Open-source software platform for medical image segmentation applications

R. Namías, J. P. D'Amato, M. del Fresno

R. Namías, J. P. D'Amato, M. del Fresno, "Open-source software platform for medical image segmentation applications," Proc. SPIE 10572, 13th International Conference on Medical Information Processing and Analysis, 105721J (17 November 2017); doi: 10.1117/12.2283487

SPIE.

Event: 13th International Symposium on Medical Information Processing and Analysis, 2017, San Andres Island, Colombia

Open-source software platform for medical image segmentation applications

R. Namías[a,c], J. P. D'Amato[b,c], and M. del Fresno[b,d]

[a]CIFASIS, UNR-CONICET/UAM (France), Rosario, Argentina
[b]Instituto PLADEMA, Universidad Nacional del Centro, Tandil, Argentina
[c]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina
[d]Comisión de Investigaciones Científicas de la Prov. de Buenos Aires (CIC-PBA), Argentina

ABSTRACT

Segmenting 2D and 3D images is a crucial and challenging problem in medical image analysis. Although several image segmentation algorithms have been proposed for different applications, no universal method currently exists. Moreover, their use is usually limited when detection of complex and multiple adjacent objects of interest is needed. In addition, the continually increasing volumes of medical imaging scans require more efficient segmentation software design and highly usable applications. In this context, we present an extension of our previous segmentation framework which allows the combination of existing explicit deformable models in an efficient and transparent way, handling simultaneously different segmentation strategies and interacting with a graphic user interface (GUI). We include the object-oriented design and the general architecture which consist of two layers: the GUI at the top layer, and the processing core filters at the bottom layer.

We apply the framework for segmenting different real-case medical image scenarios on public available datasets including bladder and prostate segmentation from 2D MRI, and heart segmentation in 3D CT. Our experiments on these concrete problems show that this framework facilitates complex and multi-object segmentation goals while providing a fast prototyping open-source segmentation tool.

Keywords: Medical Imaging Analysis , Multiple Segmentation Framework , Parallel Segmentation

1. INTRODUCTION

With increasing use of medical imaging modalities like Computed Tomography (CT) and Magnetic Resonance imaging (MRI) for diagnosis, treatment planning and clinical studies, it has become almost compulsory using computers to assist radiological experts in their tasks.[1] Hence, computer-aided diagnosis (CAD) systems have become one of the major research areas in medical imaging and diagnostic radiology.[2]

In particular, medical image segmentation is an essential part of CAD systems, which usually determines the eventual success or failure of the subsequent steps of image analysis. Segmentation basically consist in delimiting the structures or regions of interest (ROI), and discriminating them from the background. This process is used to detect organs, such as brain, lungs or liver in CT or MRI scans; or for the extraction of pathological tissues, such as a tumors or plaques.

Although there is a wide range of proposed methods to solve the problem, no universal algorithm for segmentation of every medical image exists. Some methods are more general as compared to specialized algorithms, and can be applied to a wider range of data. Nevertheless, segmenting anatomical or functional structures from medical images in an efficient and accurate way is still an open and challenging problem .

CAD systems frequently require delimiting multiple adjacent structures, for instance, multiple organs in 3D abdominal CT,[3,4] brain tumor segmentation,[5,6] pelvic organ prolapse,[7] abdominal anatomy segmentation in MRI, among others. Other challenging problems demand segmenting complex objects such as cardiac structures in CT,[8],[9] colon segmentation for virtual colon unfolding in virtual colonoscopy,[10] uterus delineation in dynamic

Further author information: (Send correspondence to J.P.D.)
J.P.D..: E-mail: jpdamato@exa.unicen.edu.ar, Telephone: 54 2494 4385690

13th International Conference on Medical Information Processing and Analysis, edited by Eduardo Romero, Natasha Lepore, Jorge Brieva, Juan David García, Proc. of SPIE Vol. 10572, 105721J
 · doi: 10.1117/12.2283487

MRI.[11] Most of these systems focus on multiple objects detection or complex structures segmentation. To the best of our knowledge, no system can mix the best segmentation techniques for each ROI or share spatial restrictions derived from the implicated models.

In general, when a novel medical imaging segmentation problem arises, researchers adapt previous methods, or propose new specific segmentation techniques. Therefore, a significant effort has been dedicated to the improvement of a wide range of segmentation algorithms. Nevertheless, little research has been undertaken for the integration and combination of existing techniques. Within the context of concrete segmentation applications, implementations often does not extend beyond the prototype version. Moreover, because of the high computational workload required, along with the continually increasing medical imaging scans sizes, it become indispensable a more efficient software design and development to extend their limits.

Several general medical imaging processing software toolkits are available for researching, prototyping and application development[12] (Eg: Julius,[13] Sim ITK-VTK,[14] Nifty-Seg). Most of them use a pipeline architecture (tubes and filters) where each filter implements a different stage of the process. Nevertheless, these tools are usually closed-source (privative), too general or simply not adaptable for further improvement.

The present paper proposes an extension and detailed design of the open-source segmentation framework named *Deformable Models Array* (DMA).[15] This framework proposes an extensible toolkit to be adapted to different medical segmentation tasks using a combination of explicit deformable models (DM). The different DM can interact either in a 2D or 3D normalized space while they evolve, according to their conditions. We have introduced an object-oriented design based on the Insight Toolkit (ITK) libraries, which allows to handle different segmentation strategies. For instance, simple segmentation (one ROI only), multiple segmentation (several ROIs, adjacent or isolated), and complex segmentation (one complex ROI with different image characteristics), including a novel cooperative interaction scheme to control the models interactions. DMA can handle large amounts of geometric and image data in both 2D and 3D scenarios using parallel architectures. Moreover, one of its main advantages is the possibility of easily include existing DM algorithms to quickly develop and test segmentation strategies in different medical imaging modalities.

The original proposal has been extended, and the framework re-designed to include new functionality such as volume difference metrics and distributed processing in order to test several algorithms in an efficient and transparent manner. DMA is open-source and runs on a PC computer under any common OS (eg: Microsoft Windows, Linux or iOS). The framework offers a user-friendly environment that lets users to prepare the inputs and to visualize the results. The initialization module provides different DM techniques to be applied for the particular segmentation tasks and allows the creation of mesh models to facilitate the experiment set-up. Then, the visualization module enables the interactive inspection of the resulting mesh models superimposed to the image volume.

The rest of the paper is organized as follows: Section 2 describes the related work. Section 3 introduces the framework design. Section 4 explains the used methods, presents the medical image datasets, and describes the experiments. Next, Section 5 shows the application results. Finally, the paper is concluded in Section 6 with a further discussion of the usage and performance of DMA.

2. RELATED WORK

Segmentation methods based on deformable models have come into the spotlight in the field of medical applications.[16] DM are curves (in 2D) or surfaces (in 3D) defined within the image domain, which are deformed under the influence of internal and external forces. Depending on their representation, the DM family is commonly subdivided into two groups: explicit methods and implicit methods.[16] The explicit methods were introduced with the original snakes presented by Kass *et al.*[17] whereas the implicit ones (level sets) were first proposed by Malladi *et al.*[18] Since those original proposals, several improvements and extensions have been presented to overcome their original limitations. While level-sets are an attractive mathematical framework, the implicit formulation are not nearly as convenient as explicit, parametric formulations for incorporating additional control mechanisms. Figure 1 describes a DM evolution represented by a polygonal. The initial model state is depicted in red, the middle states during the evolution are shown in green, and the final result appears as a thick yellow line.

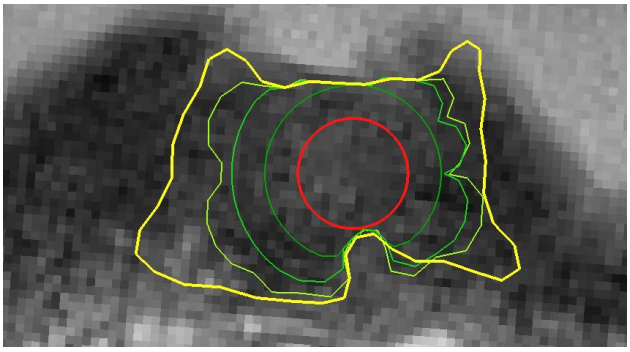

Figure 1. Explicit DM evolution example. In red the initial polygonal state, in green the middle states and in thick yellow line the final result.

Several works have been proposed to resolve particular segmentation problems and significant effort has been dedicated to the improvement of the algorithms. Abe and Matsuzawa[19] propose the use multiple active contour models (ACM) to segment a single object. Around the object boundary, the ACMs compete with each other and each ACM extracts a subregion of uniform image properties. The result is the union of a set of subregions. Chen and Metaxas[20] present a hybrid segmentation framework, based on deformable models, where a global energy function is introduced to combine prior-shape, region-based and boundary techniques under a fixed schema for 3D brain image segmentation. In,[21] a combined strategy for 3D volume segmentation is considered, using first a region growing algorithm for a robust approximation of the objects of interest and then a deformable model strategy for the final refinement of the segmented meshes.

Under the realm of level sets formulations, Shang *et al.*[22] have developed a region competition active contour model. The algorithm is derived by minimizing a region based probabilistic energy function and implemented in a level set framework. The model combines edge and region features in a level set evolution equation and has been extended to 3D to extract medical objects. Being based on a level set algorithm, the model can deal with topological changes in a natural way. It uses the minimization of a region based probabilistic energy function. The competing regions are only the ROI and the background, and it cannot manage multiple segmentations. Other authors[23,24] have used variations of the Chan-Vese formulation[25] to segment the image in more than one object plus the background, searching for homogeneous intensity regions. Nevertheless, this hypothesis does not hold for some medical imaging modalities because two distinct organs can present similar intensities.

Additionally, Gao *et al.*[26] have developed a 3D multi-object segmentation tool, which not only grants mutual exclusion between regions but also parallel segmentation and models interaction using a principle of action and reaction. The authors have introduced a particular novel active contour formulation for this framework which, however, allows neither segmenting complex structures nor changing the interaction scheme.

On contrary, DMA implements complex- and multi-object segmentation as a whole, both in 2D and 3D images considering a simple and easy formulation. As it was previously mentioned, it may advantageously employ any existing explicit DM to tackle diverse segmentation goals.

3. FRAMEWORK DESIGN

The *Deformable Models Array* main purpose is the medical image segmentation of multiple and complex anatomical structures. To obtain good quality results, the framework has to adapt to different images modalities (e.g. MRI, CT), and dimensions: 2D and 3D. Although there is not a segmentation technique which can achieve high quality results in any scenario, the correct selection of existing state-of-the-art techniques in each case could solve this issue. Particularly, explicit deformable models are quite suitable techniques for parallel and interacting segmentation due to their capability to incorporate control mechanism such as evolution restrictions, collision detection, shape guidance, etc. These models are geometrically represented by a closed polygonal for 2D images and by a triangular surface mesh for 3D volumes, which can be initialized with a given geometry or from the result of a previous region growing segmentation step.[21] As an example, Figure 2 presents a 3D multisegmentation problem of different pelvic organs in MRI. In this case, two previous segmented organs (in yellow) were

considered as evolution restrictions for the evolutive meshes (in green). The initial models are depicted on the left and the final result using DMA on the right.

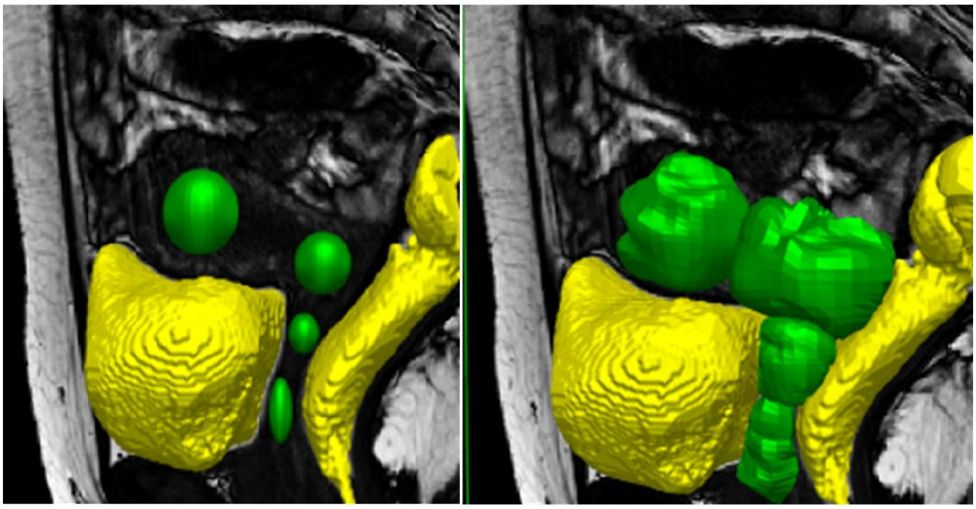

Figure 2. 3D multiple segmentation problem.

3.1 General framework design

The framework design has a two-layered architecture. At the top layer we find the user interface, and at the bottom, the processing layer (Figure 3). This scheme let us separate the presentation module from the processing options. The full block diagrams depict the main modules, the dashed lines blocks group modules into standard and custom filters, the arrows represent data flow between modules and the thick horizontal line separates the two layers.

The required input data comprises: the image to be segmented, a set of initial models called evolving models (e_M), a set of deformable model techniques (dm_T) to evolve the e_M, and the parameters of each dm_T. Hence, these two items are grouped in tuples (e_M, dm_T), one for each ROI. Next, DMA enables to include previous segmentation knowledge by mean of a set of spatial restrictions known as fixed models (f_M). Finally, a few framework parameters, to be described later, must be set to run the segmentation.

Consequently, a convenient GUI was designed using the Qt libraries[27] to facilitate the DMA experiment set-up, enabling to load and save the complete project state, visualize the scene, etc. We are now working in a new User Interface based on WEB Technology, in order to improve the data delivery and presentation.

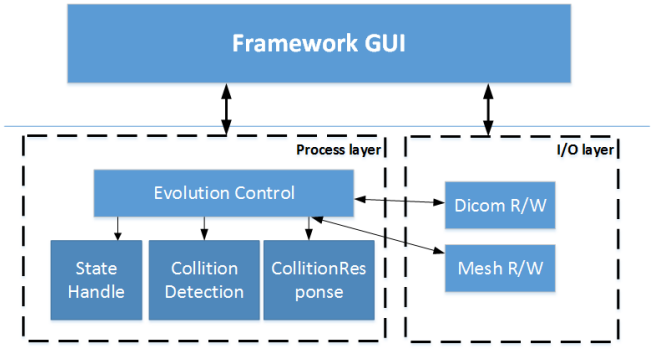

Figure 3. DMA two-layers framework architecture. Top layer: GUI implemented with Qt. Bottom layer: the processing core filters grouped in Standard and Custom ITK Filters.

Beneath this top level layer, there are two main groups embedded in the ITK libraries. The the most important one implements the segmentation process ruled by the *evolution control* (*EvCtrl*) module, followed by the *collision detector* (*ColD*), *safe state* (*SS*) and *collision resolution* (*ColR*) modules, which are also later described. The remaining standard functionality group consists of image and mesh reading and writing filters, which are standard ITK filters.

3.2 Modules

The processing layer software was designed based on the information hiding technique. Each module has its specific functionality, hiding the actual implemented technique while improving the adaptability and maintenance of the software. For instance, the only requirement to replace any module is to keep its abstract interface. Figure 4 shows an UML class diagram which presents the modules main interface methods, and covers most of the associations, dependencies and relationships of the modules which comprise this layer. Following in this section, we describe the role, interface, interaction, dependencies and challenges of each of the processing modules, resembling a module guide.

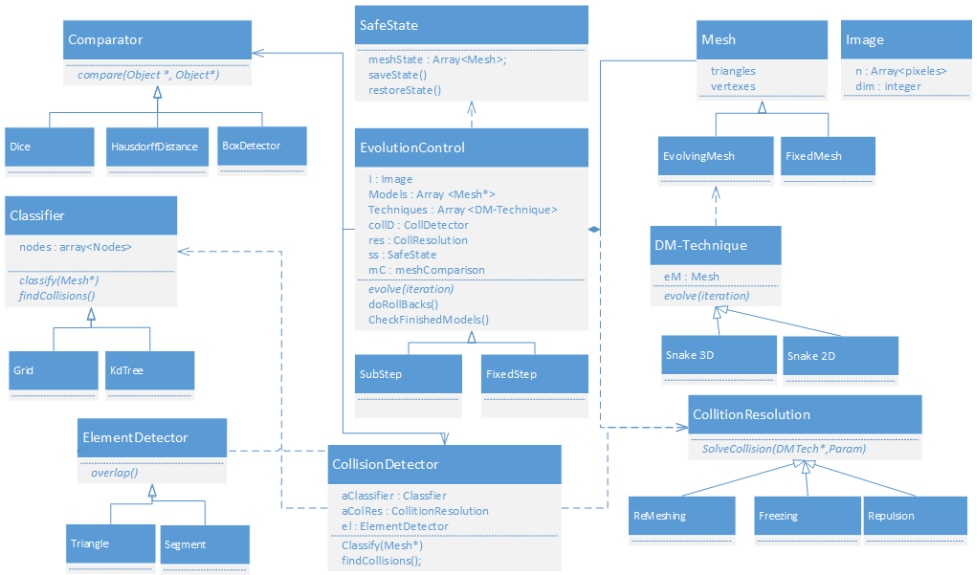

Figure 4. (UML) DMA processing layer class relationship scheme, including the modules most important interface methods.

3.2.1 Evolution Control module

The *EvCtrl* module is the main coordinator of the segmentation process. Its function is to synchronize all the dm_T, and to control their interaction. The interface provides methods to set its inputs: the e_M+f_M models, the dm_T with their parameters, the input image, and also the DM maximum number of iterations (*Max_It*), and the *synchronization distance* parameter(N), used for the *ColD* and *ColR* modules. As each e_M needs one instance of an *SS* object, as well as one dm_T object, it keeps a list of SS objects, and another list with dm_T objects.

On the one hand, the *SS* keeps a history of previous states of the particular e_M to be able to restore their state to the last collision-safe state or *checkpoint* whenever is required by the *EvCtrl* module. On the other hand, the dm_T guide the e_M evolution across the iterations. Next, the *ColD* module tracks the e_M evolutions and interactions, looking for collisions between or inside each e_M. The *ColR* module is able to resolve the collisions, and is implemented as a Visitor design pattern because it visits the dm_T objects modifying the e_M evolution when a collision is detected.The *advance* method, iterates through all techniques.

The *evolution control* main module first calls the *ColD* to check for initial collisions and the *SS* module to save the first safe states. Then, the evolution loop starts: the dm_T make N iterations, consequently the *ColD* checks for collisions, and if any occurs, the *EvCtrl* ask the different dm_T to solve the collision problems by calling

the *ColR* object. The *SS* module saves a list of e_M states. The method `restoreState()` is invoked when a rollBack is required.

The *SS* module uses the saved states to restore the models to a safe configuration previous to a mesh overlapping. During evolution, whenever it is necessary, the *EvCtrl* can rollback the e_M to any of the saved states, apply a collision solution to the problematic nodes, and re-evolve the e_M towards the current synchronization point. Finally, the new *checkpoint* is saved in the *SS* module. The loop continues until all dm_T have finished their evolution. Below, the method *evolve()* represents the *EvCtrl* main loop calling all the presented modules.

```
1  class EvolutionControl:evolve()
2  /* No collisions at the beginning */
3  if(ColD->findCollisions())
4    abort("Collisions before start.");
5  /* Save first safe state in the SS */
6  SS->SaveStates();
7  while ( !allModelsFinished )
8  {
9   /* Evolve the models  */
10     this->advance(step);
11     /* Check Collisions */
12     collisions=ColD->findCollisions();
13     /* Manage Collisions */
14     while(collisions)
15     {
16        SS->restoreStates();
17        ColR->SolveCollisions();
18        this->advanceNIterations(step);
19        collisions=ColD->findCollisions()
20     }
21     /* Save Checkpoint */
22     SS->SaveStates()
23     /* Control the stop criterion */
24     if (CheckFinishedModels())
25        break;
26 } /* End while */
```

As it was empirically tested, the initial meshes, associated with the e_M+f_M models, are far from each other. As a result, long evolution steps can be used during the first iterations. But, once collisions are detected, a small evolution step is recommended.

The evolution algorithm may be extended to handle different situations, using either of two strategies: *fixed-step evolution* or *sub-step advance and restore*. In the first one, the algorithm has a constant step. In the second one, the step is adaptative. Consequently, the *AdvanceNIterations* method may also need to be changed.

3.2.2 Collision Detection

This module implements a collision detector algorithm. This algorithm concerns with the detection of intersecting geometrical elements from different models or elements of the same model (self-collisions). It is worth noting that if a collision is not solved, the segmented models would be invalid, with meshes struggling or overlapping. Hence, the *collision detector* should efficiently search for potential elements in conflict and, if a collision is detected, a resolution strategy should be applied. Elements collision checking is performed several times almost in every iteration, adding a high computational cost to the process. This behavior could be implemented both in 2D and 3D, according to the problem dimension.

The collision detection algorithm is computationally expensive, particularly in 3D. In real time solutions, grids or kd-trees are good approaches to find elements in the space. Some works have already dealt with mesh-to-mesh collisions in medical simulations in an efficient way.[28] They basically use classifications schemes to do real time collision detection. Generally, if the collision detection is computed in each iteration, the detection time is higher than the dm_T evolution time.

The framework can accept several strategies, of both classification and detection. In this work, we have implemented for 2D a grid classifier and a segment detector and for 3D, also a grid classifier and a triangle overlapping detector, which check in parallel for superposed triangles in each cell.

3.2.3 Collision Resolution scheme

The *collision resolution* module implements a resolution strategy, in collaboration with the dm_T involved, to avoid the detected intersections. This class is used as a Visitor pattern. Every dm_T must invoke its *ColR* when a collision is detected by calling the interface method: `SolveCollision(DMTech*,Param)`. The collision details are provided as parameters to the *ColR* to solve it.

A simple but effective method to solve collisions is to "freeze" the colliding elements. These elements are marked and they are no longer updated in the next evolution steps. Another option, instead of freezing elements, is to use a proportional repulsive force, opposite to the movement of each colliding element.[29] Even though it is a more general approach, it is unstable, and in the studied cases leads to a similar result as the freezing strategy but at a higher cost.

A particular problem arises when the e_M meshes have different elements size. In this case, bigger geometrical elements should be divided and new nodes generated in valid positions. In the same way, and depending on the application, narrow elements could also be merged to reflect a continuous surface. This approach is called "remeshing". As our implementation is limited to fixed mesh structures, we omit the remeshing step, but it will be considered for future implementations.

3.2.4 DM Technique

The final module is the actual DM technique. Every concrete DM algorithm must inherit and implement the interface methods of the *DM Technique* class to work in the framework. The interface mandatory methods are `Evolve(n)`, required to evolve each model e_M for n iterations and must support a collision visitor to deal with the intersections. In this way, it is possible to easily wrap the existing explicit dm_T to use them in the framework.

To use a new dm_T in DMA it is first necessary to incorporate a new class in the framework. Second, to update the DM list in the GUI doing the new object instantiation , when selected, and passing the object pointer to the *EvCtrl* dm_T list. Finally, compile the code with the modifications.In future implementations, this could be supported in real time using Dynamic Libraries bindings.

The seudo-code below 3.2.4 presents the core of the `Evolve` method for a generic 2D Snake technique implementation. The specific model evolution takes place inside the `Advance()` method, calling `StopCriterion()` in each iteration to check if the model evolution has finished.

```
1  template <class Image, class Mesh>
2  Snake2D<Image, Mesh>::Evolve(unsigned N)
3  {
4   /* Initialization */
5   if(!Initialized)
6    Initialize();
7    it = 0;  /* Checkpoint iterations */
8    while ( (it < N) /* Checkpoint (N iterations)*/
9        && !hasFinished     /* Stop criterion */
10       && (totIts < maxIt))  /* Max iterations */
11   {
12     /* Evolve the points */
13     AdvancePoints();
14     /* Check if it has finished */
15     hasFinished = StopCriterion();
16     /* Checkpoint iterations */
17     it++;
18     /* Accumulated iterations */
19     totIts++;
```

```
20     } /* End while */
21     ...
22   } /* End Evolve */
```

Following, two explicit DM techniques formulations are described, which will be used in the examples and that are implemented in the *advancePoints* method of each sub-class.

T-Snakes First presented in,[30] are discrete approximations to a conventional parametric snakes model while retaining many of its properties. The deformation of the snake model is governed by discrete Lagrangian equations of motion and each element $\mathbf{x}(t)$ evolves according to the following motion equation:

$$\mathbf{x}(t+1) = \mathbf{x}(t) - \Delta t\,(a\alpha(t) + b\beta(t) - p\rho(t) - q\mathbf{f}(t)) \tag{1}$$

where α, β are the internal forces (tension and flexion), $\mathbf{f}$, ρ are the external forces (balloon and image gradient forces) and a, b, p and q are the force weighting parameters. The T-Snakes formulation modifies the original external force adding an adaptive inflation force $\rho(t)$ term that depends on the image intensity features.

GVF-Snakes Gradient vector flow (GVF) snakes introduce a new external force for active contour models. The difference between traditional snakes and GVF-Snakes is that the latter converge to boundary concavities and they do not need to be initialized close to the boundary.[31]

To improve the original snake formulation, the authors introduced a non-irrotational external force field $v(x,y) = [u(x,y), v(x,y)]$ known as gradient vector flow field. The field is calculated as a diffusion of the gradient vectors of a gray-level or binary edge map:

$$GVF = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |v - \nabla f|^2 dxdy \tag{2}$$

where μ is an input parameter.

3.3 Segmentation quality evaluation module

One important step in the whole segmentation process is to compare the obtained results respect to manual segmentations provided by experts, which can be considered as ground truth. Therefore, after applying a particular method, we can estimate the segmentation quality using different indicators or measures. These indicators are considered in the framework as another module, that could be invoked during the segmentation or at the end. The idea is to have a "comparison indicator" between a resulting contour (in 2D) or mesh (in 3D) or even an image respect to a reference one. To model this behavior, we define a *Comparison interface*. The main method `compare()` receives two objects (a mesh or image) and return a doubled value. This value indicates 0 when they are completely different and 1 when the objects are geometrically equal. Below, we present three different techniques to compute this metric.

3.3.1 Dice Similarity Coefficient

The first indicator is the Dice Similarity Coefficient ($\mathcal{D}$)[32,33] which in this case measures the spatial overlap accuracy between the ground truth and the DMA result. $\mathcal{D}$ basically compares the region similarity or intersection as follows:

$$\mathcal{D}(A,B) = \frac{2 \times |A \cap B|}{|A| + |B|} \tag{3}$$

where A is associated with the ground truth and and B represents the segmented regions. $\mathcal{D}$ lies between 0 and 1. The greatest the intersection between the masks, the closest to 1 is $\mathcal{D}$.

3.3.2 Hausdorff Distance

The undirected partial Hausdorff Distance ($\mathcal{H}$) indicates the degree of mismatch between two sets by measuring the farthest distance between two points of each sets.

The undirected partial Hausdorff Distance is defined as follows:

$$\mathcal{H}(A,B) = \max(h(A,B), h(B,A)), \tag{4}$$

where A and B are regions (sets of points) and

$$h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\|.$$

We use the Euclidian distance in millimeters (mm) using the pixel resolution taken from the DICOM information. For this case, the mesh should be converted to a 3D grid and then compute the measure.

3.3.3 Mesh Similarity Coefficient

To compare 3D surfaces, we could use the volumetric differences to evaluate two triangular meshes models according to the following volumetric coefficient metric, as presented in:[34]

$$\mathcal{M}(A,B) = 1 - (f_n + f_p) \tag{5}$$

$$f_n = \frac{V_{A/B}}{V_A \bigcup V_B} \quad \text{and} \quad f_p = \frac{V_{B/A}}{V_A \bigcup V_B} \tag{6}$$

where f_n is the false-negative rate, f_p is the false-positive rate, V_A is the volume of the reference model A and $V_{A/B}$ is the volume difference between A and B. In a similar way, V_B is the volume of the DMA resulting model B and $V_{B/A}$ is the volume difference between B and A. Similar to $\mathcal{D}$, $\mathcal{M}$ lies between 0 and 1 where 0 means no similarity and 1 a total similarity. As was discussed in this work, the method could run even in parallel architectures but only works with 3D meshes.

3.4 Implementation issues

The *Deformable Models Array* framework has two major implementation issues. The first one is the dimensionality problem, to allow working with either 2D and 3D images and the corresponding e_M geometrical representations. Most of the framework modules must deal with generic *Image* and *Mesh* types, which is not that simple to be solved. The second issue is the time and space complexity when working with 3D scenarios.

3.4.1 Dimensionality

Although 2D models are not large, 3D meshes could have up to 50.000 points, for a typical medical image, rapidly increasing the memory demand. The *SS* class must keep track of previous states of each mesh including particular checkpoints associated to safe-states where the mesh is free of intra- or inter- models collisions. As the meshes topology does not change in our model, it is just necessary to save the points coordinates and not the cells (segments or triangles). Anyhow, this procedure is time consuming when the mesh is considerably large.

The *collision detector* module is called, when necessary, by the *EvCtrl*. On the initial setup process, the *EvCtrl* must set the specific *ElementDetector*, depending on the dimension and mesh morphology, as well as the e_M and f_M models. For instance, a segment intersection routine is suitable for 2D and a spatial triangle intersection routine for 3D meshes. An efficient collision detection algorithm is mandatory for the volumetric meshes, otherwise it would be extremely time-consuming. Therefore, we model a general *Classifier* to encapsulate the spatial clustering methods. In 3D, we implemented a regular 3D grid, as shown in Figure 5. Each bin or cell of this grid has a set of the geometric elements that intersects it. Collisions are checked inside each bin, reducing overall computing time from $\Theta(N^2)$ to $\sum_{i=1}^{S}(n_i^2)$, where N is the total number of triangles, n_i is the number of triangles in the bin i, S is the total number of bins from the grid, and generally $n << N$. Figure 4 shows part

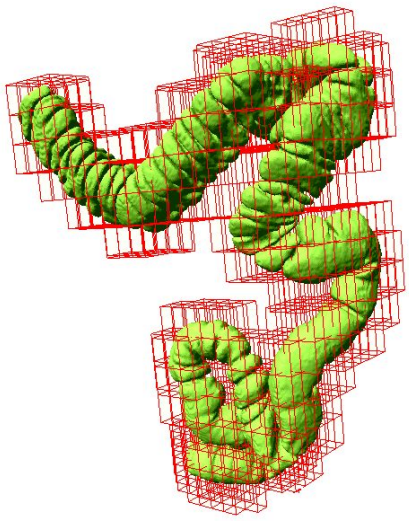

Figure 5. Rectum classified by a 3D Grid to detect self-collisions.

of the module interface where the gathered list of collisions is accessible by calling the `findIntersections()` method.

All the above described modules must work in 2D as well as in 3D. The ITK libraries architecture design, built in `C++` using templates, naturally allows to deal with dimension switch. Only the *Detector*s and the *DM-Technique* modules are dimensionally dependent, requiring concrete implementations. For the rest, is completely transparent.

3.4.2 Memory usage

Although in a 2D scenario the RAM memory in not a determining resource, a proper and efficient use of it must be considered in 3D scenarios. Not only a high resolution DICOM image can take up a lot of space (e.g.: $40 \cdot 10^6$ voxels) but also the mesh models all together can have a quarter of million points and an eight million of geometric elements (triangles). For this reason, it is prohibitive to make copies of these structures. The image is globally shared by all the DMA modules, whereas each mesh is communal to the proper dm_T and the *ColD* object. The specialized filters load the image and meshes, and hereafter, the memory pointers are passed to the necessary modules to share these memory spaces, resulting in an efficient usage of the memory space that allows to work with more complex and larger segmentation tasks.

3.5 Software and hardware requirements

Finally, it is important to remark that *Deformable Models Array* runs under any computer having CMake, QT v5.0, ITK v4.11 and VTK v5.6 libraries (or newer) installed. The minimum suggested RAM memory is 4Gb, although it actually depends on the specific segmentation scenario. Volumetric scenarios would demand more computational power and memory, while 2D scenarios are less demanding. The application, with GUI and processing modules, could also be deployed both locally or distributed as a client-server. The source code could be downloaded at *https://github.com/guidomaiola/dma-tool*.

4. EMPIRICAL ASSESSMENT

In this Section, we aim to show the efficacy and adaptability of the software design. We implemented the three quantitative metrics to assess the segmentation results (Dice, HDistance and MeshDif). Then, we describe two real-case segmentation tasks in different dimensions, image modalities, and DM techniques, which covers most of the DMA functionality. The first example is the bladder and prostate segmentation in 2D transversal MRIs,

and the other is the heart segmentation in 3D CTs using previous knowledge as constraints. Both experiments include a comparison between the stand alone techniques and the DMA results. The experiments were ran in an Intel(R) Core (TM) i5-3570 CPU @ 3.40GHz with 16Gb RAM and Linux kernel version: 3.11.0-23-generic.

4.1 Bladder and prostate in 2D-MRI

In the first experiment, we present the segmentation of two adjacent pelvic organs (bladder and prostate) from the MICCAI 2012 prostate segmentation challenge* dataset of transversal T2-weighted MR. The datasets were acquired under different clinical settings. They are multicenter and multivendor, and have different acquisition protocols (e.g., differences in slice thickness, with/ without endorectal coil). Reference segmentations of the prostate are available for the dataset and the bladder was manually segmented by an expert for the experiment purpose. We selected a single transversal slice from each of the 10 considered patients where these pelvic organs were adjacent to each other. We use between one and three e_M for the bladder as well as for the prostate according to the images inhomogeneities and organs complex shapes. For the dm_T , we use GVF-Snakes for the bladder and T-Snakes for the prostate.

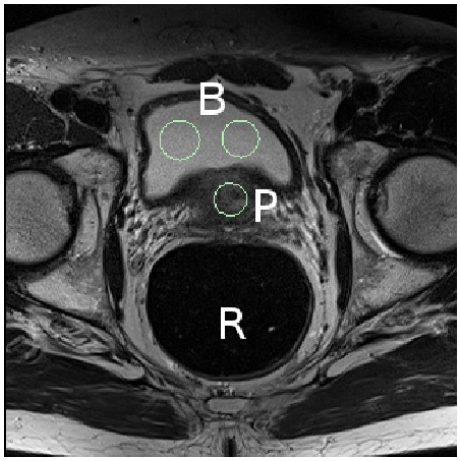

Figure 6. Bladder and prostate segmentation in transversal MRI. e_M (circles), B: bladder, P: prostate and R: Rectum.

An initialization example is shown in Figure 6. The prostate delimitation is particularly difficult because of its intensity inhomogeneities. The final result is represented by the union of the e_M final states.

4.2 Heart segmentation in CT

In this experiment, we used a public dataset†. The 3D-IRCADb-02 database is composed of two anonymized thoraco-abdominal enhanced 3D CT-scans. The first acquisition was made during the arterial phase in inhaled position.

The dataset provides the ground truth segmentation of the principal organs as DICOM images, and as triangular surface meshes as well. We segmented the heart using the ground truth meshes of the surrounding organs (lungs, stomach, liver and vena cava) as constraints or f_Ms. Figure 7 shows the initialization using the visualization module of the GUI belonging to the VTK libraries. We present three different runs. In the first one, we set the parameter configuration leading to a model expansion without using any image information (we nullified the gradient external force). The second uses all the standard forces but not the context information. Finally, the third run uses all of the available information.

*http://promise12.grand-challenge.org/
†http://www.ircad.fr/softwares/

Figure 7. Experiment initialization on CT: in green the initial e_M of the heart, in gray all the f_Ms.

5. RESULTS

5.1 Bladder and prostate in 2D-MRI

We considered ten images corresponding to different patients. This multi-object segmentation experiment exploits most of the capabilities of the framework. Firstly, we segmented the bladder and the prostate simultaneously. Both organs were considered as a complex structures employing two or three e_M using their own sets of parameters to overwhelm the image heterogeneities and complex shapes. We used $N = 25$ with a *fixed step checking* strategy for all the runs. Figure 8 presents the experiment qualitative results: on the left, the DMA results and on the right, the stand alone technique results. The spheric model initializations are depicted in yellow, the segments that did not collide in green, and the collided elements in red. On the one hand, the bladder external contours are mostly green, except for interaction with the prostate. On the other hand, when two or more e_M represent the structure of interest, their internal borders are completely red (e.g.: bladder P13a, P15a and P18a). In general, most of the results between DMA and the stand alone techniques are quite similar. Yet, intensity heterogeneities made P18 bladder particularly difficult to segment with one technique (18b Fig. 8). A similar situation occurs with the P21 prostate segmentation. The complete evolution lasts just a few seconds in each case.

			P13	P15	P18	P19	P21	P23	P27	P28	P30	P35	avg	Top 5 (Litjens 2014)
DMA Multiple	Bladder	$\mathcal{D}$	0,893	0,958	0,963	0,935	0,949	0,919	0,956	0,954	0,920	0,939	0,939	-
		$\mathcal{H}$	8,360	3,720	3,000	4,280	3,720	3,720	1,950	3,120	4,300	4,220	4,039	-
	Prostate	$\mathcal{D}$	0,883	0,912	0,956	0,850	0,827	0,827	0,891	0,921	0,907	0,886	**0,886**	0,890
		$\mathcal{H}$	3,790	2,480	4,590	3,520	3,970	3,970	4,570	3,170	9,120	3,140	**4,232**	5,300
Single no-collision	Bladder	$\mathcal{D}$	0,805	0,963	0,884	0,939	0,949	0,949	0,965	0,952	0,921	0,934	0,926	-
		$\mathcal{H}$	3,760	2,480	12,720	3,610	2,770	2,770	1,950	3,920	5,240	4,220	4,344	-
	Prostate	$\mathcal{D}$	0,881	0,823	0,954	0,839	0,757	0,757	0,823	0,925	0,902	0,877	0,854	0,890
		$\mathcal{H}$	15,200	5,330	4,570	5,300	7,490	7,490	8,670	3,150	9,120	3,780	7,010	5,300

Table 1. Multiple organ segmentation, quantitative comparison using $\mathcal{D}$ and $\mathcal{H}$ (mm) against GT. The DMA results in the upper part, the stand alone dm_T at the bottom.

Table 1 shows the Dice Metric and Hausdorff Distance of the segmentation results against the GT. The DMA achieved excellent results in this segmentation task: $\overline{\mathcal{D}}_{Bl} = 0.938$, $\overline{\mathcal{H}}_{Bl} = 4.47mm$ for the bladder and $\overline{\mathcal{D}}_{Pr} = 0.897$, $\overline{\mathcal{H}}_{Pr} = 4.95mm$ for the prostate while the standalone techniques gathered $\overline{\mathcal{D}}_{Bl} = 0.923$, $\overline{\mathcal{H}}_{Bl} = 5.16mm$ for the bladder and $\overline{\mathcal{D}}_{Pr} = 0.872$, $\overline{\mathcal{H}}_{Pr} = 7.39mm$ for the prostate. We compared and shown in Table 1 the results of the Prostate segmentation against the best results presented in[35] and they were almost equally.

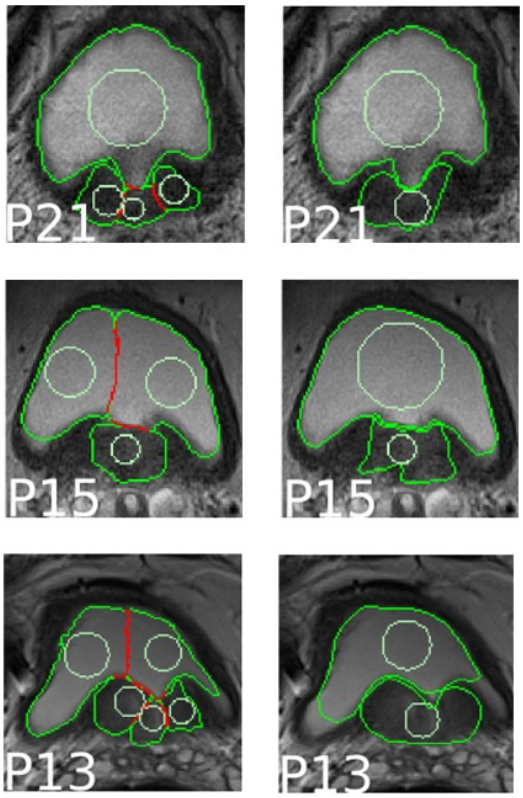

Figure 8. Bladder and prostate results for several patients: using the DMA framework (left) and using the standalone techniques (right). The model initializations are shown in yellow, the non-collided segments in green and the collided ones in red.

Information	$\mathcal{M}$	f_P	f_N
Context	0.765	0.202	0.053
Image	0.845	0.096	0.059
Both	0.892	0.068	0.040

Table 2. Quantitative comparison of the different information usage during segmentation.

5.2 Heart segmentation in CT

In this experiment, we did three different runs to assess the importance of the information sources. For each run we used a *sub-step checking* strategy with $N = 50$. In the first run we appreciate the importance of the context information. Just using the context information, we can achieve an acceptable segmentation: $\mathcal{M} = 0.765$ (Table 2, first row). As the e_M does not use image information, the DMA should over segment the heart since the e_M only stops when it collides with a f_M. This phenomenon is remarked in the f_p column of the Context experiment in Table 2. The f_p, which represents the over-segmentation phenomenon, comprehends the overall greatest error. In the second run, we performed a standard segmentation. Despite having a better overall segmentation metric: $\mathcal{M} = 0.845$ and half of the f_p volume compared to the first run, the e_M intersects boundary organs, especially the liver and the lungs. The final experiment used all of the available information. In this case, the e_M molded to the surrounding organs, and obtained the best result, $\mathcal{M} = 0.892$, with the lowest f_p and f_n volumes.

Each mesh has about 50 thousand triangles. The complete evolution lasts about one minute each. Using the sub-step strategy, time is about three times lower respect to the constant step strategy.

6. DISCUSSION AND CONCLUDING REMARKS

In this paper, we have described the extended version of *Deformable Models Array*, an object-oriented segmentation framework dedicated to complex and multiple structures segmentation using existing or new DM techniques

specially useful in the medical analysis area. A novel cooperative interaction scheme has been included, which was benefited by the context spatial information, generally improving the individual segmentation techniques in adjacent structure segmentation scenarios.

The DMA layered software design enabled to satisfy all its requirements. On the one hand, at the top layer, the specific GUI provides a user-friendly environment for easy segmentation prototyping. On the other hand, the bottom processing layer, based on the information hiding technique and design patterns, deals with the high computational workload, and the dimensionality problem, providing a simple way to wrap existing DM techniques and use them in the framework. The usage of open-source libraries makes it possible to install DMA on any personal computer.

In the current DMA version, employing an existing dm_T requires an implementation in `C++`, as a new class inherited from the *DM-technique* abstract class. This could be improved using a component-oriented method to combine different programming languages including Matlab or Python which are worldwide used to implement these kind of techniques.

The presented results demonstrate the performance of the framework applied to two real-case segmentation tasks. In the MRI bladder and prostate experiment, DMA not only achieved better segmentation results than the standalone technique but also did it in a parallel way. The contextual information along with the different DM techniques and parameter settings across the models, enhanced the results in the most problematic cases including intensity variations and blurred borders. The CT heart volumetric experiment showed in detail the benefits of using previous knowledge as spatial restrictions to increase the overall segmentation quality.

Summarizing, DMA is capable of: firstly, using any dm_T as a standalone technique; secondly, doing multiple-segmentations by using two or more models evolved by the same dm_T with different parameters or by employing the most suitable technique for each ROI; thirdly, performing complex-segmentations with the same variations of the multiple-segmentation; and finally, combining standalone, multiple and complex segmentation, all together, in a parallel way. On top of that, it is possible to work in both 2D and 3D scenarios, leading to an incredibly powerful and versatile segmentation framework.

Acknowledgments The authors would like to acknowledge the Prof. Maximiliano Cristiá for his contributions to the redaction of this paper and also the National Scientific and Technical Research Council (CONICET) and Buenos Aires Scientific Research council (CICPBA) from Argentina to support this research.

REFERENCES

[1] Sharma, N. and Aggarwal, L. M., "Automated medical image segmentation techniques," *Journal of medical physics/Association of Medical Physicists of India* **35**(1), 3 (2010).

[2] Okada, T., Linguraru, M. G., Hori, M., Summers, R. M., Tomiyama, N., and Sato, Y., "Abdominal multi-organ segmentation from ct images using conditional shapelocation and unsupervised intensity priors," *Medical Image Analysis* **26**(1), 1–18 (2015).

[3] Shimizu, A., Ohno, R., Ikegami, T., Kobatake, H., Nawano, S., and Smutek, D., "Segmentation of multiple organs in non-contrast 3d abdominal ct images," *International Journal of Computer Assisted Radiology and Surgery* **2**(3-4), 135–142 (2007).

[4] Okada, T., Linguraru, M. G., Hori, M., Suzuki, Y., Summers, R. M., Tomiyama, N., and Sato, Y., "Multi-organ segmentation in abdominal ct images," in [*Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*], 3986–3989, IEEE (2012).

[5] Balafar, M. A., Ramli, A. R., Saripan, M. I., and Mashohor, S., "Review of brain MRI image segmentation methods," *Artificial Intelligence Review* **33**(3), 261–274 (2010).

[6] Yang, H., Zhao, L., Tang, S., and Wang, Y., "Survey on brain tumor segmentation methods," in [*Medical Imaging Physics and Engineering (ICMIPE), 2013 IEEE International Conference on*], 140–145, IEEE (2013).

[7] Bay, T., Chambelland, J.-C., Raffin, R., Daniel, M., and Bellemare, M.-E., "Geometric modeling of pelvic organs," in [*Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*], 4329–4332, IEEE (2011).

[8] van Rikxoort, E. M., Isgum, I., Arzhaeva, Y., Staring, M., Klein, S., Viergever, M. A., Pluim, J. P., and van Ginneken, B., "Adaptive local multi-atlas segmentation: Application to the heart and the caudate nucleus," *Medical image analysis* **14**(1), 39–49 (2010).

[9] Ecabert, O., Peters, J., Walker, M. J., Ivanc, T., Lorenz, C., von Berg, J., Lessick, J., Vembar, M., and Weese, J., "Segmentation of the heart and great vessels in ct images using a model-based adaptation framework," *Medical image analysis* **15**(6), 863–876 (2011).

[10] Lu, L. and Zhao, J., "An improved method of automatic colon segmentation for virtual colon unfolding," *Computer methods and programs in biomedicine* **109**(1), 1–12 (2013).

[11] Namias, R., Bellemare, M.-E., Rahim, M., and Pirró, N., "Uterus segmentation in dynamic MRI using lbp texture descriptors," in [*SPIE Medical Imaging*], 90343W–90343W, International Society for Optics and Photonics (2014).

[12] Chui, C.-K., Wang, Z., Zhang, J., Ong, J. S.-K., Bian, L., Teo, J. C.-M., Yan, C.-H., Ong, S.-H., Wang, S.-C., Wong, H.-K., et al., "A component-oriented software toolkit for patient-specific finite element model generation," *Advances in Engineering Software* **40**(3), 184–192 (2009).

[13] Keeve, E., Jansen, T., von Rymon-Lipinski, B., Burgielski, Z., Hanssen, N., Ritter, L., and Lievin, M., "An open software framework for medical applications," in [*Surgery Simulation and Soft Tissue Modeling*], 302–310, Springer (2003).

[14] Dickinson, A. W., Abolmaesumi, P., Gobbi, D. G., and Mousavi, P., "Simitk: Visual programming of the itk image-processing library within simulink," *Journal of digital imaging* **27**(2), 220–230 (2014).

[15] Namías, R., D'Amato, J., del Fresno, M., Vénere, M., Pirró, N., and Bellemare, M., "Multiobject segmentation framework using deformable models for medical imaging analysis," *Medical and Biological Engineering and Computing* (2015).

[16] He, L., Peng, Z., Everding, B., Wang, X., Han, C., Weiss, K., and Wee, W., "A comparative study of deformable contour methods on medical image segmentation," *Image and Vision Computing* **26**(2), 141–163 (2008).

[17] Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: Active contour models," *International Journal of Computer Vision* **1**(4), 321–331 (1988).

[18] Malladi, R., Sethian, J. A., and Vemuri, B. C., "Shape modeling with front propagation: a level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**(2), 158–175 (1995).

[19] Abe, T. and Matsuzawa, Y., "A region extraction method using multiple active contour models," in [*Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*], **1**, 64–69, IEEE (2000).

[20] Chen, T. and Metaxas, D., "A hybrid framework for 3d medical image segmentation," *Medical Image Analysis* **9**(6), 547–565 (2005).

[21] del Fresno, M., Vénere, M., and Clausse, A., "A combined region growing and deformable model method for extraction of closed surfaces in 3d ct and mri scans," *Comput. Med. Imaging Graph* , 369–376 (2009).

[22] Shang, Y., Yang, X., Zhu, L., Deklerck, R., and Nyssen, E., "Region competition based active contour for medical object extraction," *Computerized Medical Imaging and Graphics* **32**(2), 109–117 (2008).

[23] Gong, Y.-Y., Luo, X.-N., Huang, H., Liao, G.-J., and Zhang, Y., "Multi-objects extracted based on single level set," *Jisuanji Xuebao/Chinese Journal of Computers* **30**(1), 120–128 (2007).

[24] Vu, D., Ha, T., Song, M., Kim, J., Choi, S., and Chaudhry, A., "Generalized chan-vese model for image segmentation with multiple regions," *Life Science Journal* **10**(1), 1889–1895 (2013).

[25] Chan, T. and Vese, L., "Active contours without edges," *IEEE Transactions on Image Processing* **10**(2), 266–277 (2001).

[26] Gao, Y., Kikinis, R., Bouix, S., Shenton, M., and Tannenbaum, A., "A 3d interactive multi-object segmentation tool using local robust statistics driven active contours," *Medical image analysis* **16**(6), 1216–1227 (2012).

[27] Blanchette, J. and Summerfield, M., [*C++ GUI programming with Qt 4*], Prentice Hall Professional (2006).

[28] Kim, Y., Koo, S. O., Lee, D., Kim, L., and Park, S., "Mesh-to-mesh collision detection by ray tracing for medical simulation with deformable bodies," in [*Cyberworlds (CW), 2010 International Conference on*], 60–66, IEEE (2010).

[29] Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W., et al., "Collision detection for deformable objects," in [*Computer Graphics Forum*], **24**, 61–81, Wiley Online Library (2005).

[30] McInerney, T. and Terzopoulos, D., "T-snakes: Topology adaptive snakes," *Medical Image Analysis* **4**(2), 73–91 (2000).

[31] Xu, C. and Prince, J. L., "Gradient vector flow: A new external force for snakes," in [*Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*], 66–71, IEEE (1997).

[32] Crum, W. R., Camara, O., and Hill, D. L., "Generalized overlap measures for evaluation and validation in medical image analysis," *Medical Imaging, IEEE Transactions on* **25**(11), 1451–1461 (2006).

[33] Klein, S., van der Heide, U., Lips, I.M.and van Vulpen, M. S. M., and Pluim, J., "Automatic segmentation of the prostate in 3d mr images by atlas matching using localized mutual information," *Med. Phys.* **35**, 1407–1417 (2008).

[34] D'Amato, J., del Fresno, M., Garcia Bauza, C., and Vénere, M., "Ray-casting method to assess the quality of segmented surfaces from 3d images," *Proc. of the 12th International Symposium on Medical Information Processing and Analysis* **31**(1060) (2016).

[35] Litjens, G., R., T., van de Ven W., C., H., S., K., van Ginneken B., G., V., et al., "Evaluation of prostate segmentation algorithms for mri: The promise12 challenge," *Medical Image Analysis* **18**(1), 359–373 (2014).