Three-dimensional segmentation of bones from CT and MRI using fast level sets

Jakub Krátký and Jan Kybic

Center for Machine perception, Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic

ABSTRACT

Our task is to segment bones from 3D CT and MRI images. The main application is creation of 3D mesh models for finite element modeling. These surface and volume vector models can be used for further biomechanical processing and analysis. We selected a novel fast level set method because of its high computational efficiency, while preserving all advantages of traditional level set methods. Unlike in traditional level set methods, we are not solving partial differential equations (PDEs). Instead, the contours are represeted by two sets of points, corresponding to the inner and outer edge of the object boundary. We have extended the original implementation in 3D, where the speed advantage over classical level set segmentation are even more pronounced. We can segment a CT image of $512 \times 512 \times 125$ in less than 20 s by this method. It is approximately two orders of magnitude faster than standard narrow band algorithms. Our experiments with real 3D CT and MRI images presented in this paper showed high ability of the fast level set algorithm to solve complex segmentation problems.

Keywords: Segmentation, image processing, level sets, CT, MRI

1. INTRODUCTION

Problems with bones or joints, such as osteoarthrosis, aseptic necrosis, prosthesis implantation or temporomandibular disease are very frequent. Biomechanical models of bones and joints are used for diagnostics or to design prothetic replacements tailored for a specific patient. Biomechanical modeling is often done using a finite element method,[1–3] which requires a volumetric mesh of the structure being modeled, which in turn requires the structure to be segmented from the 3D CT or MRI images. The standard approach is to segment the structures manually or semi-manually, which is very tedious and time consuming. The purpose of our work is therefore to present a fast and automatic method for segmentation of bones and joints from 3D CT and MRI images.

There are many segmentation methods, such as thresholding, which segments an image using an intensity threshold, edge-based algorithms, e.g. gradient or zero-crossing methods, or a region based segmentation, that includes region growing techniques, hierarchical image splitting or watershed segmentation. We have focused on level set techniques[4–6] because they can accommodate various shape and image based segmentation cost criteria and is easily extended into 3D.

2. METHOD

The level set method is an iterative segmentation technique for tracking interfaces and shapes first proposed by Osher and Sethian.[4] Let Γ be an interface dividing the image foreground and background. The interface Γ defines a signed distance function φ, such that the value of φ is the distance to the boundary Γ at every point, with positive sign outside and negative inside. At the boundary we have $\varphi = 0$.

We are given a velocity field v, which describes the speed of Γ in the normal direction. The velocity can depend on position, time, the geometry of the interface (e.g. its curvature), and on image data.[5] The interface Γ moves according to the velocity field v by means of changing the level set φ.

The evolution of the level set φ is described by a partial differential equation (PDE)[4–6] in a continuous domain:

$$\frac{\partial \varphi}{\partial t} + v \cdot |\nabla \varphi| = 0 \tag{1}$$

E-mail: kratkj3@fel.cvut.cz, kybic@fel.cvut.cz

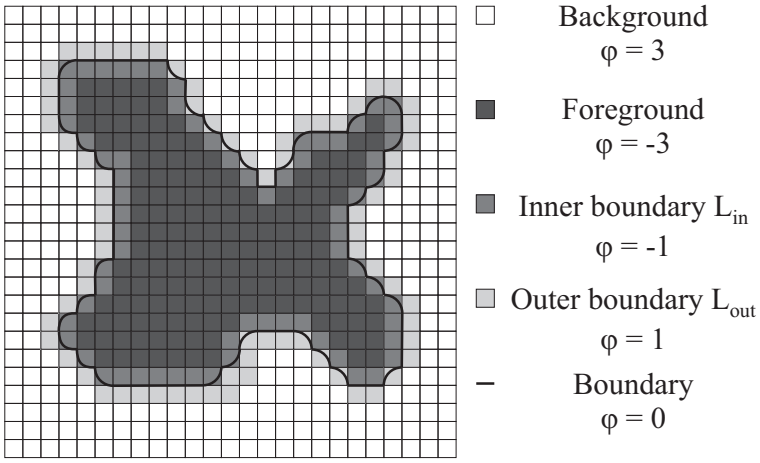


Figure 1. The implicit representation of a boundary Γ in the fast level set method. Each pixel is assigned a level set function value φ. Two sets of neighboring grid points L_{in} and L_{out} are defined. The motion of the implicitly represented curve is represent by maintaining the point sets L_{in} and L_{out}.

Solving this PDE can be computationally expensive, because the PDE has to be integrated over the whole image domain (2D or 3D) until convergence. In this work we have chosen to use a fast level set algorithm,[7] which works in the discrete domain, avoids solving the PDE and is therefore much faster.

2.1 Fast Level Set Method

We use a fast level set based algorithm developed by Shi and Karl[7] which we shall recall here briefly. It is very fast while it preserves all of the advantages of traditional level set methods. The basic idea of this algorithm is to represent the zero level set as a list of boundary points and move this discrete boundary on a discrete pixel grid without computing any PDEs. The only information we treat is where to move the boundary points, which is determined from the velocity field value at the particular pixel. The second main difference from traditional level set approaches is that the velocity field represents only the external force and the regularization is realized by smoothing as a separate step of the algorithm.

We assume an image domain $\mathcal{I} \subset R^N$, which is discretized on a uniformly sampled grid with sampling interval one.

The evolving boundary Γ between the foreground Ω and background $\Omega_b = \mathcal{I} \setminus \Omega$ is represented implicitly as the zero level set of a function φ defined over the fixed grid $\mathcal{I}$. We choose φ to be negative inside Ω and positive outside Ω. The interface $\Gamma = \partial\Omega$ is represented by two neighboring sets of grid points: L_{in} corresponds to the inner boundary of Γ and L_{out} corresponds to the outer boundary of Γ (see Figure 1). More formally:

$$\begin{aligned} L_{in} &= \{\mathbf{x}; \varphi(\mathbf{x}) < 0 \text{ and } \exists \mathbf{y} \in N(\mathbf{x}) \text{ such that } \phi(\mathbf{y}) > 0\} \\ L_{out} &= \{\mathbf{x}; \varphi(\mathbf{x}) > 0 \text{ and } \exists \mathbf{y} \in N(\mathbf{x}) \text{ such that } \phi(\mathbf{y}) < 0\} \end{aligned} \tag{2}$$

where $N(\mathbf{x})$ is a discrete neighborhood of $\mathbf{x}$ defined as

$$N(\mathbf{x}) = \{\mathbf{y} \in \mathcal{I}; \parallel \mathbf{y} - \mathbf{x} \parallel = 1\} \quad \forall \mathbf{x} \in \mathcal{I}. \tag{3}$$

where $\parallel \mathbf{y} - \mathbf{x} \parallel$ is an Euclidean distance. The interface Γ lies between the points in L_{in} and L_{out}.

For the evolution algorithm the values of φ are not important far from the boundary. For simplicity of the implementation we shall therefore confine the values of φ to a limited set of integers $\{-3, -1, 1, 3\}$ (see Figure 1). Close to its zero level set, the function φ still roughly approximates the signed distance function. Formally, we define this modified level set function as follows:

$$\varphi(\mathbf{x}) = \begin{cases} 3 & \text{if } \mathbf{x} \text{ is an exterior point} \quad (\mathbf{x} \in \Omega_b \wedge \mathbf{x} \notin L_{out}); \\ 1 & \text{if } \mathbf{x} \in L_{out}; \\ -1 & \text{if } \mathbf{x} \in L_{in}; \\ -3 & \text{if } \mathbf{x} \text{ is an interior point} \quad (\mathbf{x} \in \Omega \wedge \mathbf{x} \notin L_{in}). \end{cases} \tag{4}$$

2.1.1 Interface evolution

The evolution of the boundary is driven by a scalar velocity field v. It reflects the image based external force. Positive $v(\mathbf{x})$ value stands for a foreground image pixel and indicates that the boundary is to be moved outwards and vice-versa. Interface evolution is realized by updating the L_{in} and L_{out} sets as follows. If a value of the velocity field $v(\mathbf{x})$ corresponding to a point $\mathbf{x} \in L_{out}$ is positive, then the interface moves outward at this point. We delete $\mathbf{x}$ from the set L_{out}, add it to the set L_{in} and set $\varphi(\mathbf{x}) = -1$. At this time the outer boundary of Γ, L_{out}, is interrupted. To connect it again, we add all points $\mathbf{y}$ from the $N(\mathbf{x})$ neighborhood whose φ value is equal to 3, to the set L_{out} and set their $\varphi(\mathbf{y}) = 1$. This procedure is repeated for all points from L_{out}.

The inward move is handled in a similar way. For each point $\mathbf{x} \in L_{in}$ where $v(\mathbf{x})$ is negative, move this point from L_{in} to L_{out} and set $\varphi(\mathbf{x}) = 1$. Then we add all points $\mathbf{y}$ from the $N(\mathbf{x})$ neighborhood with $\varphi(\mathbf{y}) = -3$ to the set L_{in} and set $\varphi(\mathbf{y}) = -1$ to connect the interrupted inner Γ boundary.

After both outward and inward evolution procedures are performed, there might be some redundant points in the L_{out} and L_{in} sets. To ensure the correct topology of the outer and inner boundary of the zero level set, it is necessary to eliminate them as follows. After the outward movement, we delete all $\mathbf{x} \in L_{in}$ for which all neighbors $\mathbf{y} \in N(\mathbf{x})$ have $\varphi(\mathbf{y}) = -1$, and set $\varphi(\mathbf{x}) = -3$. Similarly, after the inward evolution, we eliminate all $\mathbf{x} \in L_{out}$, for which all neighbors $\mathbf{y} \in N(\mathbf{x})$ have $\varphi(\mathbf{y}) = 1$, and set $\varphi(\mathbf{x}) = 3$.

These alternating steps make an evolution part of the algorithm. Each execution of the evolution part moves the interface Γ at most by one pixel at each interface point.

2.1.2 Interface smoothing

In traditional level sets methods regularization is mostly done using internal forces based on curvature. Internal forces are computed using the Laplacian of φ. In this algorithm,[7] we use the fact that curvature flow is equivalent to Gaussian smoothing of the level set function φ.[8–10]

The Gaussian filtering of the function φ is calculated only at L_{in} and L_{out} points. If the filtering result (see below) has a different sign than the original value $\varphi(\mathbf{x})$, the boundary is updated accordingly. We use an anisotropic Gaussian filter G

$$G(m,n,o) = e^{-\left(\left(\frac{m}{2\sigma_m}\right)^2 + \left(\frac{n}{2\sigma_n}\right)^2 + \left(\frac{o}{2\sigma_o}\right)^2\right)} \tag{5}$$

where $\sigma_m, \sigma_n, \sigma_o$ are standard deviations. The standard deviations are adapted to the voxel size. For example, for a voxel size $0.625 \times 0.625 \times 3$ mm, we might choose σ as $2 \times 2 \times 0.42$ pixels. The 3D convolution is

$$\big(G * \varphi\big)(x,y,z) = \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} \sum_{o=-O/2}^{O/2} \varphi(x-m, y-n, z-o) \cdot G(m,n,o). \tag{6}$$

where M, N, O are the mask sizes, chosen as $M = 3\sigma_m$, $N = 3\sigma_n$, $O = 3\sigma_o$.

Let us describe the smoothing operation in detail: For every point $\mathbf{x}$ in L_{out} compute $\widehat{\varphi}(\mathbf{x}) = G * \varphi(\mathbf{x})$. If $\widehat{\varphi}(\mathbf{x}) < 0$, delete $\mathbf{x}$ from L_{out}, add it to L_{in} and set $\varphi(\mathbf{x}) = -1$. For all $\mathbf{y} \in N(\mathbf{x})$ satisfying $\varphi(\mathbf{y}) = 3$, add $\mathbf{y}$ to L_{out} and set $\varphi(\mathbf{y}) = 1$. Similarly for L_{in} elements, if $\widehat{\varphi}(\mathbf{x}) > 0$, we delete $\mathbf{x}$ from L_{in}, add it to L_{out}, set $\varphi(\mathbf{x}) = 1$ and for all $\mathbf{y} \in N(\mathbf{x})$ satisfying $\varphi(\mathbf{y}) = -3$, add $\mathbf{y}$ to L_{in} and set $\varphi(\mathbf{y}) = -1$.

Redundant points can appear in the L_{out} and L_{in} sets, so we apply the same elimination procedure as in Section 2.1.1.

2.1.3 Iteration and control

Each major iteration consists of N_e evolution steps followed by N_g smoothing steps. The ratio of N_g versus N_e controls the amount of smoothing. In the original formulation,[7] the iteration is stopped after a predetermined maximum number of iterations or if the velocity field $v(\mathbf{x})$ at boundary points is stable, in the sense that no boundary point requires a movement:

$$v(\mathbf{x}) \leq 0 \quad \text{for all} \quad \mathbf{x} \in L_{out} \quad \text{and} \quad v(\mathbf{x}) \geq 0 \quad \text{for all} \quad \mathbf{x} \in L_{in} \tag{7}$$

However, we found experimentally that in many practical situations the stable configuration is never reached. Instead, the boundary starts to oscillate with a smoothing step undoing the effect of several previous evolution steps. We are therefore using an alternative stopping criterion: The iteration is stops if less than P boundary pixels change state between two major iterations. We normally set $P = \gamma\big(|L_{out}| + |L_{in}|\big)$ with $\gamma = 10^{-4}$.

2.1.4 Modified Chan-Vese segmentation criterion

The Chan-Vese energy functional[11] leads to a segmentation of an image into two regions (foreground and background) by the minimization of an energy function. Let $f(\mathbf{x})$ denote image intensity. The average intensity of the foreground Ω and the background Ω_b is c_1 and c_2, respectively. The energy function is defined as follows:

$$E(c_1, c_2, \Gamma) = \lambda_1 \int_{\Omega} (f(\mathbf{x}) - c_1)^2 d\mathbf{x} + \lambda_2 \int_{\Omega_b} (f(\mathbf{x}) - c_2)^2 d\mathbf{x} + \mu \int_{\Gamma} d\mathbf{x} + \nu \int_{\Omega} d\mathbf{x} \tag{8}$$

where $\int_{\Gamma} d\mathbf{x}$ is the length of the boundary Γ, $\int_{\Omega} d\mathbf{x}$ is the area of the object, and $\lambda_1, \lambda_2, \mu, \nu$ are non-negative weights. As in our case the regularization is handled separately, we set $\mu = 0$ and $\nu = 0$.

The following evolution equation converges to the minimum of (8) and is derived from the associated Euler-Langrage equation:

$$\partial_t \varphi = \delta(\varphi) \left[-\lambda_1 (f(\mathbf{x}) - c_1)^2 + \lambda_2 (f(\mathbf{x}) - c_2)^2 \right] \tag{9}$$

The part in brackets is proportional to the velocity field $v(\mathbf{x})$, see (1). Further simplification comes from the fact that the fast level set evolution (Section 2.1.1) needs only a binary (positive/negative) information. Hence, our speed function is defined as follows:

$$v(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad -\lambda_1 (f(\mathbf{x}) - c_1)^2 + \lambda_2 (f(\mathbf{x}) - c_2)^2 \geq 0; \\ -1 & \text{if} \quad -\lambda_1 (f(\mathbf{x}) - c_1)^2 + \lambda_2 (f(\mathbf{x}) - c_2)^2 \leq 0. \end{cases} \tag{10}$$

2.1.5 Algorithm summary

The entire algorithm is summarized in Algorithm 1. The algorithm inputs are the image f, an initial level set funtion φ, a number of evolution N_e and smoothing steps N_g, a standard deviation σ of the Gaussian kernel G, Chan-Vese criteria parameters c_1, c_2, λ_1 and λ_2, a maximum number of iterations N_{max} and the maximum percentage of changed boundary pixels γ for the stopping criterion. Alternatively, the initial sets L_{in}, L_{out} could be given. The output is the final level set function φ.

2.1.6 Implementation

We have implemented this algorithm in the *C* language as a *Matlab mex*-file. The *mex*-file handles computational operations and *Matlab* serves as a user interface.

A velocity field v is evalued by computing the Chan-Vese criteria at first with the given parameters and saved to a dynamically allocated array of the same size as the input image. As the values of v are binary, $v(\mathbf{x}) \in \{-1; 1\}$, the array is of *signed char* type. Then a Gaussian mask is created according to σ. Its values are also stored for simplicity to a dynamically allocated *signed char* typed array. The L_{in} and L_{out} sets are created searching the φ for its corresponding values (2). The sets are implemented as a dynamic bidirectionally connected lists, which has the advantage of being able to insert or delete any element in constant time. Each node contains an image point (pixel/voxel) coordinates.

In 3D, the basic ideas remain the same, although it was necessary to modify the algorithm structures and extend the operations to 3D. The implementation is general and can handle both 2D and 3D data.

1. Compute the velocity field v (10);
2. Create the Gaussian mask G (5);
3. Create the L_{in} and L_{out} from φ;
4. **while** the stopping criterion is not reached **do**:
   - **for** $i = 1$ to N_e **do**:
     - Outward evolution;
     - Eliminate redundant points in L_{in};
     - Inward evolution;
     - Eliminate redundant points in L_{out};
   - **for** $i = 1$ to N_g **do**:
     - Outward evolution smoothing;
     - Eliminate redundant points in L_{in};
     - Inward evolution smoothing;
     - Eliminate redundant points in L_{out}.
5. Return final φ

Algorithm 1: The fast level set algorithm

resolution	$128 \times 128(\times 32)$	$256 \times 256(\times 63)$	$384 \times 384(\times 94)$	$512 \times 512(\times 125)$
2D	0.009 [s]	0.023 [s]	0.047 [s]	0.061 [s]
3D	4.14 [s]	8.56 [s]	14.02 [s]	19.23 [s]

Table 1. Computational times for 2D a 3D MRI images with different resolutions.

3. EXPERIMENTS

The method was tested on real 2D and 3D CT and MRI scans. The testing computer was a 3.2GHz *Athlon 64* with 1GB memory and *Linux* operating system. We have used the han-Vese segmentation model with parametres $\lambda_1 = 0.1$ and $\lambda_2 = 0.1$ and the standard deviation $\sigma = 1$ for each experiment unless stated otherwise. Average levels c_1 and c_2 of the foreground and background were set for every experiment separately.

The first experiment (Figure 2) demonstrates the segmentation of bones from a $512 \times 512 \times 125$ 3D CT image, which is a part of The Visible Human Project.[12] The initial surface was a rectangular block placed in the center of the image with 90 % of each image dimension. The surface was then shrunk using the described method. A typical execution time was 63 s and the number of evolution iterations was 392.

The elapsed time depends on the selection of the initial surface. Figure 3 shows 3D reconstruction of the segmentation result for the same input image, with the initial surface being a small sphere centered at the coordinates (270, 240, 60) and with 10 pixel radius. Some of bones were not reached, but the execution time was only 19.3 s with 303 iterations. For both experiments we set $N_e = 250$ and $N_g = 5$.

We have also tested the dependence of the algorithm speed on the input data size, varying the resolution, for the same 3D data as in the previous experiment. We have also performed speed tests on a 2D image using 97th xy-slice of the 3D image. To change the data size, we used downsampling of the original data with scale factor s with values 0.25, 0.5, 0.75 and 1. The initial curve (resp. surface) was a circle (resp. sphere) of radius $10s$ centered at coordinate $(270, 240)s$ (resp. $(270, 240, 60)s$). For both 2D and 3D data we set $N_e = 250$ and $N_g = 5$ at all resolutions. The timings are shown in Table 1. It can be seen that the dependence of the algorithm duration time and the input data size is almost linear, especially in the 3D case.

We have also performed experiments on MRI data. The first experiment (Figure 4a) was a segmentation of corpus callosum from a 350×330 2D MRI image. The initial curve was a manually selected single point in the

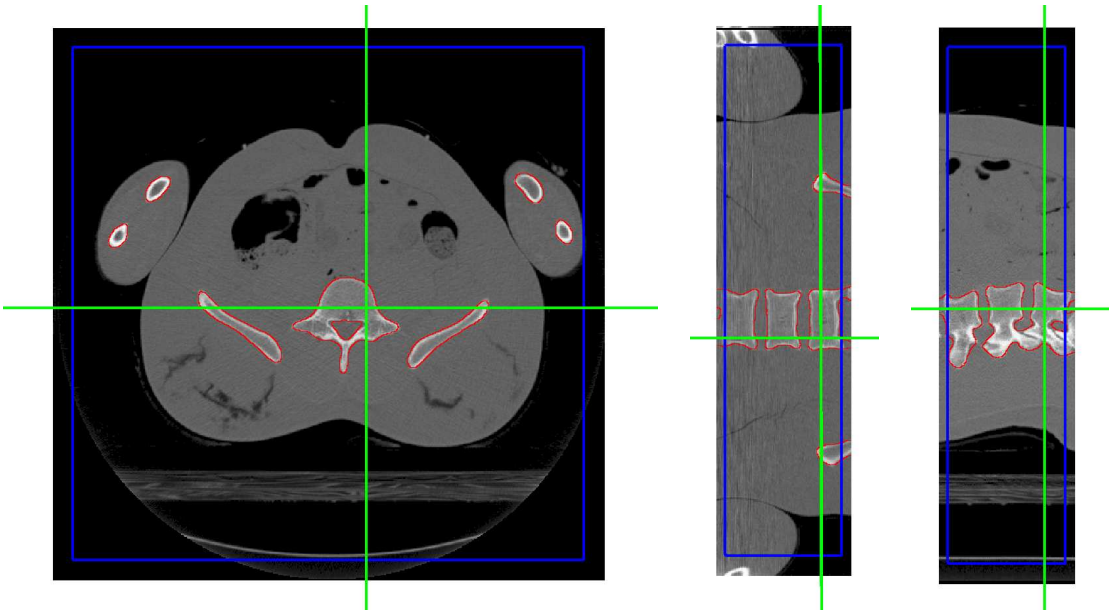


Figure 2. Segmentation of a 3D CT scan. Images (a) - (c) shows segmented 3D image of human thorax. Segmented bones are in red, the initial surface in blue and cut planes are marked by green lines. (a) *xy* slice, b) *xz* slice c) *yz* slice.

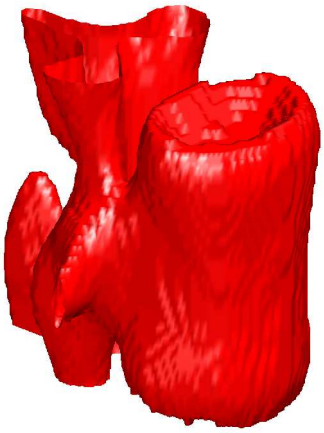
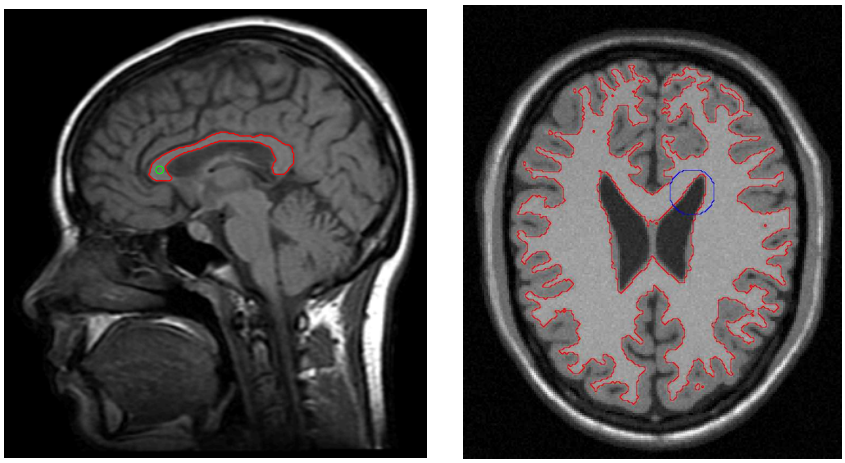


Figure 3. 3D rendering of the segmented vertebra

desired area. We used $N_e = 50$ and $N_g = 5$ for this experiment. The algorithm converged in 152 iterations and the running time was about 8 ms.

In the next experiment (Figure 4b) we segmented a 3D T1 MRI simulated brain image from the BrainWeb[13] with size of $217 \times 217 \times 181$ pixels and default settings (1mm slice thickness, 3% noise level and 20% intensity non-uniformity). The initial surface was a sphere of radius 10 centered at oordinates (90, 110, 100). N_e was set to 50 and N_g to 5. The algorithm reached the shown boundaries after 10.52 s and 192 iterations.

The algorithm speed and the resulting smoothness is dependent upon the selected ratio N_e:N_g of the number of evolution and smoothing steps. The higher the image resolution, the more evolution steps are needed. Low N_e:N_g ratio causes high smoothing and a slow convergence rate and vice versa. The smoothing cycle is very slow in comparison with a duration of the curve evolution cycle, especially in the 3D case. Table 2 shows elapsed time as a function of N_e:N_g. We have used the same input data as before (the 3D $512 \times 512 \times 125$ CT image).



(a) (b)

Figure 4. (a) Segmented 2D MRI image. The green circle marks the initial point and segmented corpus callosum is in red. (b) One slice of segmented 3D MRI image. We show the initial curve (surface) in blue and the recovered boundaries in red.

N_e:N_g	250:5	150:5	50:5
algorithm duration [s]	12.38	23.57	111.43
smoothing cycles duration [s]	2.73	6.17	46.02

Table 2. Comparison of an N_e:N_g ratio time duration dependence

Figure 5a illustrates the cause of oscillatory solutions: small thin parts of the velocity field v 'landscape' through which the boundary evolves. If the Γ interface does not reach the end of the thin part in N_e steps, the subsequent smoothing cycle will erase all of the thin part and undo the Γ evolution (Figure 5b). This situation might repeat with a longer period too, e.g. 4 iterations.

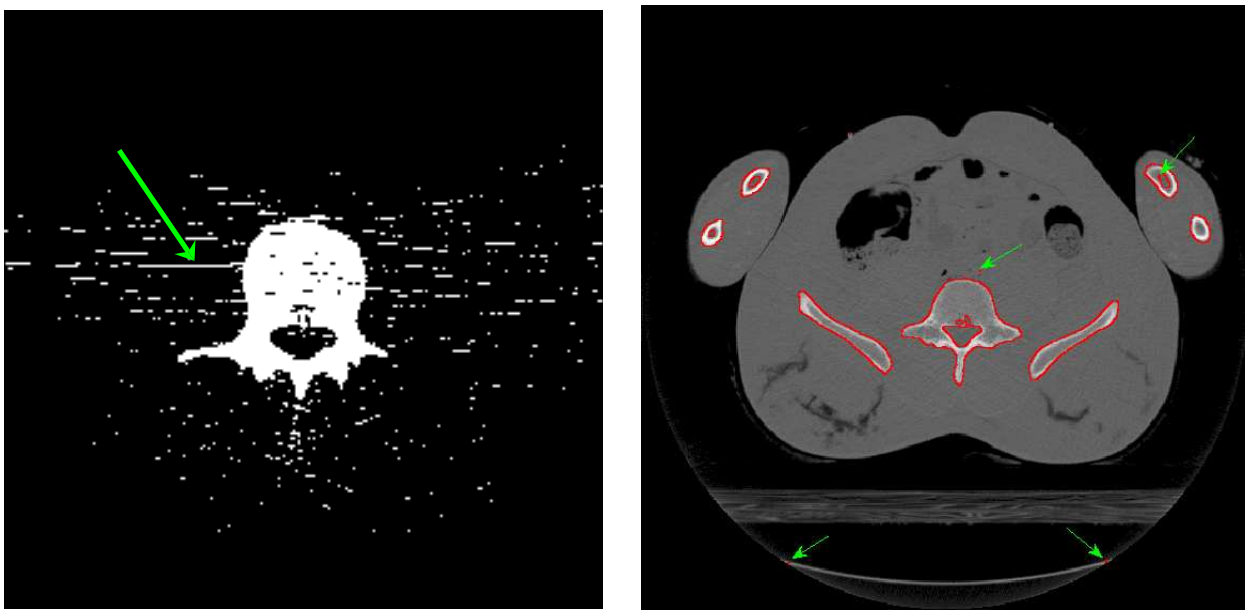
The level of the boundary smoothness and convergence speed is primarily determinated by the number of smoothing steps N_g and the standard deviation σ of the Gaussian filter (Figure 6). The impact of the N_g on the segmentation result is shown in Figure 7. The higher the N_g the smoother the boundaries, but the longer the elapsed time (Table 3).

In the final set of experiments, we compare the segmentation results and the computational time of the presented fast level set algorithm and a threshold segmentation level set image filter implemented in ITK.[14] To obtain comparable results, the v field was computed by the same thresholding procedure in both cases.

The first experiment (Figure 8a) was an ITK training example with a segmentation of a white matter from an ITK example brain image *BrainProtonDensitySlice.png* of size 217×181. The level set function φ was initialized as a circle at coordinates (60, 116) and with a diameter 5. The remaining ITK algorithm parameters were set as $L = 150$ for the lower threshold, $H = 180$ for upper threshold and $RMS = 0.005$ as the ITK stopping criterion. ITK converged in 12.38 s and 1054 iterations. The proposed fast level set algorithm finished in 0.18 s after 230 iterations. Both reported times include image loading and saving. In Figure 8a you can see that the fast level set segmentation results are slightly better: the boundary curve of the fast level set segmentation got deeper into the concave regions while preserving the boundary smoothness.

N_g	1	3	9
algorithm duration [s]	12.38	68.36	356.24
total smoothing duration [s]	2.73	24.85	242.13

Table 3. Duration of all the smoothing cycles as a function of N_g.



(a) (b)

Figure 5. (a) The green arrow points to one of the trouble-causing thin parts in the velocity field v. (b) Green arrows indicate oscillating areas during the segmentation.

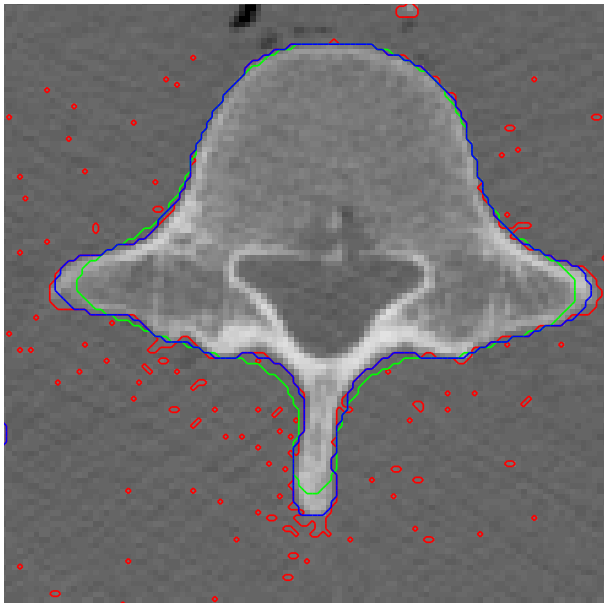
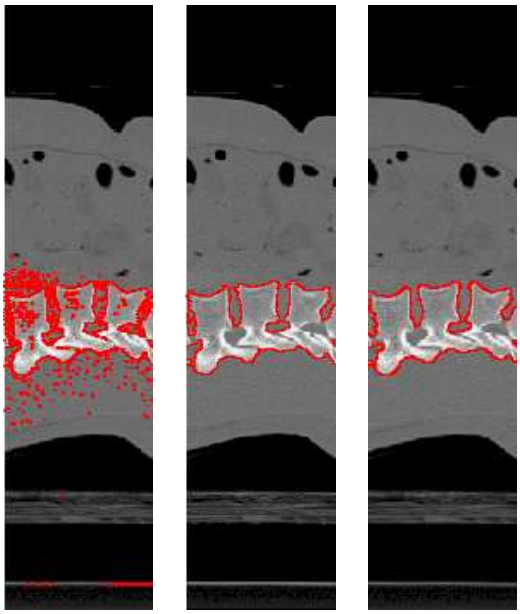


Figure 6. The impact of the choice of σ on the boudary smoothness. A low $\sigma = 0.5$ (in red) causes low smoothness with segmentation artifacts. On the other hand, a high $\sigma = 2.5$ (in green) leads to excessive smoothing. A properly chosen $\sigma = 1.5$ (in blue) fits the boundary correctly.

In the second comparative experiment (Figure 8b) we have segmented a 2D CT slice of a human cervical vertebra from the *The Visible Human Project*.[12] The image size was 512×512 and the initial contour was a circle with diameter 5 pixels, with center coordinates $(230, 256)$. The ITK algorithm parameters were set to: $L = 100$, $H = 140$ and $RMS = 0.005$. The timings were 4.56 s (ITK) versus 0.14 s (fast level set) and the number of iterations were 756 and 184, respectively.



a) b) c)

Figure 7. Effect of different selection of N_g with fixed $N_e = 250$. (a) - 1, (b) - 3, (c) - 9

4. CONCLUSION

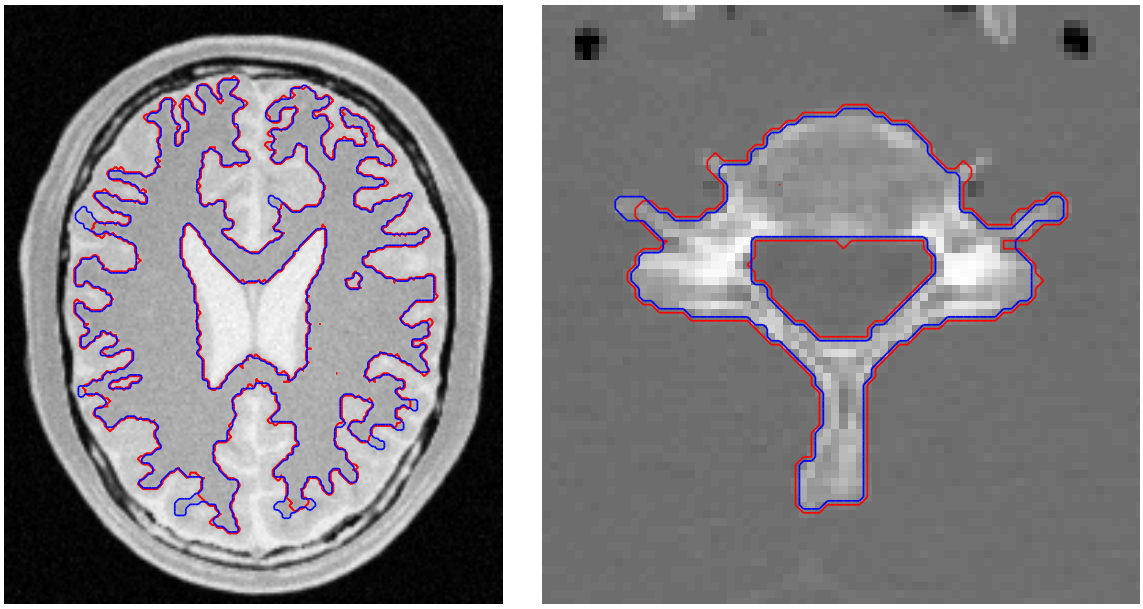
Segmentation of bones from 3D CT and MRI images for construction of 3D computer bone models is a very important task for biomechanical bone analysis. We have presented a novel implementation of the fast level set segmentation algorithm,[7] which works on both CT and MRI image modalities with promising results. The algorithm was extended to 3D.[15] Our further improvements include anisotropic smoothing and a new algorithm stopping criterion.

5. ACKNOWLEDGMENTS

The authors were supported by the Czech Ministry of Education under Projects MSM6840770012 and 1M0567.

REFERENCES

1. J. Vrána, Z. Horák, and J. Michalec, "Finite element analysis of human hip joint," *Inzynieria Biomaterialow* **8**(47), pp. 21–22, 2005.
2. L. Zach, S. Konvičková, and P. Růžička, "Development of a finite element knee model after a total replacement operation," *Proceedings of the 4th Youth Symposium on Experimental Solid Mechanics* , 2005.
3. L. Zach, S. Konvičková, and P. Růžička, "Finite element analyses and other tests of ceramic knee joint replacement (WDM system)," *The 3rd European Medical and Biological Engineering Conference* **11**(2), pp. 1727–1983, 2005.
4. S. Osher and J. A. Sethian, "Fronts propagation with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of computational physics* **79**, pp. 12–49, 1988.
5. S. Osher and R. P. Fedkiw, "Level set methods: An overview and some recent results," *Journal of Computational Physics* **169**(2), pp. 463–502, 2001.
6. V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision* **22**, pp. 61–79, 1997.
7. Y. Shi and W. C. Karl, "A fast level set method without solving PDEs," *IEEE International Conference on Acoustics, Speech, and Signal Processing* , pp. 97–100, 2005.
8. A. Hummel, "Representations based on zero-crossings in scale-space," *Proc. CVPR* , pp. 204–209, 1986.



(a) (b)

Figure 8. Comparison of the fast level set method and the ITK threshold segmentation level set image filter results. Image (a) shows the segmentation of the gray matter from a 2D MRI image and (b) the segmentation of the cervical vertebra from a 2D CT scan. The ITK results are in red, the results of the fast level set method in blue.

9. J. Koenderink, "The structure of images," *Biol. Cybern.* **50**, pp. 363–370, 1984.
10. P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Machine Intell* **20**, Jul 1990.
11. T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans on Image Processing* **10**(2), pp. 266–277, 2001.
12. U. N. L. of Medicine, "The visible human project." http://www.nlm.nih.gov/research/visible/visible_human.html.
13. "Brainweb: Simulated brain database." http://www.bic.mni.mcgill.ca.brainweb.
14. "The insight toolkit." http://www.itk.org.
15. Y. Shi and W. C. Karl, "A fast implementation of the level set method without solving partial differential equations," Tech. Rep. ECE-2005-02, Department of Electrical Computer Engineering, Boston University, January 2005.