Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/bbe

Original Research Article

MESA: Complete approach for design and evaluation of segmentation methods using real and simulated tomographic images

Daniel Reska *, *Krzysztof Jurczuk*, *Cezary Boldak*, *Marek Kretowski*

Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

ARTICLE INFO

Article history:
Received 18 October 2013
Received in revised form
21 January 2014
Accepted 4 February 2014
Available online 19 February 2014

Keywords:
Image segmentation
Magnetic resonance imaging
Deformable models
Segmentation evaluation

ABSTRACT

In this paper we present MESA: a platform for design and evaluation of medical image segmentation methods. The platform offers a complete approach for the method creation and validation using simulated and real tomographic images. The system consists of several modules that provide a comprehensive workflow for generation of test data, segmentation method development as well as experiment planning and execution. The test data can be created as a virtual scene that provides an ideal reference segmentation and is also used to simulate the input images by a virtual magnetic resonance imaging (MRI) scanner. Both ideal reference segmentation and simulated images could be utilized during the evaluation of the segmentation methods. The platform offers various experimental capabilities to measure and compare the performance of the methods on various data sets, parameters and initializations. The segmentation framework, currently based on deformable models, uses a template solution for dynamical composition and creation of two- and three-dimensional methods. The platform is based on a client–server architecture, with computational and data storage modules deployed on the server and with browser-based client applications. We demonstrate the platform capabilities during the design of segmentation methods with the use of simulated and actual tomographic images.

1. Introduction

Nowadays, the modern health care cannot be imagined without powerful and precise imaging techniques, like magnetic resonance imaging (MRI), computed tomography (CT) or ultrasonography (USG). Physicians successfully use these methods to improve the diagnosis and treatment process. Development of effective and reliable segmentation methods is one of the most important tasks in the medical image analysis, necessary in the preoperative planning, diagnostics and visualization [1]. The manual segmentation process can often be labor-intensive, especially in the case of large MRI or CT data sets. Usage of reliable segmentation tools

* *Corresponding author at*: Faculty of Computer Science, Bialystok University of Technology ul. Wiejska 45A, 15-351 Bialystok, Poland.
E-mail addresses: d.reska@pb.edu.pl (D. Reska), k.jurczuk@pb.edu.pl (K. Jurczuk), c.boldak@pb.edu.pl (C. Boldak), m.kretowski@pb.edu.pl (M. Kretowski).

http://dx.doi.org/10.1016/j.bbe.2014.02.003

can shorten the total processing time and increase precision and reproducibility of the results.

One of the initial difficulties in the creation and development of medical image segmentation methods is a need of a robust and efficient environment, responsible for the image data management, processing and visualization, followed by the segmentation results analysis. In practice, there are different approaches to this problem. A researcher can create its own application using a general-purpose programming language. The development process can be also conducted in a general computing environment (e.g. MATLAB [2], Mathematica [3]), or in a more specialized, image-oriented package like ImageJ [4], 3D Slicer [5]. Different tools offer various levels of extendability that could require a lot of adaptation to a chosen image segmentation method. Either way, the researcher must take care of the data persistence and analysis of the results. Different segmentation methods (two- (2D) or three-dimensional (3D), region or edge based, etc.) can also produce outcomes of different nature, which can lead to difficulties in comparison of their results. Another problem is the availability of some annotated data, serving as a reference segmentation, which is crucial in the method evaluation. Although there are some available resources of example segmentation [6], in many cases the results have to be compared to a manually obtained segmentation that can vary in its quality, even when obtained with a help of an expert [7]. Finally, the applied methods often require a careful parameter adjusting phase. Finding the correct parameter set for a specific image class can be complicated and time-consuming. However, it is an unavoidable process, since the wrong parametrization can make even a very robust method useless.

In this paper we present MESA (MEdical Snake Arena): a web platform for design and evaluation of medical image segmentation methods. The goal of this platform is to address the above-mentioned problems by providing a fully functional environment for research and education. Currently, the platform focuses on the deformable models, developed within a framework with a method decomposition scheme that unifies the appearance and the way of working of different algorithms. This unification provides a way to develop, visualize, validate and compare methods of different types. Apart from the segmentation functionality, the platform also offers modules for simulation of the MR images from a previously created virtual scene (serving also as the reference segmentation). Moreover, the system provides a module for experimental evaluation, parameter adjustment and comparison of segmentation methods, where the simulated data can be used to validate the results. The system is built in a client–server architecture, offering rich clients in form of Java applets, while the computations are moved to the server machines. The user data (images, methods and results) is stored in a centralized personal profile system, accessible remotely from the client applications.

This paper is organized as follows. Section 2 outlines the background of MRI and deformable models. Section 3 describes the MESA system architecture and functionality of its modules. Usage examples of the system are demonstrated in Section 4. Conclusions and plans for future research are presented in the last section.

2. Background

2.1. Magnetic resonance imaging

MRI is one of the most important tomography methods in medicine [8]. It is partially due to its numerous advantages, like high spatial resolution, good contrast between various tissues or no side effects related to radiation exposure. Moreover, its clinical applications are still expanding rapidly as hardware and imaging technology overcome successive limitations. Some of them include functional MRI (fMRI), MR angiography (MRA), diffusion MRI (dMRI) or MR spectroscopy (MRS).

MRI relies on the intrinsic magnetic properties of body tissues in an external magnetic field [9]. Hydrogen protons are considered the most often because of their high natural abundance in the body tissues (in water and fat). When such particles are placed in a strong static magnetic field and additionally they are irradiated by an oscillating radio frequency (RF) pulse, they can absorb this additional energy (process called excitation). After the RF pulse is turned off, the particles return (relax) to the equilibrium. All external magnetic field changes cause that the magnetic moments of the particles also evolve (they rotate and change their magnitude). Thus, a current in the receiver coils placed around the body can be induced.

The receiver coils are used during the relaxation, since different tissues return to equilibrium at different speeds. Two relaxation processes can be observed, transverse and longitudinal to the main magnetic field. The number of the particles participating in MR, so-called proton density, also influences the received signal.

The acquired signal is collected in a k-space matrix. The signal from a single excitation is the sum of all subsignals coming from all excited particles. Thus, the magnetic gradients are used to encode spatial directions. Finally, the fast Fourier transform (FFT) [11] transforms the k-space matrix to the desired image.

2.2. Deformable models

Deformable models are a class of segmentation methods based on a deforming shape that tries to adapt to a specific image region, extracting it from the background. A 2D active contour (also called ''snake'') [12] is one of the first and the most influential implementation of this principle. The original snake model defines a deformable parametric curve which deforms under influence of internal and external forces. The goal of this evolution process is to minimize the snake energy. With the contour defined as $v(s) = (x(s), y(s))$ where $s \in [0, 1]$, the total snake energy could be written as:

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 E_{\text{int}}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds, \quad (1)$$

where E_{int} is the internal energy (controlling bending and stretching), E_{image} is the image force (moving the snake toward desired features) and E_{con} represents other possible constraints.

Several extensions of the original model were introduced, like addition of an extra inflation (balloon) force [13], gradient

vector field [14], or dual snake representation [15]. The dynamic topology reformulation methods [16] or alternative shape representations, like level sets [17] and electric snake [18] were also proposed. Furthermore, the 2D approach was extended into the third dimension by creation of 3D deformable surfaces [19,20] and in statistical shape models [21].

Deformable models are particularly useful in the segmentation of medical images [22,23]. Their continuous nature allows them to deal with irregularities of the segmented objects, noise and artifacts, typical for the majority of medical imaging techniques. Their interactive and semi-automatic nature can also speed up the labor-intensive segmentation process and increase the reproducibility of the results [24].

3. The MESA system

The main goal of our system is to provide a complete development framework for the researchers, allowing them to easily construct new deformable model-based segmentation methods and then to experimentally evaluate them with different data and parameter sets. The system modules offer a complete development pipeline, from the generation of test images, to the development and experimental evaluation of the segmentation methods.

3.1. *Architecture*

The system is built in the client–server architecture, providing remote computational services (performed on the back-end machines) and web browser-based clients. The created system consists of four modules, available through two independent applications that serve as Internet clients for the server-side computational logic. The back-end services also provide a profile mechanism for storing the user data: image data sets, segmentation results and created methods.

The first client application (see Fig. 1) offers two modules:

- **Problemator** – for designing a virtual scene that will be used as an MRI ''phantom'' for generation of the test images and as the perfect ''ground truth'' reference segmentation;
- **Virtual MRI Scanner** – for simulating the acquisition of MR images from the above-mentioned phantom and creating images for the segmentation evaluation.

The images generated with these tools are used as an input for the next two modules, included in the second application (see Fig. 2):

- **Segmentator** – offering tools for segmentation method development, with visualization and data management functionalities;
- **Ring** – performing the final step in the design process – evaluation of the methods by execution of experiments using various data and parameter sets, while the segmentation results are evaluated against the provided reference.

The client–server architecture separates the computational logic from the visualization and user interaction layer in the client applications. The goal of this approach was to provide a centralized and remotely available platform that will move the computational workload and data storage overhead from the client and provide a collaborative multi-user environment.

3.1.1. *Profiles and central repository*

The system provides a mechanism of personal accounts and profiles, integrated in the client applications. An unlogged

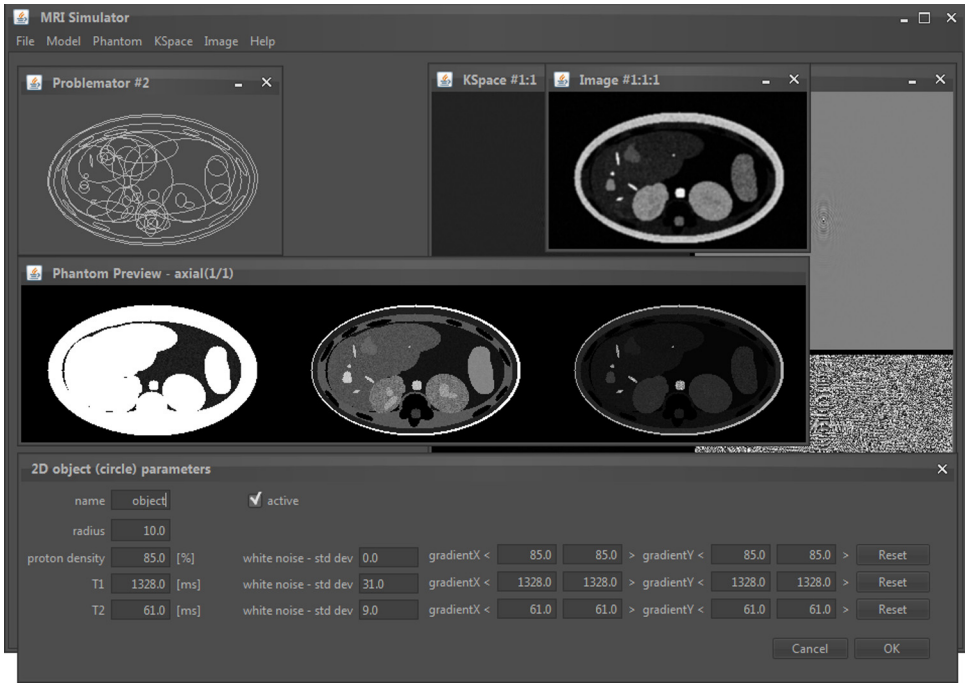


Fig. 1 – User interface of the Problemator and Virtual MRI Scanner client application.

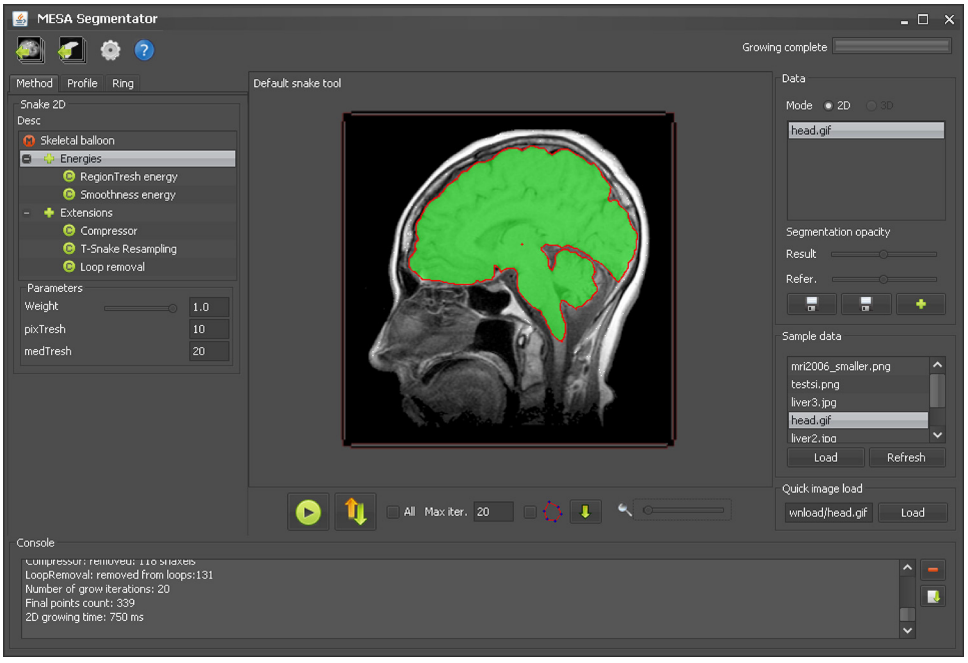


Fig. 2 – User interface of the Segmentator application in 2D mode.

(guest) user can use some basic functionalities and default data. An active account is required to develop new methods, create and upload private images and store information (e.g. image description) in the personal profile. The obtained results can be also stored. All these data are accessible within one account from every module, with the use of a central repository. Due to this, output of one module can be easily used as input to another. A new element can be also marked as a public one, making it accessible for all users.

3.1.2. *Workflow*

The research can be organized in the following workflow (see Fig. 3), where the data, located in the central repository, are passed between the modules. The experiment can be conducted on actual medial images with a provided segmentation reference or on the data generated with the Problemator and Virtual MRI Scanner modules. In the first case, the images and the reference are stored in the repository and they are used by Segmentator during the method development and

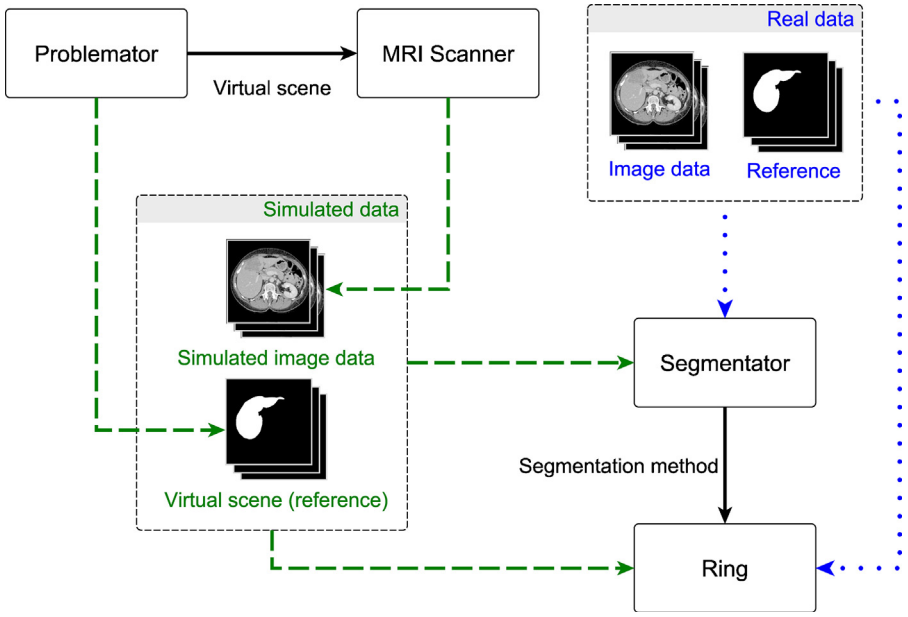


Fig. 3 – Workflow with the data flow: the research can be conducted with the real-life data (dotted links) or with the simulated MR images and reference (dashed links).

then by the Ring module in the method evaluation process. In the second case, the images are firstly simulated. The user constructs a virtual 2D/3D scene with the Problemator. For the objects in the scene, MR tissue parameters can be assigned, thereby creating a virtual MRI phantom. This virtual scene (''ground-truth'' images) is the input for the Ring module, while the virtual phantom is used by the Virtual MRI Scanner module. The Virtual Scanner utilizes the phantom to simulate MR images which are then passed to the Segmentator to develop the segmentation method. Finally, the Ring combines all these elements (virtual scene serving as the reference from the Problemator, simulated images from the Virtual MRI Scanner and the newly constructed method from the Segmentator) in an experiment evaluating the method performance.

The scenario with the simulated images provides a very convenient way to measure the segmentation quality, since the explicit formulation of the virtual scene gives an unambiguous reference for the quality measures. On the other hand, this approach can be, for example, used to test the influence of MR acquisition parameters on the segmentation method performance.

3.2. *Problemator*

The Problemator module is used to create virtual 2D/3D scenes and MRI phantoms. The user communicates with the module by means of the graphical interface (see Fig. 1). The scene consists of a set of parametrized objects in the shape of geometrical figures. The user can choose from among predefined shapes and set the position of the chosen object in 2D/3D space. The objects can be also moved, rotated and scaled. For each object, unique MR contrast parameters (like *T1*, *T2* and proton density) can be assigned. As a result, the created scene can be used as a virtual MRI phantom that undergoes the imaging process. The resulting images are later used to test the segmentation methods in the Segmentator and Ring modules. Moreover, the explicit formulation of the phantom allows the module to generate an ideal reference segmentation of the objects present in the MR images. This ideal segmentation is a sequence of binary images marking exactly the object shape in the phantom. Its great advantage is the lack of errors common for the manual segmentation. Simulated MR images and reference segmentation can be uploaded to the central repository, making them available for other modules. The virtual phantom can be constructed in two modes:

- **2D** – where a scene contains a set of predefined 2D objects, in the form of, e.g., rectangles, circles or ellipses;
- **3D** – where a scene consists of a set of predefined 3D objects, in the form of, e.g., cuboids, spheres or ellipsoids.

The MR contrast parameters can vary inside a single object by the use of linear gradients applied in different directions. In addition, a Gaussian noise can be added to imitate natural variability of the MR tissue properties. The user can introduce all these changes individually for each object in an option window (see the bottom of Fig. 1). The objects can be arbitrarily added/deleted or made invisible in the Virtual MRI Scanner module. The user can pick an object (or objects) from the scene and generate the ideal reference segmentation. The reference images are stored remotely in the repository. The created virtual MRI phantom can be locally exported to an XML file. Thus, it can be also modified without the application by other external software.

3.3. *Virtual MRI Scanner*

This module takes as the input the virtual phantoms, created by the Problemator module, and performs MRI simulations. Simulated MR images along with the generated reference segmentation can be used as evaluation data for the Ring module.

The module can simulate three imaging sequences: spin echo (SE), gradient echo (GE) and inversion recovery (IR). For the imaging sequences, contrast parameters, like repetition time (TR), echo time (TE), inversion time (TI) or flip angle (FA), can be set. Signal acquisition parameters can be also tuned, e.g. acquisition time, sampling interval or number of frequency as well as phase encoding steps and encoding gradient strengths.

The mathematical model of the virtual MRI scanner consists of the three components. The first one is used to simulate the excitation and relaxation processes. The second component is responsible for the signal acquisition, while the third reconstructs images.

3.3.1. *MR excitation and relaxation*

This part of the virtual scanner is based on the Bloch equation [10]. It is used to model the behavior of the magnetization $\mathbf{M} = M_x\hat{\mathbf{i}} + M_y\hat{\mathbf{j}} + M_z\hat{\mathbf{k}}$ produced by the magnetized particles during excitation and relaxation:

$$\frac{d\mathbf{M}}{dt} = \gamma(\mathbf{M} \times \mathbf{B}) - \frac{M_x\hat{\mathbf{i}} + M_y\hat{\mathbf{j}}}{T2} - \frac{M_z - M_0}{T1}\hat{\mathbf{k}}, \tag{2}$$

where $\mathbf{B}$ represents the applied magnetic field, M_0 is the equilibrium magnetization determined by the proton density, *T1* and *T2* are the longitudinal and transverse relaxation times, γ is the gyromagnetic ratio (various for different particles) and $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ are the unit vectors in the x, y and z directions, respectively. The discrete solution of the Bloch equation [25] was used. Moreover, we applied a few simplifications to obtain a less time-consuming solution [26], e.g. the excitation is modeled as an instantaneous rotation of magnetization through a given FA. The external magnetic field $\mathbf{B}$ takes the following form:

$$\mathbf{B} = [B_0 + \Delta B + \mathbf{r} \cdot \mathbf{G}]\hat{\mathbf{k}}, \tag{3}$$

where B_0 is the main strong, static magnetic field, ΔB represents the local magnetic field inhomogeneities and $\mathbf{G}$ is the applied encoding gradients at position $\mathbf{r}$.

3.3.2. *Signal acquisition*

Signal detected during the MRI is a voltage induced in the receiving coils. Based on the Faraday's law of electromagnetic induction [8], the equation to simulate the induced signal in time *t* reads:

$$E(t) = -\int C\frac{\delta\mathbf{M}(\mathbf{r}, t)}{\delta t}dV, \tag{4}$$

where C describes the magnetic properties of the receiver. For simplicity, we consider that the receiver is a long cylindrical coil of length L with N turns encompassing a sample of volume V.

3.3.3. *Image reconstruction*

Acquired signals are collected in the k-space matrix. When the matrix is filled, it contains all information needed to create the image, however, it is spatially encoded. FFT is used to decode the signal, in other words, to arrange for each image pixel the signal strength according to its origin placed in the body.

3.4. *Segmentator*

The Segmentator module offers tools for the development of the deformable model-based methods. It provides a template-based segmentation framework, 2D and 3D visualization capabilities and integration with the repository system for the storage of image data, segmentation results and created methods. This module offers a set of predefined method components, which can be extended with new user-defined elements.

3.4.1. *Segmentation method decomposition scheme*

The segmentation framework uses a *method decomposition scheme* that offers extensible templates and components for a dynamic creation and configuration of the deformable model-based methods. The system proposes a set of built-in and ready-to-use components and templates implementing some basic functionalities (basic contour models, simple energies), as well as more advanced features (active surface template, advanced topology refinement mechanisms). The default set can be then extended by the user (with a scripting language) to construct completely new components.

The decomposition scheme was inspired by the original active contour formulation (Eq. (1)). Within the proposed scheme, each segmentation **method** defines the topological structure of the deformable model (e.g. 2D continuous or discrete curve, 3D parametric surface). Each method has an assigned deformation **model** that controls its evolution process, a set of **energies** applied to the model and driving its evolution, and a set of **extensions**, manipulating the contour directly. The model defines and performs the deformation procedure using the applied energies (representing, for instance, E_{int} and E_{image} in Eq. (1)). The extensions can be used to modify the contour structure independently of the model (e.g. by applying additional forces or topology modifications).

The main goal of this approach was the interchangeability of the components: every model should be compatible with all of the energies and extensions defined for the segmentation method. This allows a simple creation of new methods with the graphical interface (see Fig. 4), where the user can use the predefined components as configurable building blocks.

3.4.2. *New energies and extensions*

The default set of the method components can be dynamically extended with user-defined energies end extensions. The framework uses Beanshell [27], a Java scripting library, to define the method script that is executed on the server. The user can inject its own code into the script, providing a rapid development capabilities at run-time. The user interface

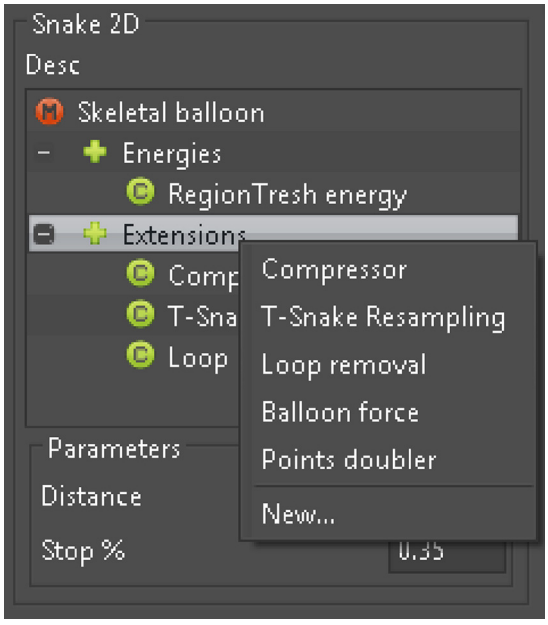


Fig. 4 – User interface of the segmentation method: a tree view of the current model (as its root) and its energies and extensions (as branches).

provides features to create and configure the parameters of the user-defined components and a built-in source code editor (see Fig. 5) for the component definition.

The component code has a set of variables, e.g., the model and the segmented image, injected at run-time to provide the information essential for the functioning of the component.

For the moment, the scripting capabilities only allow the creation of new energies and extension, while new models have to be developed externally. The segmentation framework offers a plugin system for incorporation of externally compiled components, which will be included in a stand-alone version of the Segmentator.

3.4.3. *Segmentation process*

Currently, the system has two built-in modes: discrete 2D active contour (snake) and 3D deformable surface. The snake is structured as a set of points (called "snaxels") interconnected with edges and forming a closed curve. The deformable surface is a face-vertex polygon mesh, consisting of vertices interconnected with edges that form triangular faces. These discrete representations were chosen because of their simplicity of computation and visualization, especially in 3D. Apart from the models, the user is given a set standard energies (intensity- and gradient-based or controlling the elasticity and rigidity) and extensions (providing the topology refinement).

During the segmentation process the user can (in the given order):

- load the input images from the central repository;
- load the "ground-truth" images to be used as the segmentation reference (superimposed on the image with some transparency);

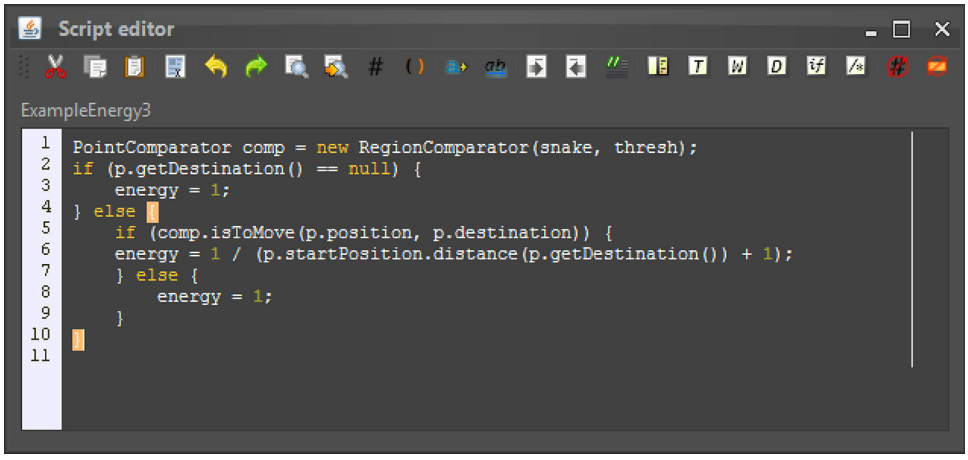


Fig. 5 – The code editor with a custom energy: the editor provides standard text editing tools, as well as syntax highlighting and code completion.

- construct the segmentation method by combining the default and created components;
- initialize the deformable model;
- run the segmentation process with controlled progress (iteratively);
- examine the results (superimposed on the analyzed image), adjust the method and relaunch the segmentation process.

The segmentation can be performed on a single image (2D mode) or on a sequence of images (2D and 3D mode). For the 3D mode, an image sequence can be processed in two approaches:

- the sequence is treated as a stack of 2D slices with a single 2D active contour running independently on each of them;
- the sequence is treated as a volumetric voxel set and can undergo the segmentation with a full 3D method – the active surface.

In the 2D mode, a single image from the stack is visible at a time. In the 3D mode, the entire volumetric data set is visualized using planar reconstruction [28] and direct volume rendering [29]. In both methods, the deformable model separates the segmented object from the background. To uniformly visualize and compare the results, the different spatial representations of the methods have to be converted to a set of region bitmaps, corresponding with the source images. In the case of the 2D contour, this operation is relatively simple, since it requires only to paint the contour region on the segmented image. As for the active surface, the polygon mesh has to be firstly transformed to a volumetric representation with a voxelization algorithm [30] and then projected on the input images.

3.5. *Ring*

The Ring module provides the essential and final stage in the workflow – the evaluation and comparison of the segmentation methods on various data sets, parameters and initializations. The module shares the user interface with the Segmentator module and has access to the created methods and the image repository data. The evaluation is performed by executing the automated segmentation experiments, where the results are compared with an independently loaded "ground-truth" reference segmentation (see Fig. 6(a)). The reference is a sequence of binary images that mark the segmented region. There is no specific constraint for the data and reference origin, since the module can use a set of any valid images. In a complete platform pipeline, the reference is obtained from the Problemator module, while the image data is generated with the Virtual MRI Scanner.

To conduct the experiment, the module requires:

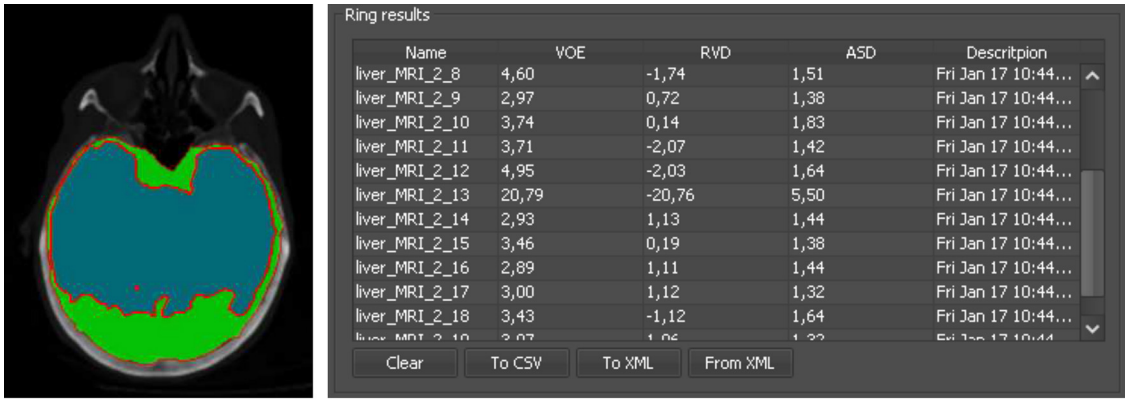
- at least one segmentation method;
- image data to perform the segmentation on;
- reference segmentation to evaluate the results;
- initial position of the deformable model.

After the initialization, the user can select one of the experiment mode (see below) and wait for the quality evaluation results, which are presented in a table view (see Fig. 6(b)). The results can be exported to a CSV format, suitable for a further analysis (e.g. in a spreadsheet), or in a serialized XML form that can be reloaded in future.

3.5.1. *Experiment modes*

The experiments can be performed in four modes: an execution and evaluation of a single selected method, a comparison of many methods working on a single data set, a parametric study of a single method and evaluation mode with the usage of many data sets:

1. In the first mode, the selected method is simply executed on the loaded image data and evaluated with the reference. The user can manually adjust the method parameters, just like during the development stage.
2. The second mode allows the user to run a set of methods on the data, where each execution is performed with a consecutive method from the set. This mode provides an



(a) (b)

Fig. 6 – Snapshots of the Ring module interface: (a) a reference segmentation (marked in green) with a superimposed result (visible in blue) and (b) a table view of the results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

easy way to compare different methods, regardless of their type (2D or 3D).

3. The third mode offers an evaluation of a single method on a given set of runtime settings – a parametric study to find the optimal combination of the parameter values. The user can choose specific method elements (current model, energies or extensions) and set a value range (start, stop and step) for each of the element parameters. Next, a Cartesian product of all the possible parameter values is calculated and set of method execution scripts is created. All these scripts are then executed sequentially, with evaluation of each script result and resetting the method initialization between the iterations. The finished experiment provides a set of quality measurement values for each of the results. This mode can be especially useful for finding the optimal parameter set for a specific image class.
4. The last experiment mode provides a way to run the method evaluation process on different data sets. By utilizing the simulated image data, this mode can be used to investigate the influence of the MR acquisition parameters on the segmentation result. This scenario requires a set of image data generated with Virtual MRI Scanner using different imaging parameters, but originated from the same Problemator-defined phantom. Next, the selected method is executed and validated on each of the image set elements. This approach can help to find a set of acquisition parameters suitable for the specific method, in contrast to the previous modes, where the method is adjusted to the given data.

3.5.2. *Quality metrics*

After each iteration of the experiment, the resulting segmentation is exported to a bitmap format and compared with the "ground-truth" reference. The segmentation quality is calculated with the help of three commonly used segmentation error measures: Volume Overlap Error (VOE), Relative Volume Difference (RVD) and Average Symmetric Surface Distance (ASD). Generally, a lower absolute value of each of the metrics indicates a better segmentation, with the 0 value being the ideal case (see [24] for full definitions).

3.6. *Implementation concerns and availability*

The framework is implemented mostly in the Java language, using the SE 7 version of the platform. The client applications are Java applets running in a browser plugin, using the Swing toolkit for the user interface and Java3D library for 3D capabilities. The server back-end is based on the Axis2/Java web service engine [31], deployed on a Tomcat server [32], alongside the data repository and LDAP applications (see Fig. 7). The segmentation framework is implemented purely in Java, while the core computational logic of the Virtual MRI Scanner module is written in C++ and incorporated in the web service using Java Native Interface. The MRI simulator kernel algorithms were parallelized with the use of OpenMP Application Program Interface.

The server and repository is deployed on a rack server with quad-core Intel Xeon E5620 CPU and 32 GB RAM, running Linux 2.6. As the computations are performed on the server side, the client applications require a standard-grade computer with Internet connection and a modern 3D graphics card.

The platform is freely available through the website: http://mesa.wi.pb.edu.pl. An unregistered user has only limited access to the framework functionality. To fully utilize its capabilities, please contact the administrator for the registration information.

4. Usage examples

This section demonstrates the system capabilities on three examples. The first one shows an evaluation process of the 2D snake method, tested with a data generated by the Problemator and the Virtual MRI Scanner modules. The second example presents a case of 3D segmentation on a real medical

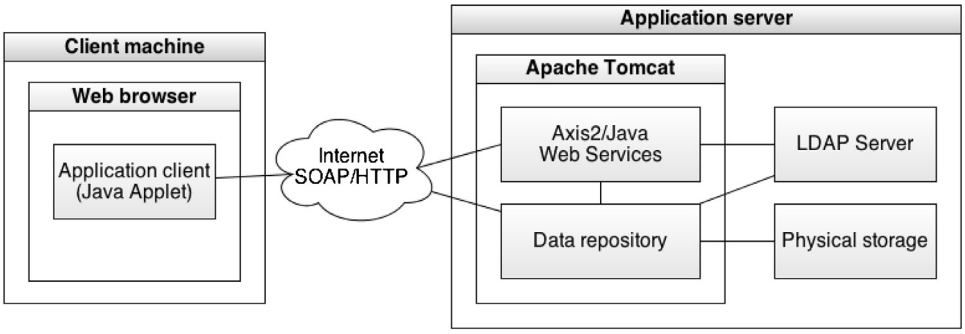


Fig. 7 – Diagram of the application system.

data set. Finally, the scripting capabilities are shown in the third example.

4.1. *2D complete workflow*

This example shows an experimental validation process of a 2D snake segmenting a liver, accordingly to the proposed workflow. The Problemator module was used to create a specific abdominal phantom (Fig. 8(a)) that brings out the difficulties with the liver segmentation, like irregularities of the organ shape, ambiguous boundaries with adjacent organs and presence of blood vessels, lymph nodes and pathological changes within the organ. Next, the phantom was used to generate a set of simulated images (Fig. 8(b) and (c)) with the Virtual MRI Scanner module. The T1 and T2 parameters for different tissues were set according to [33,34], while the proton density was set to 1 for all tissues. The images and reference segmentation were stored in the repository and served as an input for the Segmentator module. The tested method built with this module was based on a energy-minimizing discrete 2D active contour [35] with a dynamic topology adaptation extension [16]. The method used a single energy that allowed the contour to expand to the positions with an image intensity similar to the intensity of the pixels covered by the initial

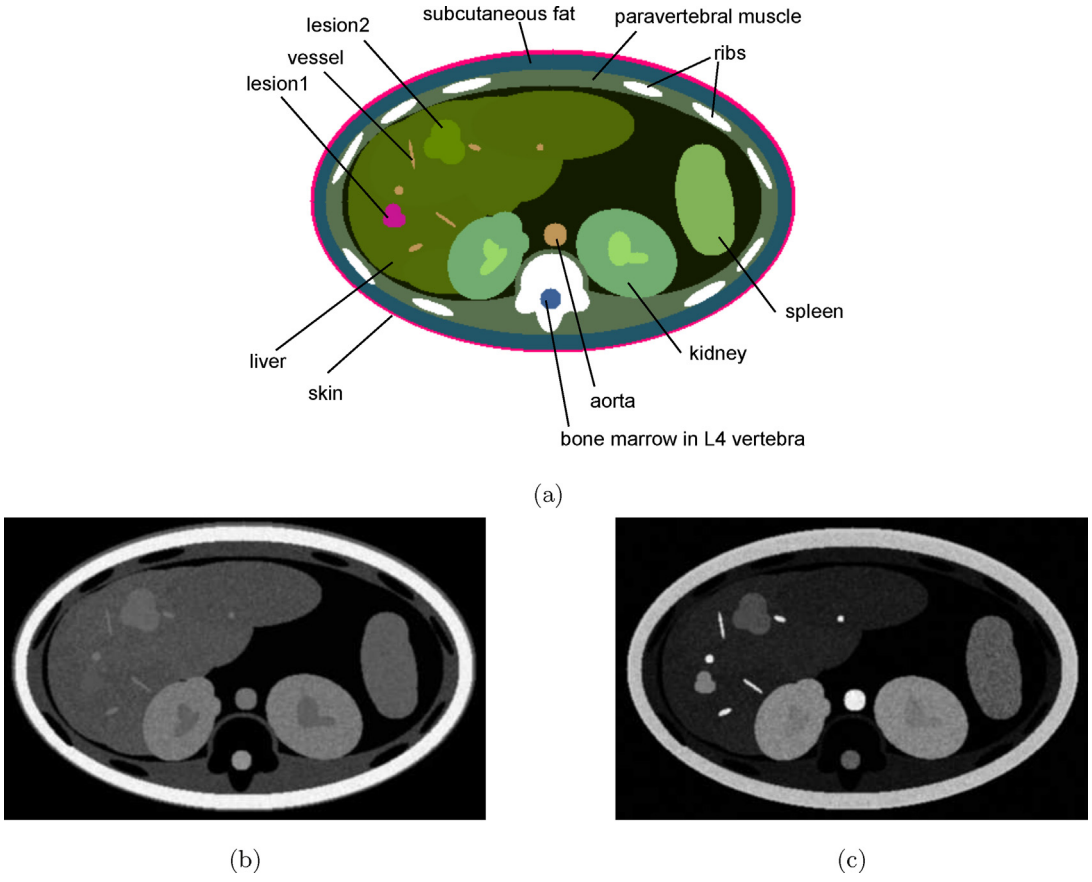


Fig. 8 – Input data for the 2D example: (a) structure of the phantom created by the Problemator and simulated MR images with the use of spin echo sequence (SE) with the following parameters: (b) TR = 500 ms, TE = 40 ms; (c) TR = 1000 ms, TE = 100 ms.

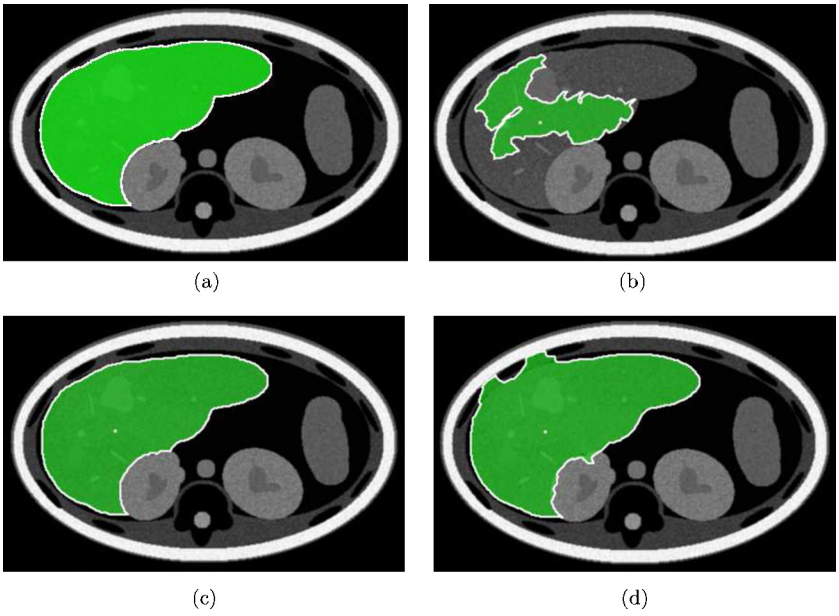


Fig. 9 – Output segmentation of the 2D example: (a) referential segmentation from the Problemator module and the tested method result for (b) $\sigma = 2$, (c) $\sigma = 5$ and (d) $\sigma = 9$.

contour. A single double-precision number (the σ variable) was the only method parameter. Finally, the method was evaluated on the simulated images in the Ring module. The experiment goal was to find an optimal value of the σ parameter in the interval $\{\sigma|0 \leq \sigma \leq 20\}$, with the step of 1. Selected segmentation results are presented in Fig. 9 and the calculated error measures are shown in Fig. 10. The results clearly show the optimal interval ($\{\sigma|5 \leq \sigma \leq 8\}$), where the results are the closest to the ideal segmentation. The values lower than 5 are too "strict" to include the entire organ (see Fig. 9(b)), while values greater than 8 cause a "leakage" of the contour and result in an over-segmentation (see Fig. 9(d)).

The entire experiment in the Ring module took about 55 s, while the MRI simulation took about 6 min for each image of size of 512 × 512 pixels. The simulation time is dependent on the result image size, e.g. it takes about 30 s for the images of size of 256 × 256 pixels and less than 10 s for 128 × 128 pixels. The average single image segmentation time was only about 1.5 s, which can indicate a potential for a practical application of the tested method.

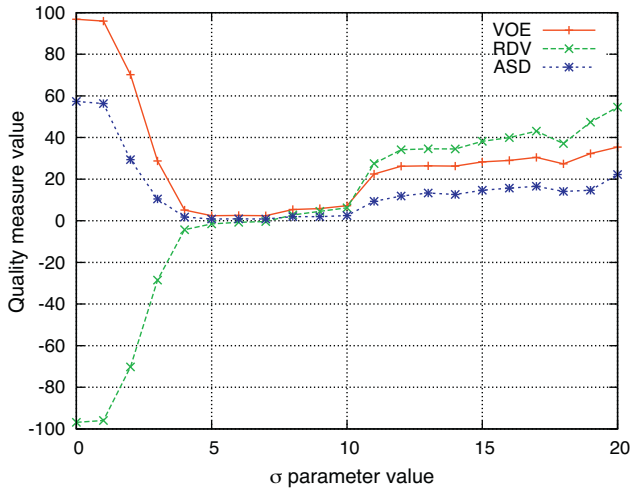


Fig. 10 – Results of the experimental validation of the 2D example method for $\{\sigma|0 \leq \sigma \leq 20\}$.

The Ring module provided a controlled environment for the execution of the experiment and result management. Each of the partial evaluation results is stored along with the final contour form and the corresponding method script.

4.2. *3D segmentation*

This example shows an experiment performed on an actual medical data, obtained from a 3D-IRCADb [6] database – a publicly available repository of medical images with provided manual segmentation references, performed by clinical experts. The analyzed data set (db1.12) contains an abdominal CT scan of 260 slices (images). This example demonstrates a segmentation of the hepatic tumor (see Fig. 11(a) and (b)), performed with the default adaptive active surface [36] and evaluated against the provided reference.

The 3D method used two energies: an image intensity energy (analogical to the 2D example) and a curvature energy, which controls the smoothness of the mesh. Both energies use a single tuning parameter. The optimal parametrization of the method was achieved by running the automated validation on a set of the possible parameter values, calculated as a Cartesian product of the given value ranges, i.e., [0, 5] with the step of 0.5 for the region and [0.1, 1] with 0.1 step for the curvature energies. The segmentation results are presented in Fig. 11 and the quality metrics are shown in Fig. 12. Both measures show the best result for the region parameter value of 1.5 and for the curvature of 0.4: at this point, the VOE metric is in its global minimum (see Fig. 12(a)), while the RVD is the closest to the perfect value of 0 (see Fig. 12(b)).

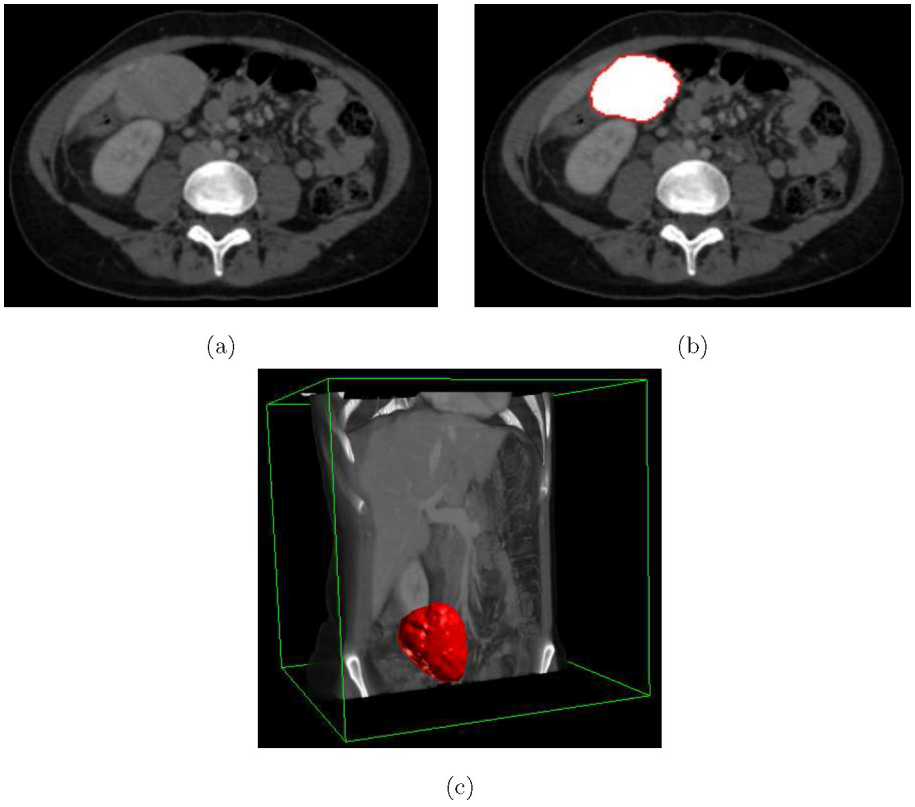


Fig. 11 – 3D segmentation example: (a) source hepatic tumor image, (b) the provided reference and (c) a sample 3D result visualization (the segmented surface visible in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

This particular example exhibited a considerable computational workload and demonstrated the benefit of client–server design. The experiment consisted of 110 segmentation tasks, each resulting in final 3D surface with about 5000 triangles. While the segmentation algorithm itself was relatively fast (about 8 s for each task), the entire experiment took about 35 min. The remote execution of the computations not only takes the workload from the client, but also provides a stable and reliable execution environment, independent of the client machine capabilities.

4.3. *New components scripting*

This section demonstrates the scripting capabilities of the Segmentator module. The example in Fig. 13(a) shows a simple extension of a 2D active contour. The extension iterates

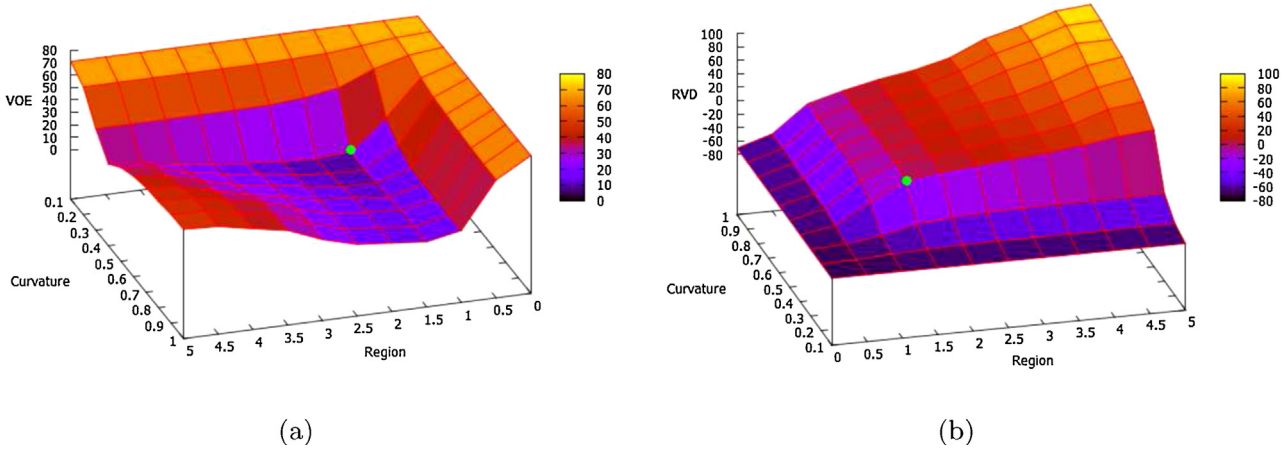


Fig. 12 – Results of the experimental validation of the 3D example method: (a) VOE and (b) RVD metrics according to the region and curvature parameters.

```
1 snake.log("My first snake extension");
2 Vector2d trans = new Vector2d(4,0);
3 for(int i=0; i<points.size();++i) {
4         Snaxel s = points.get(i);
5         s.translate(trans);
6 }
```

(a)

```
1 vec = new Vector2d(4,0);
2 for(Snaxel s: points) {
3         s.translate(vec);
4 }
5 snake.log("My first snake extension v2");
```

(b)

Fig. 13 – Source code of a simple 2D extension: (a) defined with the standard Java syntax and (b) using the Beanshell enhancements.

```
1  Point dest = p.getDestination();
2  if (dest == null) {
3          energy = 1;
4  } else {
5      if (Math.abs(snake.getRasterValue(dest) - snake.getStartMean())
6          <= sigma*snake.getStartStdDev()) {
7          energy = 1.0 / (p.startPosition.distance(p.getDestination()) + 1);
8          } else {
9                  energy = 1;
10         }
11 }
```

Fig. 14 – Source code of a sample 2D energy.

through the points of the curve and translates each of them by a given vector. The script uses two of the available run-time injected variables: *snake* – the contour, and *points* – a list of the contour points. Firstly, a message is added to the snake debug output with the *log* method. A simple "for" loop is then used to iterate through the list of points and translate them by a vector, defined by variable *trans*. This example can be simplified and shortened by taking advantage of a more modern "for-each" loop and dynamic typing, provided by Beanshell (see Fig. 13(b)).

Another example in Fig. 14 demonstrates an implementation of a 2D region intensity energy, used in the experiment in Section 4.1. In the case of a 2D discrete contour, the script has to calculate the energy value for the given snaxel *p* that is trying to move to a new position. Firstly, the energy retrieves the new position with the *Snaxel::getDestination* method. The assignment of the destination is controlled by the segmentation model. Since different models vary in their behavior, the energy has to check if the destination point is actually available. Without the new position, the code assigns the maximal return value of 1 to the built-in variable *energy*. In the opposite case, the energy calculates the absolute value of the difference between the image intensity in the destination position (*Snake::getRasterValue* method) and the initial contour intensity mean (*Snake::getStartMean*). Next, the code checks if this difference is lower or equal to the intensity standard deviation of the pixels included in the initial contour (*Snake::getStartStdDev*), multiplied by the *sigma* input variable. If this condition is reached, the energy is assigned a value inversely proportional to the distance between the destination and the *p* start position, otherwise the energy returns 1. This implementation gives smaller energy values for the more distant snaxels, enabling the expansion of the contour during the global energy minimization process.

5. Conclusions and future works

In this paper a web platform for development of medical image segmentation methods is presented. The platform offers a complete infrastructure for the method development and evaluation, providing a segmentation framework for 2D and 3D data, modules for test data generation and experiment planning and execution, as well as content storage functionalities. The framework is currently further evaluated and is still continuously developed.

At the moment, we are integrating a parallel computing platform [37] in order to distribute the computations over a cluster of commodity machines. This distributed approach aims for the improvement of the experiment performance, especially in the case of large parameter sets during the parametric study in the Ring module. The parallel platform will utilize heterogeneous cluster hardware and general-purpose computing capabilities of graphics processing units.

We are also working on stand-alone version of the client applications that will offer all the functionalities of the on-line counterparts, but with greater extensibility and ability to perform the computations off-line on the client machine.

Financial support

This work was supported with the European funds in the frame of the project ''Information Platform TEWI'' (Innovative Economy Programme) and by the grant S/WI/2/2013 from Bialystok University of Technology.

REFERENCES

[1] Meinzer HP, Thorn M, Cardenas CE. Computerized planning of liver surgery – an overview. Computers & Graphics 2002;26:569–76.
[2] MathWorks MATLAB; 2013, http://www.mathworks.com/products/matlab/.
[3] Wolfram Mathematica Technical Computing Software; 2013, http://www.wolfram.com/mathematica/.
[4] ImageJ – image processing and analysis in Java; 2013, http://rsbweb.nih.gov/ij/.
[5] 3D Slicer; 2013, http://www.slicer.org.
[6] 3D-IRCADb – 3D image reconstruction for comparison of algorithms database; 2013, http://www.ircad.fr/softwares/3Dircadb/3Dircadb.php.
[7] Campadelli P, Casiraghi E, Esposito A. Liver segmentation from computed tomography scans: a survey and a new algorithm. Artificial Intelligence in Medicine 2009;45:185–96.
[8] Kuperman V. Magnetic resonance imaging – physical principles and applications. San Diego: Academic Press; 2000.
[9] Westbrook C, Roth CK, Talbot J. MRI in practice. 4th ed. Oxford: Wiley-Blackwell; 2011.
[10] Bloch F, Hansen WW, Packard M. Nuclear induction. Physical Review 1946;69:127.
[11] Aibinu AM, Salami MJE, Shafie AA, Najeeb AR. MRI reconstruction using discrete Fourier transform: a tutorial. World Academy of Science Engineering and Technology 2008;18:179–85.
[12] Kass M, Witkin A, Terzopoulos D. Snakes: active contour models. International Journal of Computer Vision 1988;1 (4):321–31.
[13] Cohen LD. On active contour models and balloons. CVGIP: Image Understanding 1991;53:211–8.
[14] Xu C, Prince JL. Snakes, shapes, and gradient vector flow. IEEE Transactions on Image Processing 1998;7:359–69.
[15] Gunn SF, Nixon MS. A robust snake implementation; a dual active contour. IEEE Transactions on Pattern Analysis and Machine Intelligence 1997;19:63–8.
[16] Mcinerney T, Terzopoulos D. T-snakes: topology adaptive snakes. Medical Image Analysis 2000;4:73–91.
[17] Sethian JA. Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge University Press; 1999.
[18] Jalba AC, Wilkinson MHF, Roerdink JBTM. CPM: a deformable model for shape recovery and segmentation based on charged particles. IEEE Transactions on Pattern Analysis and Machine Intelligence 2004;26.
[19] Miller JV, Breen DE, Lorensen WE, O'Bara RM, Wozny MJ. Geometrically deformed models: a method for extracting closed geometric models form volume data. SIGGRAPH Computer Graphics 1991;25:217–26.
[20] Montagnat J, Delingette W, Ayache N. A review of deformable surfaces: topology, geometry and deformation. Image and Vision Computing 2001;19:1023–40.
[21] Heimann T, Meinzer HP. Statistical shape models for 3D medical image segmentation: a review. Medical Image Analysis 2009;13:543–63.
[22] McInerney T, Terzopoulos D. Deformable models in medical image analysis: a survey. Medical Image Analysis 1996;1:91–108.
[23] Tsechpenakis G. Deformable model-based medical image segmentation. Multi modality state-of-the-art medical image segmentation and registration methodologies. Springer Publishing; 2011. pp. 33–67.
[24] Heimann T, van Ginneken B, Styner MA, et al. Comparison and evaluation of methods for liver segmentation from CT datasets. IEEE Transactions on Medical Imaging 2009;28:1251–65.
[25] Bittoun J, Taquin J, Sauzade M. A computer algorithm for the simulation of any nuclear magnetic resonance (NMR) imaging method. Magnetic Resonance Imaging 1984;2:113–20.
[26] Jurczuk K, Kretowski M. Virtual magnetic resonance imaging – parallel implementation in a cluster computing environment. Biocybernetics and Biomedical Engineering 2009;29:31–46.
[27] BeanShell – lightweight scripting for Java; 2005, http://www.beanshell.org.
[28] Rhodes ML, Glenn WV, Azaawi YM. Extracting oblique planes from serial CT sections. Journal of Computer Assisted Tomography 1980;4:649–57.
[29] Cabral B, Cam N, Foran J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. ACM Symposium on Volume Visualization. 1994. pp. 91–8.
[30] Thon S, Gesquiere G, Raffin R. A low cost antialiased space filled voxelization. GraphiCon of Polygonal Objects. 2004. pp. 71–8.
[31] Apache Axis2/Java – next generation Web services; 2013, http://axis.apache.org/axis2/java/core/.
[32] Apache Tomcat; 2013, http://tomcat.apache.org/.
[33] Stanisz GJ, Odrobina EE, Pun J, Escaravage M, Graham SJ, Bronskill MJ, Henkelman RM. T1, T2 relaxation and magnetization transfer in tissue at 3T. Magnetic Resonance in Medicine 2005;54:507–12.
[34] de Bazelaire CMJ, Duhamel GD, Rofsky NM, Alsop DC. MR imaging relaxation times of abdominal and pelvic tissues measured in vivo at 3.0 T: preliminary results. Radiology 2004;230:652–9.
[35] Reska D, Kretowski M. HIST – an application for segmentation of hepatic images. Zeszyty Naukowe Politechniki Bialostockiej Informatyka 2011;7:71–93.
[36] Reska D, Boldak C, Kretowski M. Fast 3D segmentation of hepatic images combining region and boundary criteria. Image Processing & Communications 2012;17:31–8.
[37] Reska D, Boldak C, Kretowski M. A distributed approach for development of deformable model-based segmentation methods.Image processing and communications challenges 5. Advances in intelligent and soft computing, vol. 233. 2014. pp. 21–28.