Image Segmentation Based on Active Contours without Edges

Anca Morar
Faculty of Automatic Control and Computers
University POLITEHNICA of Bucharest

Florica Moldoveanu
Faculty of Automatic Control and Computers
University POLITEHNICA of Bucharest

Eduard Gröller
Institute of Computer Graphics and Algorithms
Vienna University of Technology

Abstract—**There are a lot of image segmentation techniques that try to differentiate between background and object pixels, but many of them fail to discriminate between different objects that are close to each other. Some image characteristics like low contrast between background and foreground or inhomogeneity within the objects increase the difficulty of correctly segmenting images. We designed a new segmentation algorithm based on active contours without edges. We also used other image processing techniques such as nonlinear anisotropic diffusion and adaptive thresholding in order to overcome the images' problems stated above. Our algorithm was tested on very noisy images, and the results were compared to those obtained with known methods, like segmentation using active contours without edges and graph cuts. The new technique led to very good results, but the time complexity was a drawback. However, this drawback was significantly reduced with the use of graphical programming. Our segmentation method has been successfully integrated in a software application whose aim is to segment the bones from CT datasets, extract the femur and produce personalized prostheses in hip arthroplasty.**

Active contours without edges, image segmentation, nonlinear anisotropic diffusion, parallel image processing

I. INTRODUCTION

Image processing is a popular technique in many domains, like medicine, security and surveillance, traffic control and image editing. The human eye can easily distinguish important characteristics even in very poor quality images. The aim of image-processing software-applications is to interpret images in a similar (or even better) way as compared to a human being, but only faster. The results obtained by computers can be very impressive, for example, the 3D visualization of a CT dataset, with the highlighting of some important characteristics, like different human tissues. Often these automatic processing algorithms may lead to errors. The challenge in this domain is to discover fast and fully automatic image processing algorithms, or algorithms that require only little human intervention.

The research presented in this paper concentrates on the segmentation of poor quality images, like CT images, that have a granular aspect, small contrast between foreground and background and inhomogeneities regarding the intensity within the foreground. The second chapter discusses the state of the art in image segmentation. The third chapter describes a series of image processing techniques that were used in our algorithm and in the segmentation methods that were compared to our algorithm. Next, we state our motivation for designing a new segmentation technique. The fifth chapter presents the steps of our new algorithm. In the sixth chapter we focus on implementation details both on the CPU and on the GPU with CUDA. In the last chapter we present a comparison between the results obtained with our algorithm and the results produced by other segmentation algorithms.

II. RELATED WORK

There are a lot of image segmentation techniques, some based on intensity or texture, others on gradient or shape characteristics. Some of the methods that have proven to lead to good results in the segmentation of poor quality images are briefly presented in this section.

Kass et al. [1] introduce the concept of snakes, or active contours. Snakes are energy-minimizing splines guided by external constraint forces and influenced by image forces that pull them toward features such as lines and edges. Chan and Vese [2] propose active contours without edges. It is a new model for active contours, which is based on techniques of curve evolution, the Mumford-Shah functional for segmentation, and level sets. This method will be detailed in the next chapter.

Boykov and Veksler [3] describe the use of graph cuts in computer vision and graphics through theories and applications. In image segmentation, a graph is created from the image or the set of images. The graph construction and the characteristics that divide the pixels into two disjoint parts, i.e., the background and the foreground, will be detailed in the next section.

In grey scale mathematical morphology, the watershed transform, originally proposed by Digabel and Lantuejoul [4] and later improved by Buecher and Lantuejoul [5], is considered to lead to very good results in image segmentation. Roerdink and Meijster [6] wrote a review of several definitions and algorithms of the watershed transform. They describe in this review the geographic idea behind the watershed transform as that of a landscape or topographic relief which is flooded by water. Watersheds are the dividing lines of the domains of attraction of rain falling over the region. When the water level has reached the highest peak in the landscape, the process is stopped, and the result is a landscape partitioned into basins separated by dams, called watershed lines.

Porwik and Lisowska [7] present the use of the Haar-wavelet transform in digital image processing. Their paper describes a method of image analysis by means of the wavelet-

Haar spectrum. Glavasova et al. [8] discuss the wavelet transform for feature extraction, based on texture analysis, for the final goal of image segmentation.

There are a lot of other segmentation algorithms, but most of them are based to some extent on one of the techniques mentioned above. The challenges in image segmentation come from the problems previously stated. The small contrast between foreground and background makes it difficult to extract all the edges or lines that are used for example in active contours, graph cuts and watersheds. The inhomogeneities within the objects prove to be a drawback for active contours without edges, because this method tries to minimize the differences within the foreground and the background. The lack of a texture pattern could be a problem for the wavelet transform based segmentation. Our algorithm takes into account the imperfections of poor quality images (like CTs), not only differentiating between objects and background, but also between different objects.

III. Image Processing Techniques

For the understanding and motivation of our algorithm, we shortly describe in this section the existing techniques that we used in our image segmentation. We also present some details regarding the other segmentation methods that were compared to our algorithm.

A. Noise Reduction Using Gaussian Filtering

One of the most common types of noise is Gaussian noise [9]. Gaussian noise is modeled using a probability density function. In our implementation we used a convolution mask approximating a 2D Gaussian distribution function.

B. Nonlinear Anisotropic Diffusion

Nonlinear anisotropic diffusion [10, 11] reduces noise, but preserves the image information. We present in this sub-section some concepts of diffusion filtering. Diffusion is a physical process that balances the concentration changes of a certain substance. Having a concentration distribution u, Flick's law states that the concentration gradient determines a flow:

$$j = -D \cdot \nabla u \,. \tag{1}$$

D is the diffusion tensor, a positive-definite symmetric matrix. Diffusion represents mass transport without destroying or creating new mass. From the continuity equation we can derive the following equation:

$$\partial_t u = -div\,(j) = -\left(\frac{\partial j}{\partial x} + \frac{\partial j}{\partial y}\right), \tag{2}$$

where t denotes time, and $\partial_t u$, the deviation of u with respect to t. From (1) and (2) we can deduce the following result:

$$\partial_t u = -div\,(D \cdot \nabla u) \,. \tag{3}$$

In image processing the image intensity can be seen as "concentration", and image noise as concentration inhomogeneities. These inhomogeneities can be smoothed through diffusion. In order to preserve edges, the filtering will be accomplished in the following manner:

- no diffusion across edges
- diffusion parallel to the edges.

The diffusion tensor is a 2x2 matrix computed with the following expression:

$$D = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \cdot \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T \,. \tag{4}$$

The eigenvectors v_1 and v_2 provide the local diffusion orientations. Their corresponding eigenvalues give the contrast along these directions. The first eigenvector denotes the direction of maximum variation. Therefore, v_1 represents the direction parallel to the image gradient. The other eigenvector is determined considering the orthogonality property:

$$v_1 = \frac{\nabla u}{|\nabla u|} \text{ and } v_2 = \begin{bmatrix} v_{1y} \\ -v_{1x} \end{bmatrix}. \tag{5}$$

The eigenvalues are computed so that diffusion is inhibited across the edges, and activated parallel to the edges:

$$\lambda_1 = g\left(|\nabla u|^2\right) \text{ and } \lambda_2 = 1 \,. \tag{6}$$

The first eigenvalue is determined using the following expression proposed by Perona and Malik [12]:

$$g\left(|\nabla u|^2\right) = 1 - \exp\left(-C_m \Big/ \left(|\nabla u|^2 / \lambda\right)^m\right) \cdot \tag{7}$$

The constant C_m is computed so that the diffusion is big for $|\nabla u|^2 \in [0, \lambda]$ and small for $|\nabla u|^2 \in (\lambda, \infty)$. λ denotes the threshold for the gradient. Values bigger than the threshold determine edges. In our implementation we have chosen $\lambda = 4$ and $m = 4$. The value of m determines C_m=3.31488.

After computing the eigenvectors and the eigenvalues of the diffusion tensor $D = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}$, equation (3) is solved with finite differences. $\partial_t u$ can be replaced through a forward difference approximation.

The obtained explicit scheme allows the iterative computation of several versions of the image:

$$u(x,y,s) = u(x,y,s-1) + \Delta t \cdot [\frac{\partial}{\partial x}(D_{11}\frac{\partial u}{\partial x}) + \frac{\partial}{\partial x}(D_{12}\frac{\partial u}{\partial y}) + \frac{\partial}{\partial y}(D_{21}\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(D_{22}\frac{\partial u}{\partial y})] \quad , \tag{8}$$

where Δt denotes the time step size, and $u(x,y,s)$ represents the image at time $t_s = s \cdot \Delta t$. The standard scheme for the approximation of spatial derivatives is based on central differences. Fig. 1 presents the results of applying nonlinear anisotropic diffusion on a CT image.

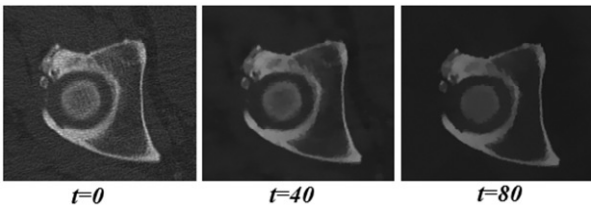

Figure 1. Nonlinear anisotropic diffusion filtering of a CT image at different moments in time (different number of iterations)

C. Graph Cuts

The mechanisms of graph cuts for image segmentation are given in more detail. This provides some insight into the graph construction algorithm used in the implementation that was compared to our algorithm. In a tutorial illustrating graph cuts in the context of computer vision and graphics, Boykov and Veksler [3] explain general theoretical properties that motivate their use.

Let $G = \langle V, E \rangle$ be a graph composed of a set of nodes V and a set of oriented edges E. The set of nodes $V = \{s,t\} \cup P$ contains two nodes called terminals, i.e., the source s and the sink t, as well as a set of non-terminal nodes P. Every edge in the graph is assigned a nonnegative weight $w(p,q)$. An edge is called t-link if it connects a non-terminal node with a terminal one. It is called n-link if it connects two non-terminal nodes. An s/t cut is a partitioning of the graph nodes into two disjoint sets S and T in a manner that the source s is in S and the sink t in T.

Starting from an image, the graph is built in a manner that every pixel in the image defines a non-terminal node of the graph. The cost of an n-link is based on the "likeliness" of the neighboring non-terminal nodes. The cost of a t-link is based on the "likeliness" of the connected non-terminal node and the terminal one. After the cut, some pixels belong to the source (being labeled as "object") and others to the sink ("background"). The minimal cut minimizes the energy function introduced by Boykov and Veksler [3]:

$$E(f) = \sum_{(p,q)\in Q} V_{p,q}(f_p, f_q) + \sum_{p\in P} D_p(f_p) \ , \tag{9}$$

where D_p is the per-pixel term that reflects the penalty of assigning the label f_p to pixel p. $V_{p,q}$ is the border term that encourages spatial coherence within the objects and the background. Q represents the set of all the n-links, f_p is the label assigned to pixel p ("object" or "background") and P denotes the set of all non-terminal nodes.

In order to define the border term $V_{p,q}$ we use the observation of Boykov and Veksler [3] that pixels with high image gradient would imply low cost of n-links and vice-versa. This is why we compute the gradient of the image using a Sobel filter. For smoother borders, we decided to apply the Sobel filter after convolving the image with a Gaussian filter and a nonlinear anisotropic diffusion filter. If the absolute difference between the gradient magnitude of two neighboring pixels p and q is greater than a given threshold k, $V_{p,q}$ is directly proportional with that difference. If there is a small variation of the gradient, the value of $V_{p,q}$, given by the constant v, is high. The border term can be defined as follows:

$$V_{p,q} = \begin{cases} \dfrac{1}{\left\|\nabla u_f(p)\right| - \left|\nabla u_f(q)\right\|}, if \left\|\nabla u_f(p)\right| - \left|\nabla u_f(q)\right\| > k \\ v, otherwise \end{cases} \tag{10}$$

where u_f is the image obtained after applying the Gaussian filter and the nonlinear anisotropic diffusion filter. This definition encourages borders in regions where there are abrupt variations of the gradient magnitude.

Boykov and Jolly [13] designed a technique for general purpose interactive segmentation of N-dimensional images. In their workflow, the user marks certain pixels as "object" or "background". We extend their approach by marking certain pixels based on their intensities. We introduce two thresholds, T_{low} and T_{high}, that determine the most probable background and foreground pixels. If the intensity of a pixel is greater than T_{high}, the pixel is most likely an object pixel. If its intensity is lower than T_{low}, there is a high probability that the pixel belongs to the background. If the pixel does not belong to any of the two categories, the per-pixel term depends on its intensity relative to T_{low} and T_{high}:

$$D_p(f_p) = \begin{cases} w, if\ f_p = "object"\ and\ u_0(p) > T_{high} \\ 0, if\ f_p = "background"\ and\ u_0(p) > T_{high} \\ w, if\ f_p = "background"\ and\ u_0(p) < T_{low} \\ 0, if\ f_p = "object"\ and\ u_0(p) < T_{low} \\ w \cdot \dfrac{u(p) - T_{low}}{T_{high} - T_{low}}, if\ f_p = "object" \\ \qquad\qquad and\ T_{low} \le u_0(p) \le T_{high} \\ w - w \cdot \dfrac{u(p) - T_{low}}{T_{high} - T_{low}}, if\ f_p = "background" \\ \qquad\qquad and\ T_{low} \le u_0(p) \le T_{high} \end{cases} \ . \tag{11}$$

where u_0 denotes the initial image. The constant w assures a high cost for t-links that connect a non-terminal node and a terminal one that have the same label (either "object" or "background").

Our algorithm was compared to two segmentation methods based on graph cuts. In the first implementation, the set of n-links Q contains only 4-connected neighbors. The second implementation can be used only for volumetric data, because it considers also connections of pixels from adjacent slices. The first method is further called 2D graph cuts (2D GC), because it segments the images individually. The second one is being referred to as 3D graph cuts (3D GC), because it can be applied only to 3D images (like CT datasets).

D. Active Contours without Edges

This section presents the active contours model without edges. Part of the method is included in the workflow of our algorithm. This technique is also used as a comparison to our segmentation. In their paper, Chan and Vese [2] propose a new active contour model for object detection in a 2D image. The stopping term for the involved curve evolution process does not depend on the image gradient, but is related to a particular segmentation of the image.

An evolving curve C, in the image space Ω, can be defined as the frontier of a subset ω of Ω ($\omega \subseteq \Omega$ and $C=\partial\,\omega$). ω represents the region occupied by foreground pixels. *inside*(C) denotes the region ω and *outside*(C) denotes the region $\Omega \setminus \overline{\omega}$. The image u_0 is assumed to be composed of two regions of approximately constant intensities c_1 (the intensity of the object) and c_2 (the intensity of the background). If the object's boundary is C, then inside of C the intensity value should be equal to c_1. Outside of C, the intensity value should be equal to c_2. Chan and Vese [2] introduce the following energy:

$$F(c_1, c_2, C) = \mu \cdot Length\,(C) + \nu \cdot Area\,(inside\,(C)) + \eta_1 \int_{inside\,(C)} |u_0(x,y) - c_1|^2\,dxdy + \eta_2 \int_{outside\,(C)} |u_0(x,y) - c_2|^2\,dxdy \tag{12}$$

where $\mu \geq 0$, $\nu \geq 0$, $\eta_1, \eta_2 > 0$ are fixed parameters. The length of the curve, *Length*(C), and the area of the region inside C, *Area*(*inside*(C)), are two regularizing terms. Chan and Vese [2] set $\nu = 0$, $\eta_1 = 1$ and $\eta_2 = 1$. We keep the values of ν, η_1 and η_2 defined by them, but we also set μ to 0. We explain in the fifth chapter the reason for omitting the length term. The image segmentation into foreground and background is accomplished by solving the minimization problem $\inf_{c_1, c_2, C} F(c_1, c_2, C)$. Let $C \subset \Omega$ be defined as the zero level set of a Lipschitz function $\phi : \Omega \to R$, so that:

$$\begin{cases} C = \partial\omega = \{(x,y) \in \Omega : \phi(x.y) = 0\} \\ inside\,(C) = \omega = \{(x,y) \in \Omega : \phi(x.y) > 0\} \\ outside\,(C) = \Omega \setminus \overline{\omega} = \{(x,y) \in \Omega : \phi(x.y) < 0\} \end{cases}.$$

Using the Heaviside function H and the one-dimensional Dirac measure δ_0 defined by Chan and Vese [2], the energy $F(c_1, c_2, C) = F(c_1, c_2, \phi)$ can be expressed as follows:

$$F(c_1, c_2, C) = \int_\Omega |u_0(x,y) - c_1|^2 H(\phi(x,y))\,dxdy + \int_\Omega |u_0(x,y) - c_2|^2 (1 - H(\phi(x,y)))\,dxdy \tag{13}$$

The constants c_1 and c_2 can be expressed relative to ϕ:

$$c_1(\phi) = \frac{\int_\Omega u_0 H(\phi(x,y))\,dxdy}{\int_\Omega H(\phi(x,y))\,dxdy} \quad \text{and} \tag{14}$$

$$c_2(\phi) = \frac{\int_\Omega u_0 (1 - H(\phi(x,y)))\,dxdy}{\int_\Omega (1 - H(\phi(x,y)))\,dxdy}. \tag{15}$$

The evolution of ϕ can be parametrized as follows:

$$\frac{\partial \phi}{\partial t} = \delta_0(\phi)[(u_0 - c_1)^2 + (u_0 - c_2)^2] = 0 \cdot \tag{16}$$

The segmentation algorithm follows an iterative method. Knowing $\phi(x,y,s)$ at time $t_s = s \cdot \Delta t$, $c_1(\phi, s)$ and $c_2(\phi, s)$ can be computed by using (14) and (15). Then, $\phi(x,y,s+1)$ can be computed by the following discretization and linearization of (16) in ϕ:

$$\phi(x,y,s+1) = \phi(x,y,s) + \Delta t \cdot \delta_0(\phi(x,y,s)) \cdot [-(u_0(x,y) - c_1(s))^2 + (u_0(x,y) - c_2(s))^2] \tag{17}$$

where Δt denotes the time step size and $u_0(x,y)$ represents the initial image.

IV. MOTIVATION

In this chapter we give the reasons for designing a new segmentation method. The algorithms described in the previous chapter were tested on ten noisy CT datasets. The aim of the segmentations was to differentiate between bones and other tissues, but also to discriminate between different bones. Fig. 2 presents an image before (a) and after being processed with 2D graph cuts (c), 3D graph cuts (d) and active contours without edges (e). Compared to the manual segmentation (b), the results show that, even if the most pixels are correctly labeled as "object" or "background", there are some problems with differentiating between bones.

As previously mentioned, our segmentation is part of a bigger project, whose aim is to obtain prototypes for personalized implants in hip arthroplasty. Since one of the requirements of automatically producing an artificial implant is

to extract the femoral bone and to differentiate it from other bones, we decided to implement a new segmentation technique. Our algorithm segments the objects from the background quite well, without connecting different objects. In the example provided in Fig. 2, it can be observed that our segmentation (f) was the closest to the output of the manual segmentation. In chapter seven we present the results of a comparison with the other segmentation algorithms on ten datasets, in order to demonstrate the quality of our algorithm in more detail.

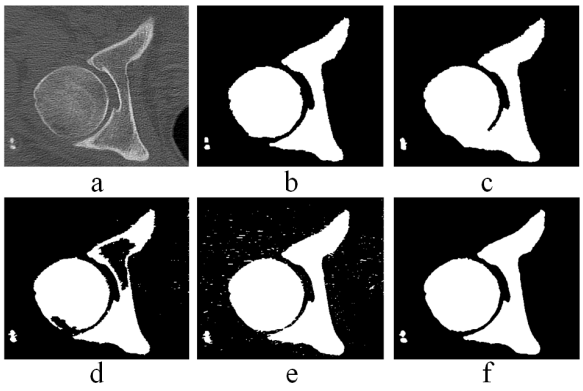

Figure 2. Comparison of different CT image segmentation methods: (a) original image, (b) image segmented by a human specialist, (c) image segmented with 2D graph cuts, (d) image segmented with 3D graph cuts, (e) image segmented with active contours without edges, (f) image segmented with our algorithm

The algorithm has been exemplified using CT images, but it can be applied on all kinds of grayscale images that have the following characteristics:

- The foreground has a higher intensity than the background
- The images contain multiple objects that are positioned close to each other and should not be connected
- There are big inhomogeneities within the foreground.

V. SEGMENTATION ALGORITHM

This chapter presents our new segmentation technique. The steps of the algorithm are identified in the workflow of Fig. 3 and are described in detail in the following subsections.

Step 1: Gaussian and Active Contours (G + AC)

As mentioned before, our segmentation is based on the active contours model by Chan and Vese [2]. Here we state our reasons for making some changes to the original algorithm. The first change was the omission of the length term in equation (12). We tested the original algorithm on images where two different objects were positioned very close to each other. The output image contained a single object, composed of the two initial objects. An explanation could be the one given by Chan and Vese [2] regarding the use of the length term as a scale parameter. If the constant μ from (12) is small, then also smaller objects will be detected. If it is larger, then only larger objects are detected, or objects that are grouped together. We do not want different objects close to each other to be interpreted as a single object. This is the reason we decided to set $\mu = 0$ in (12), and to use (17) for the computation of ϕ.

Most of the tested images had a lot of noise. In order to remove this noise we smooth the initial image u_0 with a Gaussian filter. The image segmentation that divides the pixels based on the value of ϕ relative to 0 is too rough in the sense that it leads to a binary image (the white pixels belong to the foreground and the black pixels belong to the background). The original active contours approach without edges also requires a large number of iterations in order to get to a stable configuration (the final segmentation). An alternative to converting the image into a binary segmented one could be a segmentation where the active contours model represents only an enhancement step.

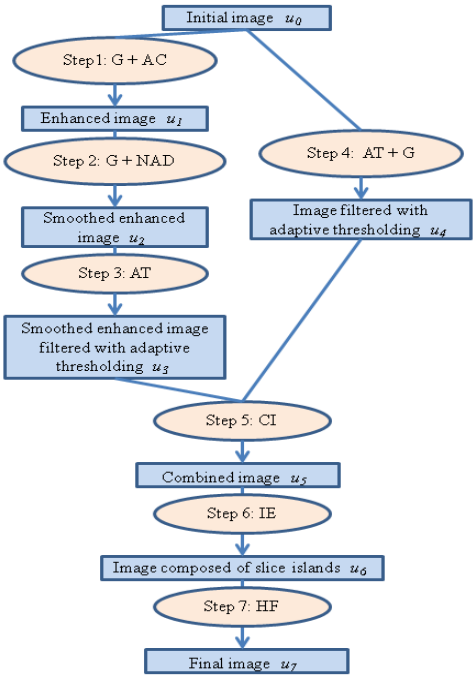

Figure 3. Image segmentation workflow

After computing the values of ϕ based on (17), in the last iteration we normalize the values of ϕ to [-1,1]. In the original active contours approach without edges, if $\phi < 0$, then the segmented image $u_1(x, y) = 0$ and if $\phi \geq 0$, $u_1(x, y) = I_{\max}$. The biggest change to the original algorithm is in computing the output image by normalizing ϕ to $[0, I_{\max}]$, where $I_{\max}$ is the maximum level of intensity (255 in our case):

$$u_1(x, y) = \frac{(1 + \phi(x, y)) \cdot I_{\max}}{2}. \quad (18)$$

With our approach, the output image u_1 will have shades of gray, with an enhanced contrast between foreground and background. Fig. 4(b) presents a CT image after applying the first step from our algorithm.

Step 2: Gaussian and Nonlinear Anisotropic Diffusion (G+NAD)

For the removal of small discontinuities within the object (and especially near the borders) we apply a Gaussian filter on the image u_1. We also apply a nonlinear anisotropic diffusion filter, considering the image gradient, in order to avoid connecting different objects. Fig. 4(c) presents the output image u_2 after applying these two smoothing filters.

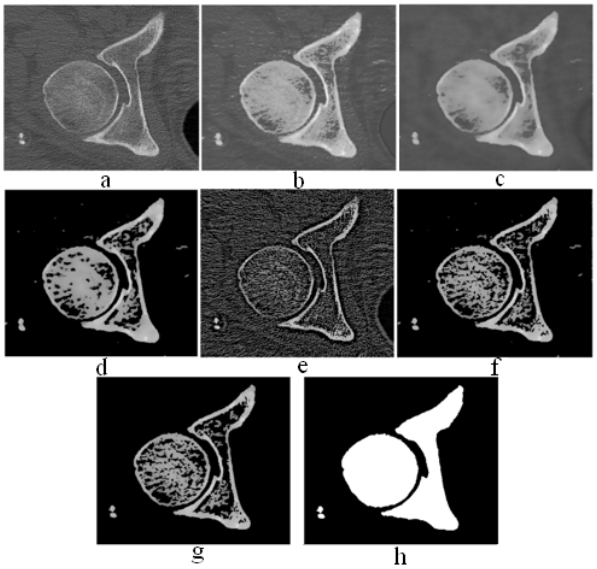

Figure 4. Output images in the image segmentation steps: (a) original image, (b) image enhanced with active contours without edges (Step 1), (c) image smoothed with nonlinear anisotropic diffusion (Step 2), (d) result of adaptive thresholding (Step 3), (e) output image after applying an adaptive threshold on the original image (Step 4), (f) image obtained by combining the images u_3 and u_4 (Step 5), (g) image resulting from connecting the foregorund pixels into slice islands and removing those islands with a low intensity (Step 6), (h) final image after applying the hole filling step (Step 7)

Step 3: Adaptive thresholding (AT)

This sub-section states the reasons for introducing another processing step, an adaptive thresholding. The intensities of the image u_2 are between 0 and I_{max}. A simple segmentation would be to separate the pixels based on their intensity relative to $I_{max}/2$. But there are two problems with this segmentation:

- Some pixels with intensity close to the threshold $I_{max}/2$ but lower than $I_{max}/2$ should belong to the foreground. We set the interval to search for these pixels to $[I_{max}/2 - \Delta I, I_{max}/2]$, where ΔI is a fixed parameter.
- There are some pixels whose intensities are close to the threshold $I_{max}/2$ but greater than $I_{max}/2$ which should belong to the background. We set the interval to search for these pixels to $[I_{max}/2, I_{max}/2 + \Delta I]$.

In order to eliminate the above problems, we use two thresholds $T_1 = I_{max}/2 - \Delta I$ (low) and $T_2 = I_{max}/2 + \Delta I$ (high). All the pixels with intensity value lower than T_1 are labeled as background pixels (the intensity is set to 0). All the pixels with intensity value greater than T_2 are considered foreground pixels (their intensity remains unchanged). For the pixels that belong to the interval $[T_1, T_2]$ we apply an adaptive threshold. This filter removes pixels that could wrongly connect different objects. Choosing a window of neighbors W_p of size nxn for the current pixel p we compute the average intensity, multiplied by a fixed parameter α:

$$M = \frac{\sum_{q \in W_p} u_2(q)}{n^2} \cdot \alpha \cdot \tag{19}$$

M is a threshold for dividing the pixels into foreground and background. If $u_2(p) < M$, then p is labeled as background pixel. If $\alpha = 1$, there are still some pixels wrongly considered as part of the foreground. This is why we set a slightly higher threshold with $\alpha = 1 + \Delta\alpha$, where $\Delta\alpha > 0$ is very close to 0. Fig. 4(d) shows the result u_3 of applying the adaptive thresholding to image u_2.

Step 4: Adaptive thresholding and Gaussian (AT+G)

Due to Step 3, there are fewer wrongly labeled pixels. However, in case of objects that are very close to each other, or small contrast between foreground and background, there could still be pixels that connect different objects. For this problem we decided to apply another adaptive threshold, but this time, on the initial image u_0. We also smooth the output with a Gaussian filter. This adaptive threshold does not use T_1 and T_2 but is applied to all the image pixels. Applying the second adaptive threshold assures the removal of the pixels that could have been wrongly labeled as "object" because of the smoothing steps (the two Gaussian filtering and the nonlinear anisotropic diffusion). Fig. 4(e) presents the output image u_4 after applying the adaptive threshold on the initial image u_0.

Step 5: Combining images (CI)

The next step consists of combining the images u_3 and u_4 in order to obtain an image with very few or no pixels that connect different objects. For the current pixel p, if $u_3(p)>0$ and $u_4(p)<T_3$ (where T_3 is an experimentally determined threshold), then the combined image $u_5(p)=0$, else $u_5(p)=u_3(p)$. Fig. 4(f) shows the output image u_5 after combining images u_3 and u_4.

Step 6: Island extraction (IE)

The low value of the threshold T_2 from Step 3 removes small discontinuities within the objects. However, this also adds wrongly labeled pixels. We treat this problem in the following manner: the foreground pixels that are connected are grouped into islands. For every island, the average intensity I_{avg} is computed. If $I_{avg}<T_4$, where T_4 is a fixed parameter, then the current island is considered to belong to the background. This way we are assured that only those low intensity pixels which are connected to a large number of high intensity pixels are considered to belong to the foreground. Fig. 4(g) presents the output image u_6 after removing the low intensity islands from image u_5.

Step 7: Hole filling (HF)

The last step in the segmentation process solves the problem of inhomogeneities within the object. Starting from the first black pixel in the image (from upper left to lower

right) we do a breadth-first-search (BFS) in order to visit all the background pixels connected to the first one. All the other pixels that have not been visited in the BFS are considered to be part of the foreground, as in Fig. 4(h). This step is called hole filling because it re-labels all the background pixels that are located inside a closed foreground island. The problem of this step is in not differentiating between inhomogeneities within an object and real holes. This drawback can be overcome in the following way: if we can be sure that a pixel belongs to a real hole, and not an inhomogeneity within the object, this pixel can be set as another seed point for the BFS that searches for all the connected background pixels.

VI. IMPLEMENTATION

The segmentation algorithm was implemented both on the CPU and on the GPU. Even for relatively small datasets, the CPU implementation takes a lot of time. On the other hand, the CUDA architecture [14] provides the possibility of running the same instructions for each pixel in a parallel manner, decreasing the computing times considerably. Based on the implementation of the Sobel filter from the CUDA SDK example [15] and the paper by Bojsen-Hansen [16], we parallelized almost all the steps in our algorithm. Table 1 shows all the processing steps that were implemented in CUDA. It also provides a comparison between the running times on a dataset of 256 images (each of size 512^2) for the implementation on the CPU and on the GPU. We set the size of the Gaussian filters to 9x9 and the size of the neighborhood window for the adaptive thresholds to 21x21. The maximum number of iterations for active contours without edges is 50 and the maximum number of iterations for nonlinear anisotropic diffusion is 20. The tests were made on an i7-2600K 3.40 GHz processor with 8GB RAM and an Nvidia GeForce GTX 590 GPU card with 1.5 GB RAM. The island extraction and the hole filling step were not implemented on the GPU, because they are based on recursion (BFS) and cannot be intuitively approached in a parallel manner.

TABLE I. COMPUTING TIMES FOR OUR ALGORITHM ON THE CPU AND THE GPU

Step in the algorithm	Time on CPU (sec)	Time on GPU (sec)
Step 1: G + AC	2245.71	4.02
Step 2: G + NAD	340.19	2.85
Step 3: AT	11.78	0.39
Step 4: AT + G	205.98	1.21
Step 5: CI	0.14	0.1
Steps 1-5	2803.8	8.79

The CUDA implementation has a great impact on the speed of the segmentation process. Thus, with the help of the GPGPU paradigm we now have a quite fast algorithm. In the next chapter we will see how accurate this algorithm is.

VII. RESULTS

Our segmentation algorithm was tested on ten CT datasets with different number of slices, each of size 512x512. The parameters from the fifth chapter were the same for all the images: $\Delta I = 30 \Rightarrow T_1 = 97.5$ and $T_2 = 157.5$; $\Delta\alpha = 1/29 \Rightarrow \alpha \approx 1.0345$; $T_3 = 50$ and T_4=110. The initial curve C is a circle located in the center of the image with a radius of 100: $\phi = -\sqrt{(x-256)^2 + (y-256)^2} + 100$.

Our implementation was compared with three other segmentation methods:

- Active contours without edges, described in subsection III.D, with the following parameters: the maximum number of iterations for computing ϕ is 100, and the parameters from (12) are $\eta_1 = 1$, $\eta_2 = 1$, $\mu = 0.2 \cdot 255^2$ and $\nu = 0.01 \cdot 255^2$.
- The segmentations using 2D and 3D graph cuts, described in subsection III.C, with the following parameters: $k = 3$, $v = 5$ and $w = 3$.

Depending on the difficulty of correctly labeling the foreground (bones) and the background pixels (other tissues), the CT datasets were divided into two categories: low and high difficulty. The images segmented by the four algorithms were compared with the segmentation made by a human specialist. The tests consisted of counting the correctly labeled pixels, the foreground pixels that were wrongly labeled as background pixels (false negatives), the background pixels that were wrongly labeled as foreground pixels (false positives), and the number of images where two different objects were wrongly connected. If F_1 is the number of correctly labeled foreground pixels, and F_2 is the number of false negatives, then the false negative percentage is $E_F = F_2 / (F_1 + F_2) \cdot 100$. Similarly, the false positive percentage is $B_F = B_2 / (B_1 + B_2) \cdot 100$, where B_1 is the number of correctly labeled background pixels and B_2 is the number of false positives. Table 2 presents the false positive and the false negative error for the low difficulty datasets.

TABLE II. ERRORS IN LABELING PIXELS FOR THE LOW DIFFICULTY DATASETS (%)

Alg.	*Data1*		*Data2*		*Data3*		*Data4*		*Data5*	
	B_F	E_F	B_F	E_F	B_F	E_F	B_F	E_F	B_F	E_F
Our alg.	0.028	0.691	0.068	0.651	0.56	0.897	0.247	1.214	0.922	1.342
2D GC	0.015	0.497	0.0337	0.676	13.724	0.677	23.905	0.663	0.258	1.584
3D GC	0.068	0.772	0.11	0.811	4.564	1.525	12.029	11.332	0.927	1.116
AC	0.015	0.585	0.027	0.814	0.861	0.814	0.128	0.434	0.859	0.818

Table 3 presents the percentage of images where two different objects were wrongly connected, for the low difficulty datasets.

TABLE III. PERCENTAGE OF IMAGES WHERE DIFFERENT OBJECTS WERE WRONGLY CONNECTED FOR THE LOW DIFFICULTY DATASETS

Algorithm	Percentage of images (%)				
	Data1	*Data2*	*Data3*	*Data4*	*Data5*
Our algorithm	0	0	0	10.42	0
2D graph cuts	0	0	4.16	14.58	34.09
3D graph cuts	0	0	8.33	29.16	13.63
Active contours	0	0	8.33	18.75	21.59

The high difficulty datasets have more noise, more inhomogeneities within the objects, and a bigger change in intensity between different images of the same dataset. Table 4 presents a comparison regarding the false positive and false negative errors between our algorithm and the other segmentation algorithms, for the high difficulty datasets.

TABLE IV. ERRORS IN LABELING PIXELS FOR THE HIGH DIFFICULTY DATASETS (%)

Alg.	Data6		Data7		Data8		Data9		Data10	
	B_F	E_F	B_F	E_F	B_F	E_F	B_F	E_F	B_F	E_F
Our alg.	0.999	1.163	1.441	1.342	2.79	0.369	6.331	1.569	6.83	1.76
2D GC	4.59	1.219	0.361	2.629	2.924	0.847	4.573	21.75	8.276	6.653
3D GC	2.396	1.689	1.147	1.631	3.404	0.505	10.774	8.96	23.748	3.138
AC	0.752	2.037	0.45	1.725	3.533	0.405	2.977	1.937	6.2	4.275

Table 5 presents the comparison between the segmentation algorithms regarding the percentage of images were different objects are wrongly connected, for the high difficulty datasets.

TABLE V. PERCENTAGE OF IMAGES WHERE DIFFERENT OBJECTS WERE WRONGLY CONNECTED FOR THE HIGH DIFFICULTY DATASETS

Algorithm	Percentage of images (%)				
	Data6	*Data7*	*Data8*	*Data9*	*Data10*
Our algorithm	0	0	0	21.73	8.69
2D graph cuts	10	100	10.46	69.56	86.95
3D graph cuts	30	30	6.58	73.91	30.43
Active contours	60	90	10.85	100	91.30

The results in computing the pixel labeling error for our algorithm were comparable to or even better than the other algorithms' results. The big difference can be observed in the percentage of images where different objects were wrongly connected. Our algorithm discriminated quite well between different objects, even on noisy images. This does not hold for the other algorithms. In order to obtain an overview of the differences between the four segmentation algorithms, we have computed, for all the datasets, the average of the false negative and false positive error in labeling the pixels, and the average percentage of images where different objects were wrongly connected. These differences can be seen in Fig. 5.

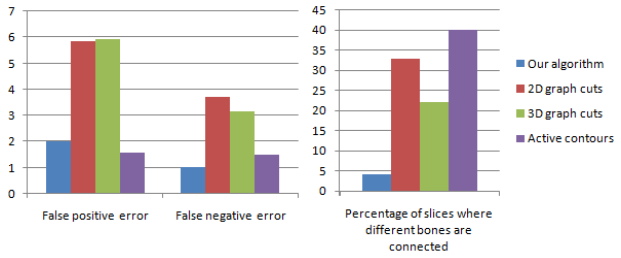

Figure 5. Comparison between our algorithm and other segmentation algorithms regarding the average of the false positive and false negative error in pixel labeling and the average percentage of images where different objects were wrongly connected

Even if the average error in pixel labeling from the active contours without edges segmentation is comparable to the result obtained with our algorithm, the average percentage of images where different objects are wrongly connected shows that our implementation is superior. From the tests described in the previous section and from Fig. 5 we can draw the conclusion that our algorithm is 98% and 99% accurate regarding the labeling of background and foreground pixels, respectively, and 96% accurate in discriminating between different objects in CT images. Also, we can observe that in seven cases out of ten, our algorithm differentiated perfectly between objects. The percentage of slices where different objects were wrongly connected is 0% in these cases. The next step in our research is to connect segmented CT images in order to obtain the whole volume occupied by bones. Also, we want to reconstruct the interior of the femoral bone, for the purpose of obtaining the 3D model of a personalized implant.

REFERENCES

[1] M. Kass, A. Witkin, D. Terzopoulos, "Snakes: active contour models", in International Journal of Computer Vision, 1988, pp. 321-331.

[2] T.F. Chan, L.A. Vese, "Active contours without edges", in IEEE Transactions on Image Processing, 2001, Issue 2, pp. 266-277.

[3] Y. Boykov, O. Veksler, "Graph cuts in vision and graphics: theories and applications", in Math. Models of C. Vision: The Handbook, Springer, 2006.

[4] H. Digabel, C. Lantuejoul, "Iterative Algorithms", in Actes du Second Symposium Europeen d'Analyse Qantitative des Microstructures en Sciences des Materiaux, Biologie et Medicine, 1978, pp. 85-99.

[5] S. Beucher, C. Lantuejoul, "Use of watersheds in contour detection", in Proc. International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation, 1979.

[6] J. B. T. M. Roerdink, A. Meijsetr, "The watershed transform: definitions, algorithms and parallelization strategies", in Fundamenta Informaticae, 2000,Issue 41(1-2), pp. 187-228.

[7] P. Porwik, A. Lisowska, "The Haar-wavelet transform in digital image processing: its status and achievements", in Machine Graphics & Vision, 2004, vol. 13, no. 1/2, pp. 79-98.

[8] A. Gavlasova, A. Prochazka, M. Mudrova, "Wavelet based image segmentation", in Proc. of the 14th Annual Conference Techincal Computing, Prague, 2006.

[9] R.C. Gonzales, R.E. Woods, Digital Image Processing, Prentice-Hall, 2002, pp. 222-224.

[10] J. Weickert, Anisotropic Diffusion in Image Processing, B.G. Teubner Stuttgart, 1998.

[11] S. Tabik, E.M. Garzon, I. Garcia, J.J. Fernandez, "Implementation of anisotropic nonlinear diffusion for filtering 3D images in structural biology on SMP clusters", in Parallel Computing: Curent & Future Issues of High-End Computing, Proceedings of the International Conference ParCo, 2005, Vol 33, pp. 727-734.

[12] P. Perona, J. Malik, "Scale-space and edge detection using anisotropic diffusion", in Proceedings of IEEE Computer Society Workshop on Computer Vision, 1987, pp. 16-22.

[13] Y. Boykov, M. P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images", in Computer Vision 2001, ICCV 2001, Proceedings, Eighth IEEE International Conference on Computer Vision, pp. 105-112, 2001

[14] Nvidia Corporation, NVIDIA CUDA C Programming Guide, Version 4.0, 2011, available at www.nvidia.com last visited in 18.05.2012.

[15] Nvidia Corporation, CUDA SDK Example, available at www.nvidia.com last visited in 18.05.2012.

[16] M. Bojsen-Hansen, "Active contours without edges on the GPU", in Project Paper for the Course in Parallel Computing for Medical Imaging and Simulation, 2010.