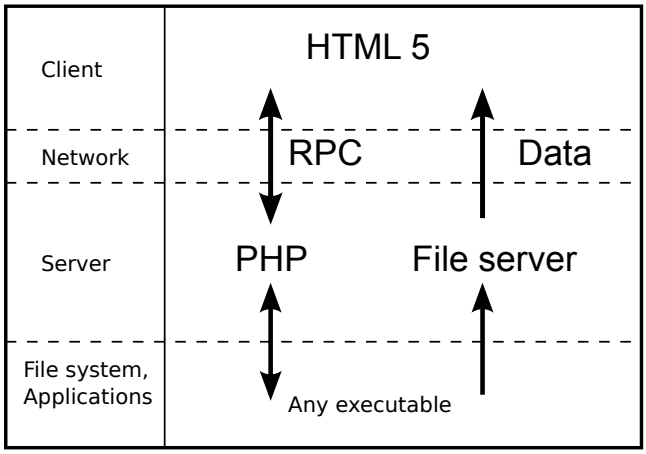


Figure 2: *Framework architecture : RPCs are performed via PHP while most of the data is served via the file server*

2.2 Client-server communication

In order to execute RPC on the server, each possible action is registered in an xml file containing location of the executable and input parameters. Table 1 shows the xml part corresponding to the remeshing algorithm we use.

The JavaScript code for the interface and the built binaries it uses for different actions are all in the server, their interactions are then completely transparent to the user's web browser (regarding the cache and data transmission). In order to keep these interactions very simple, the interface uses JSON-like parameters to accomplish RPCs.

The interface launches server-side applications by executing simple command lines in the server (meaning the interface can use any application which can be launched on the server with a command line execution, provided it is compatible with the file formats used by the interface). For this, a POST request generated in the interface is interpreted by a PHP script which executes the correct command line on the server. The standard output resulting from the execution of the command line is redirected to an action log. The used parameters are saved into a text file in the server.

Using text files to manage RPCs is less powerful than the Python scripts used in ParaViewWeb [Jourdain et al. 2010] but it is much more simple regarding the launching of server-side applications. Parameters are saved for each "action" after execution and specify the corresponding server-side application as well as the user's parameters (see Table 1). They are not only very simply defined but also useful when the user wants to relaunch the action keeping some of the previous parameters. Moreover, this indirect transmission of parameters (passing through the disk of the server) makes the modules constituting the whole processing chain fairly independent. By doing so, the impact of the eventual modification of a module (and even the replacement of the PHP system by another) will be minor to the other elements.

Table 1: *example of definition for a server-side action.*

```
<action name="acvd" executable="/home/visu/src/vtkSurfaceBuild/bin/ACVD">
      <description>generates a simplification of the input mesh</description>
      <parameter name="input_mesh" type="file" required="true"/>
      <parameter name="number_of_desired_vertices" type="int" min="0" required="true" />
      <parameter name="gradation" type="float" required="true" min="0" max="30" default="0"/>
      <parameter prefix="-m " name="force_manifold" type="int" min="0" max="1"/>
      <parameter prefix="-r " name="vertex_reserve" type="int" min="0"/>
      <anchor text="-d 0"/>
</action>
```

2.3 Data formats of medical data

Medical data, the main data input of our interface, is generally stored using the DICOM standard [Medical Imaging & Technology Alliance, NEMA 2000].

However, our interface achieves different actions thanks to server-side build C++ software. And, among these programs, there are the main image processing applications handling actions such as volume slicing, segmentation and volume meshing which were implemented using the open-source library VTK (Visualization Toolkit by Kitware) [W. Shroeder et al. 2003]. Since the VTK library is widely used in image processing for medical images, we choose to use the MetaImage format (using a ".mhd" header to indicate necessary information to read the corresponding volume data) for all the image processing applications of our interface. Implemented applications are indeed more simple when they process a single volume file (a volume presented in the DICOM format consisting of several 2D slices of the volume). Thereby, in order to import medical images but also to export segmented volumes, conversions to and from the MetaImage format are necessary. A DICOM to MetaImage conversion is accomplished with the MRIConvert[Smith 2011] free of charge utility. For image exportation, we use the GDCM [Creatis 2012] library to create volumes slices using the DICOM standard; other image formats are handled via the VTK library.

2.4 Applications

In order to achieve image processing for a given volume data, the web interface offers different graphic widgets : a file browser, action menus, volume viewers, mesh viewers, segmentation tools and meshing tools which are described in the following sections.

3 Out-Of-Core volume visualization

The main function of our interface is the visualization of volume data. This function is carried out by a 2D Multi-Planar Rendering (MPR) display. It consists actually of orthogonal views provided with 2D slices of the given volume which are extracted following 3 different directions : sagittal, coronal and axial which are commonly used in medical imaging. However, since the orientation information for a given volume is not necessarily available nor correct, the interface allows the user to modify the orientation of the displayed slices at will (the proper alignment of the patient with the anatomical axis is assumed). Slices of the input volume can be viewed in an interactive window bearing those 3 orthogonal views (see Figure 3). This viewer is used to display volume data throughout the processing chain e.g. clustering, segmentation, filtering.

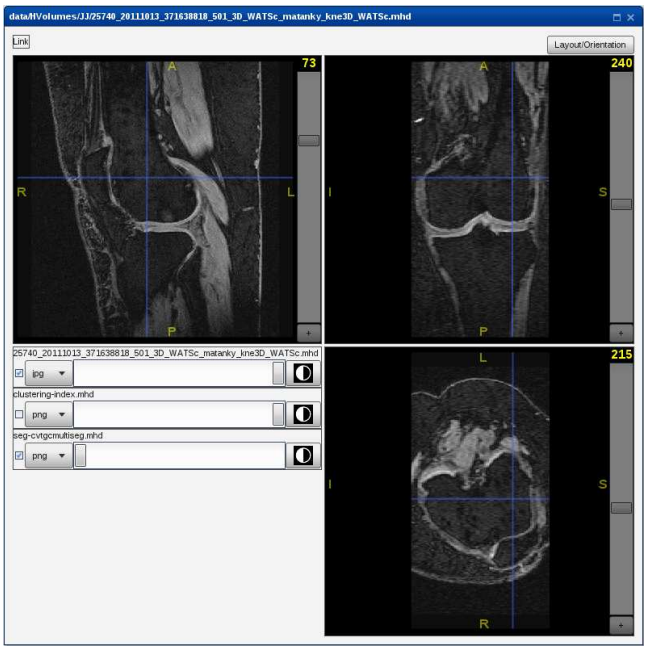


Figure 3: *Volume viewer : MPR visualization of an MRI acquisition of a knee. Upper-left : sagittal view; upper-right : coronal view; bottom-right : axial view; bottom-left : options for the different layered images.*

3.1 Image handling using HTML5

The displays in our interface were implemented using the WebGL library THREE.js [Cabello 2010]. Each oriented 2D view of the volume display is a 3D scene where a flat surface having the current slice as a texture is a placed in space parallel to the screen. In order to obtain that texture, the image data from a JPEG or PNG image file is loaded into a HTML image (meaning only the desired slice of the volume is loaded in an OOC fashion). Using WebGL shaders, the display provides two independent modes easing visualization for either a grey-scale image or a labeled image. On one hand, the intensity of the pixels is modified so the user can set the dynamic range of the gray-scale image to segment. On the other hand, the pixels of a slice of the segmented volume can be colored according a defined look-up table (matching pixel intensities to attributed labels).

When using JPEG images, the original grey levels are rescaled to fit a 8 bit dynamic range, whereas PNG images allow lossless encoding of grey levels by packing the data into red, green, blue and alpha channels. This allow us to encode data up to 32 bits

Table 2: *Volume slicing results : total slice generation time (on a 2.27GHz quad-core test server) and disk usage, for each image format (JPEG and PNG) and for a total slicing according three orthogonal orientations. Disk space refers to the maximum image size and to the total disk space of obtained slices.*

Volume (format)	Dimensions	MetaImage	Lossy compression JPEG		Lossless compression PNG	
		Disk Space	Slicing time	Disk space	Slicing time	Disk space
Original leg (short)	512x512x861	430.5 MB	32.1 s	51.1 kB - 57.5 MB	3.0 min	436.0 kB - 456.9 MB
Leg labeled (uchar)	512x512x861	4.5 MB	31.9 s	54.5 kB - 60.7 MB	29.4 s	14.0 kB - 15.5 MB
Original knee (ushort)	512x512x100	50.0 MB	5.2 s	62.3 kB - 18.7 MB	22.4 s	187.1 kB - 54.9 MB
Knee labeled (uchar)	512x512x100	706.2 k B	4.3 s	33.6 kB - 10.1 MB	3.6 s	9.1 kB - 2.7 MB

float precision. Using WebGL to implement the MPR display is very practical regarding scaling of the 2D image as well as mouse controls. The display is more responsive when it is implemented with WebGL rather than HTML5 canvas displays. Besides, since these JPEG or PNG images are loaded by the user's web browser only when asked, our system corresponds to an OOC volume visualization. Data is indeed stored in the server and indexed as in an on-line computation [Silva et al. 2002]. Our display system contributes then to the responsiveness of the interface and is most convenient as the segmentation process implies that the user has to deal only with a few slices of the volume to segment.

3.2 Volume slicing

As previously said, volume visualization is accomplished by creating 2D slices of the input volume. This task is carried out by one of the server-side applications implemented using the VTK library. This program takes the volume to visualize and creates JPEG or PNG images from the volume data. 2D slices of a given volume are saved to the server-side file system as well as to the user's web browser cache allowing a very fast image loading for previously used volumes which makes our interface cache-friendly. Similar tools such as ParaViewWeb [Jourdain et al. 2010] use a streaming mechanism for image delivery allowing a server-side 3D rendering too. However, in order to maintain simplicity, the MPR display of our interface does not involve a 3D rendering. Our cache-friendly method, visualizing volumes through 2D compressed slices, results in a lighter server load during volume visualization than the server load in the streaming method. Which is why this OOC display is also well adapted to be used by simultaneous multiple-clients.

Table 2 shows some timings and data measurements when processing various volume data with our test server whith 4 CPUS clocked at 2.27 GHz. Rows 1 and 3 correspond to original medical images, while rows 2 and 4 correspond to segmented images (see section 4). Slicing volumes to JPEG files results in a disk occupation of around 10 percent of the original image size, while slicing to PNG images results in an occupation similar to the original image size. As a consequence, once the volume is sliced into images, RAM usage is replaced by similar size disk usage, which is much cheaper.

Measurements on our server showed that the slicing step is a CPU bound operation, so slicing times could be significantly reduced by adding more CPUs to the server.

4 Volume segmentation

Semi-automatic segmentation methods are usually resorted to in order to resolve ambiguities inherent with fully-automatic methods and to alleviate prohibitive time and effort requirements of manual delineation. Interaction is carried out via the attribution of pixel labels, "seeds", inside targeted objects (anatomical structures), one for each type of structure (Figure 4). This provides clues on what the expert intends to segment which can be used to collect appearance statistics of targeted objects and to constrain the solution space of the algorithm. We use a graph-based multi-object semi-automatic segmentation method applied to a coarsened version of the 3D input image to improve runtime performance [Kéchichian et al. 2011]. To ensure the consistency of segmentation with respect to anatomical properties, we extend the aforementioned approach by anatomical vicinity prior information which we dene as a set of pairwise local constraints and integrate it into the multi-label optimization framework.

4.1 Controllable image coarsening by centroidal Voronoi diagram clustering

To coarsen a gray-level image $\mathcal{I}$, we construct a centroidal Voronoi diagram (CVD) on it by minimizing the following function:

$$E_{CVD} = \sum_{i=1}^{n} \left(\sum_{v \in C_i} \rho(v) \|v - c_i\|^2 + \alpha \|I_v - \overline{I_i}\|^2 \right) \quad (1)$$

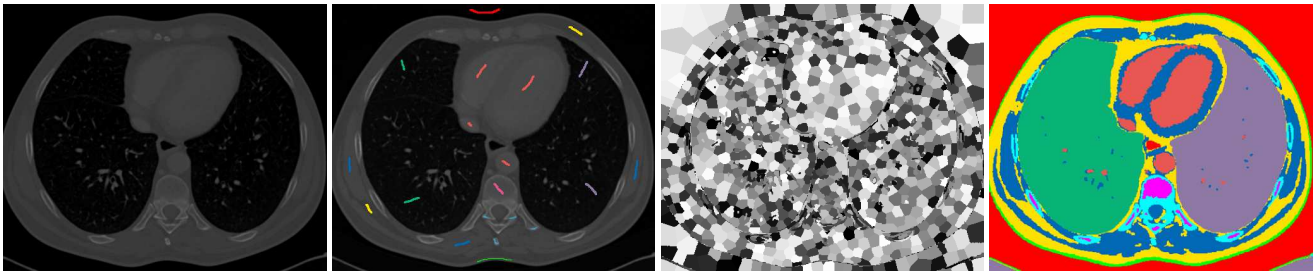
where c_i is the mass centroid of CVD cluster C_i, α a scalar, I_v and $\overline{I_i}$ the intensity levels of voxel $v \in \mathcal{I}$ and cluster C_i respectively, the latter being defined as the mean intensity of its voxels. $\rho(v)$ is a density function which we define to obtain a gradient-adaptive clustering given $\rho(v) = m|\nabla I_v| + b$, where $|\nabla I_v|$ is the magnitude of the image gradient at voxel v. In practice, $m = b = 1$. Density-function weighting encourages the formation of relatively small clusters near intensity edges, allowing fine-grained control of the segmentation boundary (see Figure 4.c).

Intuitively, minimizing function (1) corresponds to maximizing cluster compactness in terms of both color and geometry. In our construction, we use a multi-threaded version of the clustering algorithm in [Valette and Chassery 2004] which approximates a CVD in a computationally-efficient manner, involving only local queries on pairs of clusters. It is substantially faster than clustering algorithms having similar constraints, such as [Veksler et al. 2010], particularly on 3D images.

4.2 Multi-object 3D segmentation via Graph Cut

Given a CVD-clustered image $C(\mathcal{I})$ and a set of labels $\mathcal{L}$, one for each targeted object, the task is to find the optimal mapping $f : C(\mathcal{I}) \mapsto \mathcal{L}; f(C_i) = l_i$ with respect to prior domain information and evidence provided by data. If the former can be expressed as a Markov random field (MRF) for label configurations on clusters and image-derived likelihood densities can be defined for labels, then, according to the Bayes' rule, the optimal mapping is defined as the maximum *a posteriori* probability estimate (MAP) of label configurations [Li 1994].

The MAP estimate can be computed by minimizing an energy



(a) (b) (c) (d)

Figure 4: *(a) CT-cross section of a thorax (b) "seed"-marked image (c) CVD clustering of image (d) segmented image*

function of label configurations having the following form:

$$E(\ell) = \lambda \sum_{C_i \in C(\mathcal{I})} D_i(\ell_i) + \sum_{\{C_i, C_j\} \in N} V_{i,j}(\ell_i, \ell_j) \qquad (2)$$

where N is the irregular neighborhood system reflecting cluster adjacency, and $\lambda \geq 0$ is the underlying Gibbs distribution temperature parameter. In practice, $0 \leq \lambda \leq 0.5$.

4.2.1 Likelihood estimation

In the energy function (2), the unary term $D_i(\cdot)$ is a function derived from observed data to measure the cost of assigning the label ℓ to cluster C_i. Representing each user-identified object in the image by a label, we estimate the conditional probability distribution of object intensity levels $Pr(I|L)$ from user-supplied "seeds" as normalized intensity histograms and define $D_i(\cdot)$ as follows:

$$D_i(\ell_i) = \begin{cases} 0 & \exists s_k \in C_i, k = \ell \\ \infty & \exists s_k \in C_i, k \neq \ell \\ -\log\left(\prod_{v \in C_i} Pr(I_v|\ell_i)\right) & \nexists s_k \in C_i \end{cases} \qquad (3)$$

where s_k represents a "seed" of label type k, and I_v is the intensity level of the voxel v.

In subsequent refinements of an initial segmentation, and in order not to disrupt initial appearance statistics, we allow the user to introduce additional "corrective seeds" to impose hard constraints on the label preference of incorrectly segmented image regions.

4.2.2 Anatomical spatial prior

The binary term $V_{i,j}(\cdot,\cdot)$ in the energy function (2) stems from the second-order clique potentials of the MRF prior energy. This term encodes *a priori* information of interactions between labels assigned to pairs of neighboring sites, and encourages spatial consistency of the labeling. We use a discontinuity-preserving piece-wise constant prior model [Geman and Geman 1984] and extend it to enforce anatomical vicinity constraints.

We would like to penalize assignments of labels to pairs of clusters which violate anatomical consistency. For example, in a segmentation task of thoracic cage structures, the attribution of labels "marrow" and "lung" to pairs of neighboring clusters should be penalized while the attribution of labels "marrow" and "bone" should be encouraged, therefore $\forall i, j;\ V_{i,j}(\text{“}marrow\text{”}, \text{“}lung\text{”}) > V_{i,j}(\text{“}marrow\text{”}, \text{“}bone\text{”})$.

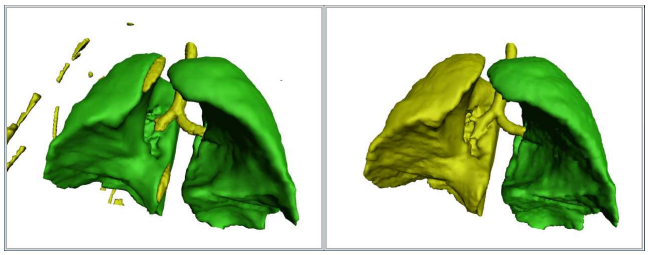


Figure 5: *Segmentation of the lungs in a CT volume. Left/Right : without/with anatomical spatial priors respectively*

Let $\mathcal{L}^{2'} = \{\mathcal{L} \times \mathcal{L}\} \setminus \{(\ell_i, \ell_j) | i \neq j\}$, and let Π be the set of binary predicates defined on $\mathcal{L}^{2'}$ representing possible label couplings, such that $\forall (l_i, l_j) \in \mathcal{L}^{2'}, \pi(l_i, l_j) = \pi(l_j, l_i)$. We define the binary energy term $V_{i,j}(\cdot,\cdot)$ as follows:

$$V_{i,j}(\ell_i, \ell_j) = \begin{cases} 0 & \ell_i = \ell_j \\ \gamma_{soft} & \exists \pi \in \Pi, \pi(\ell_i, \ell_j) \\ \gamma_{hard} & otherwise \end{cases} \qquad (4)$$

where $\gamma_{hard} > \gamma_{soft}$.

In our applications, we weigh $V_{i,j}(\cdot,\cdot)$ by the area of the common boundary of adjacent clusters $|\partial C_i \cap \partial C_j|$ so that clusters sharing longer boundaries tend to prefer similar labels, or label assignments incurring smaller penalties. In some applications, to account for intensity profile dissimilarities between such clusters in label assignments, we also weigh the binary energy term by a contrast function defined as follows:

$$\psi_{i,j} = e^{-\frac{\left(\overline{I_i} - \overline{I_j}\right)^2}{2\sigma^2}} \qquad (5)$$

where $\overline{I_i}$ and $\overline{I_j}$ represent intensity levels of neighboring clusters C_i and C_j respectively and σ can be estimated as noise in data. Figure 5 shows the improvements obtained in the segmentation thanks to anatomical spatial priors.

4.2.3 Multi-label optimization

Due to the large number of local minima, especially in high-dimensional spaces, the minimization of non-convex energy functions such as (2) is NP-hard [Boykov et al. 2001]. In certain cases, namely when $|\mathcal{L}| = 2$, the global minimum of the energy function can be computed exactly in low-order polynomial time [Boykov and Funka-Lea 2006]. In general, however, and in our multi-object segmentation problem where $|\mathcal{L}| \geq 3$, approximation

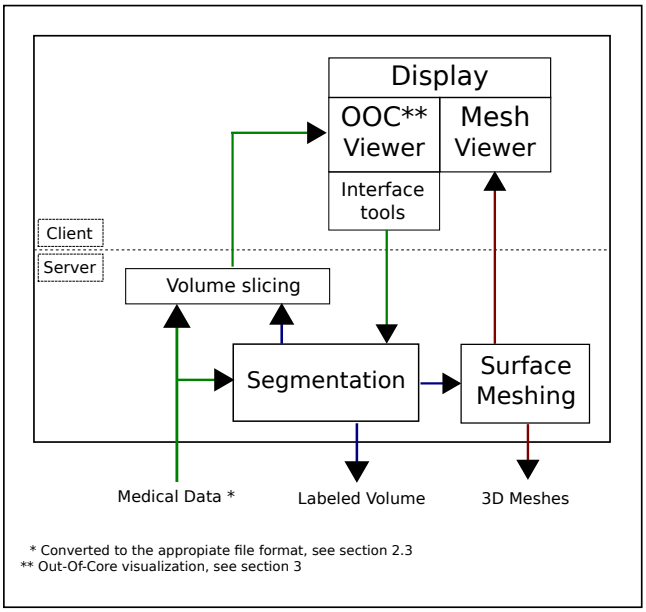


Figure 6: *Block diagram of the segmentation application*

algorithms have to be used. We use the "expansion move" algorithm described in [Boykov et al. 2001], which is capable of finding a solution within a known factor of the global minimum and has been shown to outperform other popular multi-label optimization algorithms on benchmarks in terms of both speed and quality [Szeliski et al. 2008]. The expansion move algorithm can be applied when the binary energy term is submodular [Kolmogorov and Zabih 2004]. To respect this condition (the triangular inequality), we define $\gamma_{hard} = 2\,\gamma_{soft}$ so that all binary energy terms are submodular.

4.3 Segmentation interface

Figure 6 represents the main modules of the segmentation application and their interactions.

In addition to visualization, the OCC volume viewer is used in conjunction with the segmentation widget during the segmentation process. This widget sets up, among other elements, the drawing tools which allow the user to create the colored seeds needed to compute initialization data for the segmentation algorithm. This is a crucial task where the "seeds" provided by the user's input significantly impacts results. Moreover, the interface allows the user to introduce pairwise relations on given labels reflecting anatomical adjacency constraints as mentioned in section 4. Once the segmentation process is complete, the resulting image is automatically overlaid with the original grey-scale image easing the input of corrective "seeds".

Regarding the implementation of the drawing tools, those functionalities were implemented using a HTMLCanvasElement object. From this canvas, the drawn seeds are saved into a PNG image file. The lossless compression and the transparency support of the PNG format is essential for the interpretation of the seeds files by the segmentation program.

5 Surface Meshing and visualization

5.1 Meshing

Once the volume is segmented in different organs, we extract the organs surface for 3D visualization. This can be helpful when one want to visually inspect the quality of the segmentation, and possibly to localize complex regions, which is not easy when one has only access to 2D slices.

For each organ, we first extract its surface using the Marching Cubes algorithm [Lorensen and Cline 1987]. The output mesh is then simplified using the open-source ACVD program [Valette et al. 2008], which was recently extended to generate manifold meshes [Audette et al. 2011]. This simplification has two goals : improve the quality of the rendered surface, as meshes generated by the Marching Cubes approach exhibit blocky artifacts, and to decrease the number of vertices and triangles of the mesh, which decreases the amount of data to be transmitted to the client interface.

5.2 Mesh rendering

Among already available approaches for static resolution mesh rendering [Di Benedetto et al. 2010; Cabello 2010] and progressive approaches [Maglo et al. 2010], we use the THREE.js framework, as it natively handles meshes encoded in the openCTM format [OpenCTM 2009]. When one wants to visualize a mesh which file encoding is different from openCTM, we perform a transparent conversion to the appropriate format. Similarly to what we do for volume data, the conversion result is stored in the cache section of the file browser, to accelerate access from several clients.

6 Results

The images shown in this section were obtained directly from our web interface.

Figure 7 shows 3D meshes extracted from the segmentation on a thoracic CT image [Vandemeulebroucke et al. 2007]. We have segmented bones, the heart and both lungs. The sternum is not connected to the ribs since they are connected by cartilage which shows different gray levels than bones in CT.

One of the main contributors in the project of our framework is a society which proposes patient-adapted products to help the installation of knee replacement implants during surgery. To manufacture such products, segmentation of an MRI of the patient's knee bones followed by 3D modeling is carried out. Currently, they create a 3D model from a semi-manual segmentation of a single bone at a time. This segmentation is done by drawing the contour of the given bone every few slices of the volume and then automatically creating the contours for the untreated slices. The purpose of the contribution is to reduce the over-all time of generation of the 3D models of the patient's knee bones. Our framework allows to reduce the over-all time of generation of the 3D models (reducing mainly segmentation time) from 4 hours (per bone) to 1 hour roughly (for all the bones in an MRI of the knee like the one described in table 2) while obtaining similar results to those of the current solution (see Figure 10). Figure 8 shows the 3D models of the leg bones of a CT volume segmented during the first tests of our framework on knees. Figure 9 shows the 3D meshes of the knee bones of an MRI volume, the models of the femur and the tibia bones being used afterwards for production.

Regarding other proposals, there is the integration ParaViewWeb into MIDAS by [Jomier et al. 2011] which can be extended to implement image processing and Dicom support as in our interface.

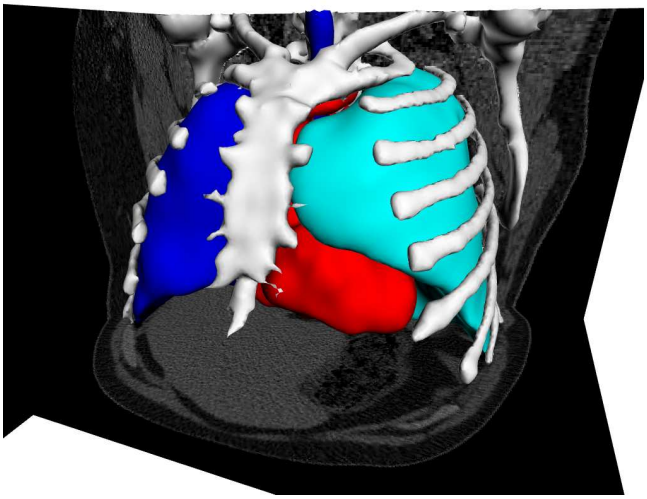


Figure 7: *3D meshes extracted from the segmentation of a thoracic CT image. Here, the lungs, the heart and the bones are extracted.*

These options do not come as built-in though. Another example is [Mahmoudi et al. 2010] which uses VRML (replaced by X3D since Mahmoudi's et al publication) to provide the 3D features. However, we chose to implement 3D in our interface with WebGL since the THREE.js library is very flexible and easy to use within the Qooxdoo JavaScript framework.

7 Conclusion

In this paper, we proposed a web framework for efficient medical image processing, which allows remote visualization and processing on modest client hardware. Stress is put on scalability and ease of implementing different processing tasks such as segmentation, meshing, data conversion. There are many possible improvements to our approach such as a scheduling paradigm to regulate server load, security handling or progressive 3D mesh transmission.

Acknowledgements

The first author is founded by OneFit Medical, SAS. The knee images are courtesy of OneFit Medical, SAS. This work is supported by the Région Rhône-Alpes via the SIMED project of the research cluster ISLE. The authors thank the Virtual Imaging Platform (VIP) project [ANR-09-COSI-03] for hosting our test server.

References

1&1 Internet AG, 2005-2012. Qooxdoo, Universal JavaScript Framework. `http://qooxdoo.org`.

Audette, M., Rivière, D., Ewend, M., Enquobahrie, A., and Valette, S. 2011. Approach-guided controlled resolution brain meshing for fe-based interactive neurosurgery simulation. In *Workshop on Mesh Processing in Medical Image Analysis, in conjunction with MICCAI 2011.*, 176–186.

Blume, A., Chun, W., Kogan, D., Kokkevis, V., Weber, N., Petterson, R. W., and Zeiger, R. 2011. Google body: 3d human anatomy in the browser. In *ACM SIGGRAPH 2011 Talks*, ACM, New York, NY, USA, SIGGRAPH '11, 19:1–19:1.

Boykov, Y., and Funka-Lea, G. 2006. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision 70*, 2, 109–131.

Boykov, Y., Veksler, O., and Zabih, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11, 1222–1239.

Cabello, R., 2010. Three.js. `http://threejs.org/`.

Creatis, 2012. The GDCM library. `http://sourceforge.net/apps/mediawiki/gdcm`.

Di Benedetto, M., Ponchio, F., Ganovelli, F., and Scopigno, R. 2010. Spidergl: a javascript 3d graphics library for next-generation www. In *Proceedings of the 15th International Conference on Web 3D Technology*, ACM, New York, NY, USA, Web3D '10, 165–174.

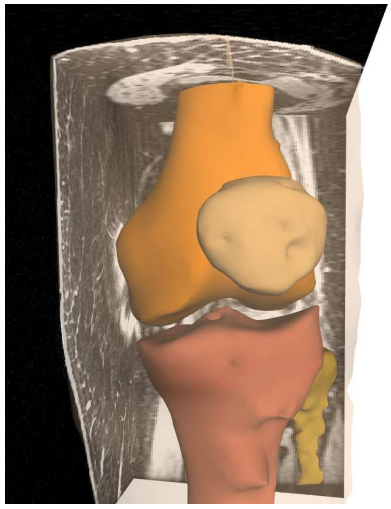


Figure 9: *Visualization of the 3D meshes extracted for a segmented MRI of a knee. Here, the four bones of the knee are shown.*

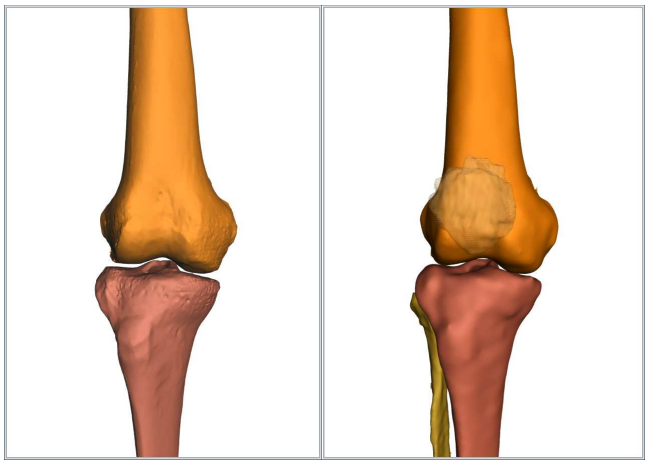


Figure 10: *Left : 3D models generated with the manual segmentation (roughly 4 hours per bone). Right : the same knee is segmented with our interface (roughly 1 hour over-all)*

Figure 8: *Surface meshing of the 3D model obtained for the bones of a segmented CT of an entire leg.*

GEMAN, S., AND GEMAN, D. 1984. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6*, 6, 721–741.

GLATARD, T., MONTAGNAT, J., EMSELLEM, D., AND LINGRAND, D. 2008. A service-oriented architecture enabling dynamic services grouping for optimizing distributed workflows execution. *Future Gener Comp Sy 24*, 7 (jul), 720–730.

GLATARD, T., MARION, A., BENOIT-CATTIN, H., CAMARASU-POP, S., CLARYSSE, P., FERREIRA DA SILVA, R., FORESTIER, G., GIBAUD, B., LARTIZIEN, C., LIEBGOTT, H., MOULIN, K., AND FRIBOULET, D. 2012. Multi-modality image simulation with the virtual imaging platform: Illustration on cardiac mri and echography. In *IEEE International Symposium on Biomedical Imaging (ISBI)*.

JOMIER, J., BAILLY, A., LE GALL, M., AND AVILA, R. 2010. An open-source digital archiving system for medical and scientific research. In *The 5th International Conference on Open Repositories (OR2010)*.

JOMIER, J., JOURDAIN, S., AYACHIT, U., AND MARION, C. 2011. Remote visualization of large datasets with midas and paraviewweb. *Integration The Vlsi Journal*.

JOURDAIN, S., AYACHIT, U., AND GEVECI, B. 2010. Paraviewweb, a web framework for 3d visualization and data processing. *IADIS International Conference on Web Virtual Reality and Three-Dimensional Worlds (07)*.

KÉCHICHIAN, R., VALETTE, S., DESVIGNES, M., AND PROST, R. 2011. Efficient multi-object segmentation of 3d medical images using clustering and graph cuts. In *IEEE Int. Conf. on Image Processing*, 2196–2200.

KITWARE, 2011. The MetaImage file format. `http://www.itk.org/Wiki/MetaIO/Documentation`.

KOLMOGOROV, V., AND ROTHER, C. 2007. Minimizing non-submodular functions with graph cuts - a review. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 7, 1274–1279.

KOLMOGOROV, V., AND ZABIH, R. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 2, 147–159.

LI, S. Z. 1994. Markov random field models in computer vision. In *European Conf. on Comp. Vision*, vol. 801. 361–370.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '87, 163–169.

MAGLO, A., LEE, H., LAVOUÉ, G., MOUTON, C., HUDELOT, C., AND DUPONT, F. 2010. Remote scientific visualization of progressive 3d meshes with x3d. In *Proceedings of the 15th International Conference on Web 3D Technology*, ACM, New York, NY, USA, Web3D '10, 109–116.

MAHMOUDI, S. E., AKHONDI-ASL, A., RAHMANI, R., FAGHIH-ROOHI, S., TAIMOURI, V., SABOURI, A., AND SOLTANIAN-ZADEH, H. 2010. Web-based interactive 2d/3d medical image processing and visualization software. *Comput. Methods Prog. Biomed. 98*, 2 (May), 172–182.

MEDICAL IMAGING & TECHNOLOGY ALLIANCE, NEMA, 2000. The DICOM standard. `http://medical.nema.org`.

OPENCTM, 2009. Openctm format. `http://openctm.sourceforge.net/`.

ROTHER, C., KUMAR, S., KOLMOGOROV, V., AND BLAKE, A. 2005. Digital tapestry. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 589–596.

SILVA, C., CHIANG, Y.-J., EL-SANA, J., AND LINDSTROM, P. 2002. Out-of-core algorithms for scientific visualization and computer graphics. *IEEE Visualization Conference*.

SMITH, J., 2011. MRIConvert. The Lewis Center for Neuroimaging, University of Oregon. `http://lcni.uoregon.edu/~jolinda/MRIConvert/`.

SZELISKI, R., ZABIH, R., SCHARSTEIN, D., VEKSLER, O., KOLMOGOROV, V., AGARWALA, A., TAPPEN, M., AND ROTHER, C. 2008. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 6, 1068–1080.

VALETTE, S., AND CHASSERY, J. 2004. Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. *Computer Graphics Forum 23*, 3, 381–389.

VALETTE, S., CHASSERY, J., AND PROST, R. 2008. Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *IEEE Trans Visu Comp Grap 14*, 2, 369–381.

VANDEMEULEBROUCKE, J., SARRUT, D., AND CLARYSSE, P. 2007. Point-validated pixel-based breathing thorax model. In *International Conference on the Use of Computers in Radiation Therapy (ICCR)*, 195–199.

VEKSLER, O., BOYKOV, Y., AND MEHRANI, P. 2010. Superpixels and supervoxels in an energy optimization framework. In *11th European Conf. on Computer Vision*, 211–224.

W. SCHROEDER ET AL. 2003. *The Visualization ToolKit (VTK), 3rd Edition*. KitWare, Inc. `http://www.vtk.org`.