GPU-Based Segmentation of Cervical Vertebra in X-Ray Images

Sidi Ahmed Mahmoudi, Fabian Lecron, Pierre Manneback, Mohammed Benjelloun and Saïd Mahmoudi
University of Mons, Faculty of Engineering
Computer Science Department
Place du Parc, 20
7000 Mons, Belgium
Email: {Sidi.Mahmoudi, Fabian.Lecron, Pierre.Manneback, Mohammed.Benjelloun, Said.Mahmoudi}@umons.ac.be

***Abstract*—The segmentation of cervical vertebra in X-Ray radiographs can give valuable information for the study of the vertebral mobility. One particular characteristic of the X-Ray images is that they present very low grey level variation and makes the segmentation difficult to perform. In this paper, we propose a segmentation procedure based on the Active Shape Model to deal with this issue. However, this application is seriously hampered by its considerable computation time. We present how vertebra extraction can efficiently be performed in exploiting the vast processing power of the Graphics Processing Units (GPU). We propose a CUDA-based GPU implementation of the most intensive processing steps enabling to boost performance.**

Experimentations have been conducted using a set of high resolution X-Ray medical images, showing a global speedup ranging from 15 to 21, by comparison with the CPU implementation.

***Index Terms*—Active Shape Model, CUDA, Edge Detection, GPU, Vertebra Segmentation, X-Ray Medical Images.**

I. Introduction

A medical image builds up a valuable source of information for a specialist. It allows him to visualize indirectly features of the human body. Thereby, medical imaging is of great importance nowadays.

In this paper, we treat the mobility of the cervical vertebræ. The interest of the present work is to measure in an automatic way the movements of every vertebra compared to the others. This quantitative information, medical imaging cannot provide it directly. A specific image processing has to be performed in order to bring the specialist quantitative data helping him in his diagnosis process.

The radiography of the spinal column remains a common medical exam. It is one of the fastest and the cheapest way to detect vertebral abnormalities. For those reasons, images we deal with are X-Ray radiographs focusing on the three particular patient's positions in this kind of study: flexion, extension and neutral.

The objective we pursue is to achieve the segmentation of the cervical vertebræ on radiographs. In a previous paper [1], we showed the effectiveness of the Active Shape Model for the study of the cervical vertebræ mobility. We carried on our research work by proposing a methodology in order to automate the segmentation process with a corner detection by polygonal approximation [2]. One drawback of our approach is the excessive computing time due to the high resolution of the radiographs. This disadvantage is often encountered in medical image processing (large database, high resolution, etc.). The GPUs represent a possible solution to this problem. In order to reduce it, we propose to deploy part of the implementation on GPU.

This paper presents a GPU implementation for edge detection phase of the proposed algorithm. Indeed, we develop a parallel implementation of the recursive contour extraction technique using Canny's criteria [3]. Our choice to parallelize this method is based on its noise robustness and its reduced number of operations. These factors allow applying the application on large sets of medical images. This enables to have more precise results for vertebra extraction.

The remainder of the paper is organized as follows: related works are described in section 2. Section 3 presents the CPU implementation of the proposed method based on Active Shape Model. Section 4 discusses the use of GPU for image processing algorithms, while section 5 is devoted to the GPU implementation of our approach based on CUDA. Section 6 presents the obtained results of vertebra extraction using a data set of medical images, and compares the performances between CPU and GPU implementations. Finally section 7 concludes and proposes further work.

II. Related Work

One can find two kinds of related work for which vertebra segmentation and optimal edge detection in medical images are the fundamental processing steps: the first one is related to sequential solutions for vertebra extraction using CPUs, and the second is related to the use of GPU to accelerate image processing algorithms, which can be exploited for medical applications.

A. Vertebra Segmentation on CPU

One can discover a lot of segmentation methods applied to vertebra extraction in the literature. Their application depends on the technology considered. MRI (Magnetic Resonance

Imaging) images have been segmented by graph theory techniques, namely the search of the minimal cut in a graph representing the image [4], [5].

A videofluoroscopic system can be used for the diagnosis of the low back pain. The active contour algorithm has been adressed to the segmentation of this kind of images. In [6], the authors work with a classical definition of the active contour and in [7], they combine it with complex wavelets.

Computed Tomography (CT) is also a medical imaging system whose the usage has increased over the last two decades. The level set method, which makes an interface evolve in the image, has been used in the context of CT images segmentation [8].

One specific characteristic of the X-Ray images is that they present very low grey level variation. Some researchers have used innovative approaches to segment vertebræ on radiographs. Let's mention the generalized Hough transform in [9] and the template matching in [10]. Nevertheless, the model-based methods such as the Active Shape Model [11] or Active Appearance Model [12] are acknowledged for being robust methods to deal with this type of images.

B. GPU for image processing

Many image processing algorithms have phases which consist of a common computation over many pixels. This fact makes these algorithms prime candidates for acceleration on GPU by exploiting processing units in parallel. This is particularly important for medical imaging applications. In this category, [13] implements several classic image processing algorithms on GPU with CUDA. OpenVIDIA project [14] has implemented different computer vision algorithms running on graphic hardware such as single or multiple graphics processing units, using OpenGL [15], Cg [16] and CUDA [17]. Authors in [18] have presented a GPU implementation of the Canny edge detector [19]. There are also some works for new volumetric rendering algorithms implemented on GPU [20], [21] and magnetic resonance (MR) image reconstruction [22].

Our contribution will deal with vertebra segmentation in X-Ray medical images, exploiting a method based on the ASM (Active Shape Model). The initialization of the ASM search is performed by edge detection, and corner extraction which is obtained by polygonal approximation. We contribute also to accelerate the computing time of vertebra extraction by parallelizing the most intensive step of the proposed approach on GPU. Indeed, we propose a parallel implementation of a recursive edge detection method using Canny's design [3]. The choice to parallelize this recursive method on GPU is essentially due to the noise truncature immunity factor and also to the reduced number of operations required by such method.

III. Medical Application on CPU

Our application of vertebra extraction is based on the Active Shape Model [11]. The principle of this method is to design a statistical shape model of a specific object (vertebræ in our case) from a sample database. Once the model is determined, it can be used to detect other similar shapes in new images. To this end, the mean shape model is extracted and placed in an area of interest. The shape then becomes warped to fit at the best the real edge of the object.

Four steps form the ASM segmentation process: learning, model design, initialization and segmentation. We review them in the next sections.

A. Learning

The learning phase seeks to build up a sample of images which will be the basis of the model. On each of these radiographs, an operator is expected to mark the vertebræ C3 to C7[1] with some predefined landmarks.

B. Model Design

As soon as the images of the sample have been marked, every vertebra form will be used in order to build the Active Shape Model. It is necessary to align the shapes with each other so that the data processing is statistically coherent. To this aim, Algorithm 1 is exploited.

Algorithm 1 Shapes Alignment

```
Aligning all the shapes of the sample on the first one
repeat
    Compute the mean shape
    Adjust the mean shape to the first shape
    Align each shape on the mean shape
until convergence
```

In the above algorithm, once the mean shape has been calculated, a group of other allowable shapes can be derived by moving the landmarks in specific directions, called the variation modes. For more details on the model design, we report the reader to [23].

C. Initialization

The search initialization consists in placing the mean shape previously computed at an interest position close to the real object. This operation can be realized manually or in an automatic way. In [2], we propose an automation procedure composed of three stages: a contrast limited adaptive histogram equalization, an edge detection and finally an edge polygonal approximation. A human intervention is required only to specify three particular positions on the cervical radiograph, *i.e.* the top of the C3 vertebra, the top of the C5 vertebra and finally the bottom of the C7 vertebra.

[1]Fig. 5(b) gives a idea of the vertebræ location. C3 is the first segmented vertebra and C7 the last one.

1) Contrast Limited Adaptive Histogram Equalization: The X-Ray radiographs we treat are characterized by very low contrasts. Sometimes, it is not obvious to distinctly visualize the edge of some vertebræ with naked eyes. Therefore, a preprocessing is necessary in order to decrease the impact of that problem and to highly improve the quality of the edge detection.

The simple histogram equalization is not adapted to the local contrast improvement and is strongly influenced by nonrelevant areas of the image. To deal with such drawbacks, the principle of the contrast limited adaptive histogram equalization [24] is to divide the image into contextual regions in which the corresponding histogram has to be equalized. Nevertheless, this equalization per region lead to unwanted visual boundaries. A bilinear interpolation scheme is proposed to reduce this effect.

Let's consider A, B, C and D, the centers of 4 contiguous contextual regions. If y is the distance between AB and a pixel p_i, and x the distance between AC and p_i, then the interpolation scheme is given by:

$$\begin{aligned} s =&(1-y)[(1-x)T_A(r)+xT_B(r)] \\ &+y[(1-x)T_C(r)+xT_D(r)] \end{aligned} \tag{1}$$

Only applying this interpolation scheme comes to perform an adaptive histogram equalization. However, this process is still dependent to the increase of the noise in the radiograph. One way to decrease this effect is to reduce the contrast improvement in the homogeneous areas. In such a way, a contrast factor is defined to limit the highest peaks in the equalized histogram. The pixels above this factor limit are uniformly redistributed in the histogram.

2) Edge Detection: The Canny filter allows to detect edges in an image by taking advantage of the information given by the intensity gradient. The first step is to decrease the noise in the image by convolving it with a Gaussian filter such as (2). In this way, the pixels with low intensity are removed from the radiograph.

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

Next, the Sobel operator is applied on the image for approximating the vertical and horizontal derivatives. It consists in a couple of 3x3 convolution masks given in (3).

$$G_x=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix}*I \qquad G_y=\begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix}*I \tag{3}$$

With the vertical and the horizontal derivatives, the gradient of each pixel can ben determined by (4). Furthermore, the orientation of the gradient has to be calculated.

$$G=\sqrt{G_x^2+G_y^2} \tag{4}$$

Once the gradient for each pixel is known, it is necessary to enumerate the local maxima. High gradient intensity means high probability of edge presence at the current pixel. For these reasons, points with local maxima are considered as part of an edge.

Finally, a binarization by hysteresis thresholding is performed in the image. A low and high thresholds are defined so that for each point, if the gradient intensity is:

- lower than the low threshold, the point is rejected;
- greater than the high threshold, the point is part of the edge;
- between the low and the high thresholds, the point is accepted only if it is connected to an already accepted point.

3) Edge Polygonal Approximation: In this work, finding the points of interest in the image is realized by an edge polygonal approximation. In [2], we have proven that usual corner detectors such as the Harris one are not suitable for X-Ray cervical radiographs. Actually, common methods to detect points of interest are based on grey level intensity. However, let's remind that the images we deal with are characterized by very low contrast variations.

Therefore, a geometrical definition of a corner is used to detect it, *i.e.* a point is considered as a corner if it is located at the intersection of two segment lines. The edges are transformed in lines by a polygonal approximation. The algorithm used is the one proposed by Douglas and Peucker in [25] and by Ramer in [26]. The main idea of this approach is that a polyline represented by n points can be reduced in a representation with 2 points if the distance between the segment line joining the extremities of the polyline and the farest point from this line is lower than a given threshold.

First, the method selects the extremities E_1 and E_2 of the polyline. Let A be the farest point from the segment line $\|E_1E_2\|$ and d, the distance between the point A and $\|E_1E_2\|$. Three scenarii are considered and presented at the Algorithm 2.

Algorithm 2 Polygonal Approximation

```
Select the extremities $E_1$ and $E_2$ of the polyline
if $d \leq \epsilon$ then
    all the points situated between $E_1$ and $E_2$ are removed
else if if $d > \epsilon$ then
    the algorithm is recursively applied on 2 new polylines:
    $\|E_1A\|$ and $\|AE_2\|$
else if there is no point between $E_1$ and $E_2$ then
    the polyline is no longer reducible : STOP
end if
```

D. Segmentation

The knowledge of the upper and lower left vertebra corners provides us a relevant information about the vertebra position, orientation and height. Therefore, it is possible to accurately place the mean shape on every detected position.

From this moment, the ASM search can be executed. For every landmark defining the starting shape, the neighbourhood texture is analyzed in a specific direction: the normal of the contour. This analysis is made by considering landmarks along the normal of the contour at the considered point. A profile is then defined as a vector containing the gradient of intensity for each point in the normal. A point on the current shape is moved to the position where the profile is the closest to the mean shape one according to the Mahalanobis distance [11].

The search algorithm is given here:

Algorithm 3 ASM Search

```
repeat
    • Search, along each normal of the shape, the best profile
    according to the computed mean shape. The new sections
    are landmarks and profiles found
    • Search the shape model which is the best suited to
    points found in the previous step. This will form the basis
    for the next iteration
until the convergence condition is met or the maximum
number of iterations is reached
```

In this algorithm, one can see that a convergence condition is used. Here, we propose to stop the search when all the landmarks of the shape remain stable. Nevertheless, it appears that this condition is too strict. Therefore, we consider the number of equivalent points that have a different position between the current and the previous shape. Finally, the search is stopped if this number is 10% of the previous one. In order to avoid indefinite loop, a maximum number of iterations can be given. Generally, the number of iterations is between 50 and 250 iterations.

E. Discussion

Among all the steps composing the search ASM process, we observed that the initialization is the most time-expensive. If we distinguish the components which define it, it appears that the edge detection step is the heaviest. A lot of computation is necessary to detect the edges in high resolution images.

Naturally, the annotating phase by an operator remains very long and tedious. Nevertheless, this step has to be performed only once and is anyway not parallelizable.

The time to build the Active Shape Model depends on the number of images in the sample. In the context of this work, we consider a number of 75 images. This leads to acceptable computation times.

Given these observations, it seems that it is relevant to concentrate only on the reduction of computing time related to the initialization of the ASM search. The model design can be time-consuming but has to be performed only once. Therefore, in the next sections, we will explain how parallel algorithms on GPU have been implemented for the steps dedicated to the initialization of the ASM search. The interest of this work is directly relevant for a specialist who treats a large set of medical images.

IV. Image Processing on GPU

Image processing algorithms represent an excellent topic for acceleration on GPU, since the majority of these algorithms have sections which consist of a common computation over many pixels. This fact is due to the exploitation of the high number of GPU's computing units in parallel. As a result, we can say that graphic cards represent an efficient tool for boosting performances of image processing techniques. This section describes firstly the key factors of GPUs and the programming languages used to exploit their high power, and secondly the proposed development scheme based upon CUDA for parallel constructs and OpenGL for visualization.

A. GPU Programming

Graphics processing units (GPUs) have considerably evolved during last years as shown in Fig. 1. This evolution makes them a very high attractive hardware platform for general purpose computations. For a better exploitation of this high power, the GPU's memory bandwidth has also well evolved as shown in Fig. 2.

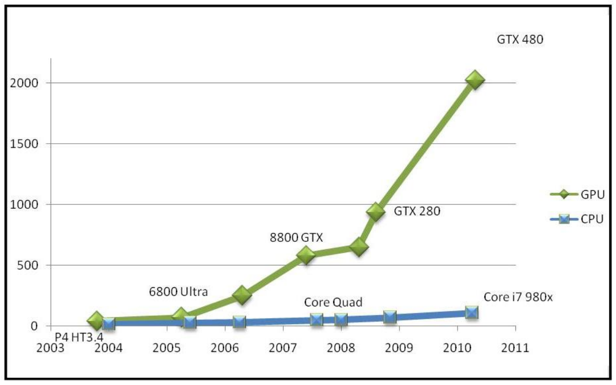


Fig. 1. Computational Power: GPU vs. CPU. Derived from [27]

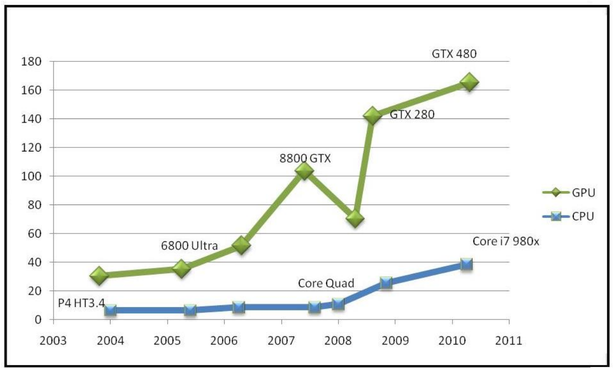


Fig. 2. Memory Bandxidth: GPU vs. CPU. Derived from [27]

In this context, NVIDIA launched the API CUDA (Compute Unified Device Architecture) [17], a new programming approach which exploits the unified design of the most current graphics processing units from NVIDIA. Under CUDA, GPUs consist of many processor cores which can address directly to

GPU memories. This fact allows a more flexible programming model than previous GPGPU programming models [28]. As a result, CUDA has rapidly gained acceptance in domains where GPUs are used to execute different intensive parallel applications.

B. Image Processing Model based on CUDA and OpenGL

This paragraph describes a proposed model for loading, processing and displaying images on GPU. This model is represented by a scheme development based upon CUDA for parallel constructs and OpenGL for visualization, which reduces data transfer between device and host memories. This scheme is based on four principle steps as follows: (Fig. 3)

- Copy input data
- Threads allocation
- Parallel Processing with CUDA
- Output Results

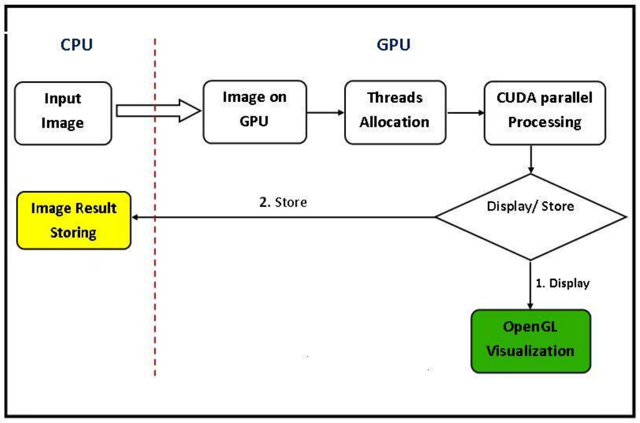


Fig. 3. Image Processing on GPU based on CUDA and OpenGL

1) Copy Input Data: The first step is the transfer of input data (image) from host (CPU) to device (GPU) memory. This enables to apply GPU processing on the copied image.

2) Threads Allocation: Once the input image is loaded on GPU memory, one can select the number of threads of the GPU's grid computing, so that each thread can perform its processing on one or a group of pixels. This will allow threads to process in parallel image pixels. We note that the selection of the number of threads depends on the number of image pixels.

3) Parallel Processing with CUDA: After selecting the threads number we can apply our CUDA kernels, these kernels will be executed N times using the selected N threads in the previous step.

4) Output Results: When performing parallel processing based on CUDA, results can be presented using two different scenarii:

1) OpenGL Visualization: In this case, we display the output images using the graphic library OpenGL [15] which allows a fast visualization since it uses buffers already existing on GPU. The choice of OpenGL is due to its compatibility with CUDA and also to avoid more data transfer between host and device memories. This scenario is useful when we apply parallel processing on one image only since we can't display many images using one video output.

2) Results Storing: The visualization with OpenGL will be impossible in the case of applying treatments on a set of images using one video output only. In this case, we must transfer the output images from GPU to CPU memory in order to store them. This transfer time can be neglected thanks to the gain obtained with GPU.

V. GPU IMPLEMENTATION

We presented in section 3 the implementation details and steps of the proposed method of vertebra extraction on CPU. One disadvantage of this method is the computing time which increases significantly with the number of images and their resolution. This section presents a GPU implementation of the most intensive step which is used for edge detection. Thus, we propose the parallelization of a recursive algorithm for this step using Canny's design [3]. The noise truncature immunity and the reduced number of required operations make this method very efficient. The sequential implementation of this technique is based on four principle steps:

- Recursive gradient computation (G_x, G_y).
- Gradient magnitude and direction computation.
- Non maxima suppression.
- Hysteresis and thresholding.

We note that the recursive gradient computation step applies a Gaussian smoothing before filtering the image recursively using two Sobel filters in order to compute the gradients G_x and G_y. While the steps of gradient magnitude and direction computation, non maxima suppression and hysteresis represent the same steps used for Canny filter described in section 3.2.

The proposed GPU implementation of this recursive method is based on the parallelization of all the steps listed below on GPU using CUDA.

A. Recursive Gaussian Smoothing on GPU

The GPU implementation of the recursive Gaussian smoothing step is provided from the CUDA SDK individual sample package [29]. This parallel implementation is applied on Deriche recursive method [3]. The advantage of this method is that the execution time is independent of the filter width. The use of this technique for smoothing allows to have a better noise truncature immunity which represents an important requirement for our application.

B. Sobel Filtering on GPU

The recursive GPU implementation of this step is also provided from the CUDA SDK individual sample package [29]. This parallel implementation exploits both shared and texture memories which allow to boost performances. This step applies a convolution of the source image by two Sobel filters of aperture size 3 in order to compute horizontal and vertical gradients G_x and G_y at each pixel. The GPU implementation is based firstly on a parallel horizontal convolution across the columns for computing G_x and secondly on a parallel vertical convolution across the lines for computing G_y.

C. Gradient magnitude and direction computing on GPU

Once the horizontal and vertical gradients (G_x and G_y) have been computed. It is possible to calculate the gradient magnitude (intensity) using (5) and the gradient direction using (6). The CUDA implementation of this step is applied in parallel on image pixels, using a GPU grid computing containing a number of threads equal to image pixels number. Thus, each thread calculates the gradient magnitude and direction of one pixel of the image.

$$Gr = \sqrt{G_x^2 + G_y^2} \tag{5}$$

$$\Theta = \arctan\left(\frac{G_x}{G_y}\right) \tag{6}$$

D. Non-maxima suppression on GPU

After computing the gradient magnitude and direction, we apply a CUDA function (kernel) which enumerates the local maxima (pixels with high gradient intensity) and suppresses all non-ridge pixels since local maxima are considered as a part of edges. We proposed to load the values of neighbors pixels (left, right, top, and bottom) in shared memory since these values are required for the research of local maxima. The number of threads selected for parallelizing this step was also equal to image pixels number.

E. Hysteresis on GPU

Hysteresis represents the final step to product edges. It is based on the use of two thresholds T_1 and T_2. Any pixel in the image that has a gradient magnitude greater than T_1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a gradient intensity greater than T_2 are also selected as edge pixels. The GPU implementation of this step is provided from the CUDA implementation of Canny filter [18]. We extended this implementation by the use of GPU's shared memory for a fast loading of connected pixels values.

Fig. 4 shows briefly the steps of the proposed method of vertebra segmentation in X-ray images, showing the use of GPU in the steps listed above.

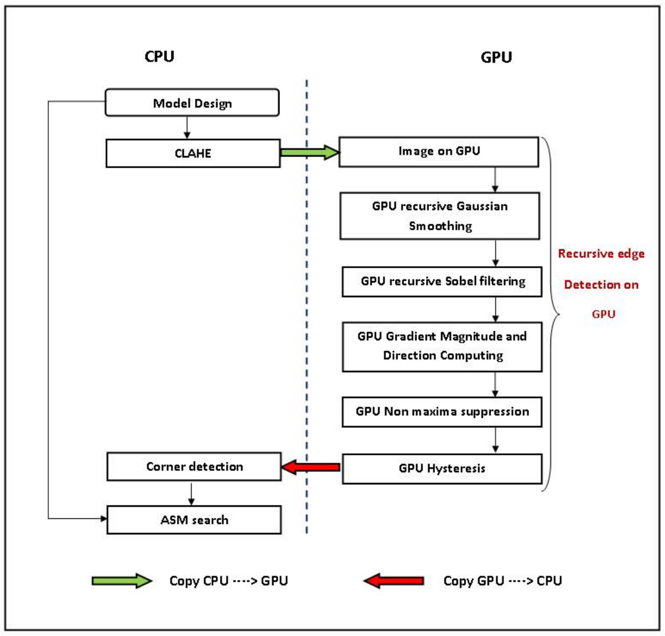

CPU
GPU
Model Design
CLAHE
Image on GPU
GPU recursive Gaussian Smoothing
GPU recursive Sobel filtering
GPU Gradient Magnitude and Direction Computing
GPU Non maxima suppression
GPU Hysteresis
Corner detection
ASM search
Recursive edge Detection on GPU
Copy CPU ----> GPU
Copy GPU ----> CPU

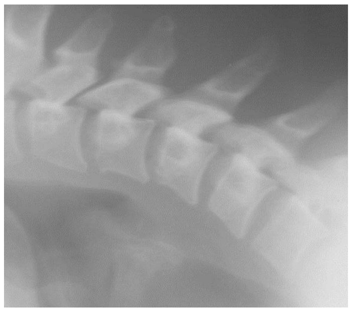
Fig. 4. Framework of Vertebra Segmentation using GPU

VI. Experimental Results

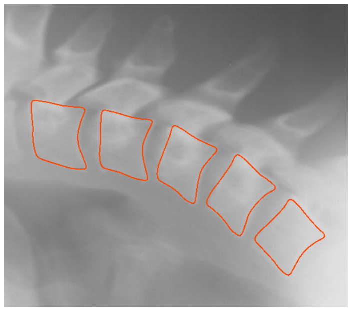
The validation of the results provided by the segmentation method based on the Active Shape Model has been proposed in a previous paper [1]. Anyway, the purpose of this paper is not to present a new segmentation technique but a novel way to implement the approach described in [1]. We invite the reader to observe Fig. 5 and Fig. 6 which give a view of the qualitative results provided by our method. A flexion and a neutral position are represented. For matter of simplicity, we trimmed the original image (1476*1680 resolution) and Fig. 5 and Fig. 6 represent the region of interest.

On the one hand, we can say that the quality of the vertebra segmentation remains identical since the procedure has not changed. Only the architecture and the implementation did. On the other hand, the use of GPU for vertebra extraction allowed us to accelerate the computation time. This acceleration is due to the GPU parallel implementation for edge detection step based on a recursive method using Canny's design. This fact enabled us to apply our proposed method on large sets of X-Ray medical images which allowed to have more precision for vertebra segmentation results.

TABLE I presents a comparison of the computing times between our proposed GPU implementation of the recursive edge detection, and the CPU implementation of Canny edge filter provided from the image processing library OpenCV [30]. This table presents edge detection results applied on one image using different resolutions, thus the GPU version exploits OpenGL for displaying results. While TABLE II presents results applied on a set of medical images (resolution: 1476*1680) that requires the transfer of resulting images from GPU to CPU memory for storing them, since we have only one video output. Notice that this transfer time can be neglected

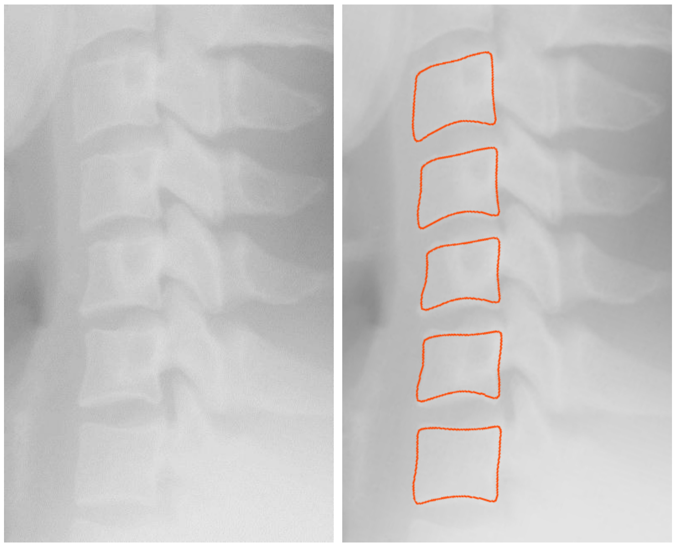


(a) Original Image.



(b) Segmented Image.

Fig. 5. Qualitative result of the segmentation in flexion position.



(a) Original Image. (b) Segmented Image.

Fig. 6. Qualitative result of the segmentation in neutral position.

thanks to the gain obtained using the GPU power, and hence is not included in TABLE II.

We note that the speedup in TABLE I is improved if we increase the resolution of images. The reason is that we explore more computing units in parallel. However, this speedup is more stable in TABLE II since the processing is applied sequentially between medical images which are treated in parallel.

Experimentations have been conducted on several platforms, *i.e.* GPU GTX280 and CPU Dual core :

- CPU: Dual Core 6600, 2.40 GHz, 2 GB RAM of Memory.
- GPU: NVIDIA GeForce GTX 280, 240 CUDA cores, 1GB of Memory.

Image Resolution	Canny Edge Detector (CPU)	Recursive Edge Detection (CUDA + OpenGL)	Speedup
512*512	30 ms	1.90 ms	15.78
1024*1024	101 ms	5.62 ms	17.97
1476*1680	267 ms	13.60 ms	19.63
3936*3936	1 497 ms	70.09 ms	21.10

TABLE I
PERFORMANCES OF GPU RECURSIVE EDGE DETECTION (CUDA + OPENGL) COMPARED TO CANNY EDGE DETECTION ON CPU (OPENCV)

Number of Images	Canny Edge Detector (CPU)	Recursive Edge Detection (CUDA)	Speedup
10	2.91 s	0.14 s	20.78
25	7.23 s	0.35 s	20.65
50	14.71 s	0.69 s	21.31
100	29.21 s	1.39 s	21.01

TABLE II
GPU PERFORMANCES OF RECURSIVE EDGE DETECTION ON A SET OF MEDICAL IMAGES

VII. CONCLUSION

In this paper, we focussed on the study of the vertebral mobility. To this aim, we developed a procedure to extract exactly the vertebra edges. We chose to base the segmentation phase on the Active Shape Model. The initialization of the ASM search is accomplished by the edge detection and the edge polygonal approximation in the image.

Generally, the computation time and noise immunity truncature represent the most important requirements in medical image processing and specifically for our application.

Thus, we proposed a GPU implementation of the recursive edge detection method using Canny's design. The use of GPU allowed us to accelerate the computation time, while the use of this recursive technique enabled a better noise removal. As a result, we obtained a faster application of vertebra extraction which can be applied on large sets of X-Ray medical images.

Finally, the parallel implementation of recursive edge detection using Canny's design is generic and can be exploited in any kind of image processing applications.

As future work we plan to parallelize the segmentation phase by deforming several models simultaneously. This way, the most intensive parts of the algorithm will be implemented fastly on GPU, and the less intensive ones stay implemented on CPU. We plan also to dispatch optimally the application steps on more available computing units (Multi CPU-GPU) by the use of the runtime system StarPU [31] which offers support for heterogeneous multicore architectures. Therefore, it will be possible to improve performances for vertebra segmentation.

Acknowledgment

The authors would like to thank the Communauté Française de Belgique for supporting this work under the ARC-OLIMP research project, grant AUWB-2008-12-FPMs11.

References

[1] M. Benjelloun, S. Mahmoudi, and F. Lecron, "A New Semi-Automatic Approach For X-Ray Cervical Images Segmentation Using Active Shape Model," in *Proceedings of the 3rd International Conference on Bio-inspired Systems and Signal Processing*, 2010, pp. 501–506.

[2] F. Lecron, M. Benjelloun, and S. Mahmoudi, "Points of interest detection in cervical spine radiographs by polygonal approximation," in *Proceedings of the 2nd International Conference on Image Processing Theory, Tools and Applications*, 2010, pp. 81–86.

[3] R. Deriche, "Using Canny's criteria to derive a recursively implemented optimal edge detector," *Internat. J. Vision,Boston*, pp. 167–187, 1987.

[4] S.-H. Huang, S.-H. Lai, and C. L. Novak, "A statistical learning appproach to vertebra detection and segmentation from spinal MRI," in *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2008, pp. 125–128.

[5] S.-H. Huang, Y.-H. Chu, S.-H. Lai, and C. L. Novak, "Learning-based vertebra detection and iterative normalized-cut segmentation for spinal MRI." *IEEE Transactions on Medical Imaging*, vol. 28, no. 10, pp. 1595–1605, 2009.

[6] S.-F. Wong, K.-Y. K. Wong, W.-N. K. Wong, C.-Y. J. Leong, and D.-K. K. Luk, "Tracking Lumbar Vertebrae in Digital Videofluoroscopic Video Automatically," in *Medical Imaging and Augmented Reality*, 2004, pp. 154–162.

[7] A. Wong, A. Mishra, P. Fieguth, D. Clausi, N. M. Dunk, and J. P. Callaghan, "Shape-guided active contour based segmentation and tracking of lumbar vertebrae in video fluoroscopy using complex wavelets." *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2008, pp. 863–866, 2008.

[8] S. Tan, J. Yao, M. M. Ward, L. Yao, and R. M. Summers, "Level set based vertebra segmentation for the evaluation of Ankylosing Spondylitis," *Proceedings of the SPIE. Medical Imaging 2006: Image Processing*, vol. 6144, pp. 58–67, 2006.

[9] G. Zamora, H. Sari-Sarraf, and L. R. Long, "Hierarchical segmentation of vertebrae from x-ray images," in *Medical Imaging 2003: Image Processing*, M. Sonka and J. M. Fitzpatrick, Eds., vol. 5032, no. 1. San Diego, CA, USA: SPIE, 2003, pp. 631–642.

[10] S. Mahmoudi and M. Benjelloun, "A New Approach for Cervical Vertebrae Segmentation," *Progress in Pattern Recognition, Image Analysis and Applications*, vol. 4756, pp. 753–762, 2008.

[11] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer vision and image understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[12] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active Appearance Models," in *5th European Conference on Computer Vision*. Springer-Verlag, 1998, pp. 484–498.

[13] Z. Yang, Y. Zhu, and Y. pu, "Parallel Image Processing Based on CUDA," *International Conference on Computer Science and Software Engineering. China*, pp. 198–201, 2008.

[14] J. Fung, S. Mann, and C. Aimone, "OpenVIDIA:Parallel gpu computer vision." *In Proc of ACM Multimedia*, pp. 849–852, 2005.

[15] OpenGL, "OpenGL Architecture Review Board: ARB vertex program. Revision 45." 2004. [Online]. Available: http://oss.sgi.com/projects/ogl-sample/registry/

[16] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard, "Cg: A system for programming graphics hardware in a C-like language," *ACM Transactions on Graphics 22*, pp. 896–907, 2003.

[17] NVIDIA, "NVIDIA CUDA," 2007. [Online]. Available: http://www.nvidia.com/cuda.

[18] Y. Luo and R. Duraiswani, "Canny Edge Detection on NVIDIA CUDA," *Proceedings of the Workshop on Computer Vision on GPUS, CVPR*, 2008.

[19] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–714, 1986.

[20] Y. Heng and L. Gu, "GPU-based Volume Rendering for Medical Image Visualization," *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai, China*, pp. 5145–5148, 2005.

[21] M. Smelyanskiy, D. Holmes, J. Chhugani, A. Larson, and al, "Mapping high-fidelity volume rendering for medical imaging to CPU, GPU and many-core architectures," *IEEE Transactions on Visualization and Computer Graphics, 15(6)*, pp. 1563–1570, 2009.

[22] T. Schiwietz, T. Chang, P. Speier, and R. Westermann, "MR image reconstruction using the GPU," *Image-Guided Procedures, and Display. Proceedings of the SPIE*, pp. 646–655, 2006.

[23] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Training models of shape from sets of examples," in *British Machine Vision Conference*, vol. 557, 1992, pp. 266–275.

[24] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. Ter Haar Romeny, J. B. Zimmerman, and K. Zuiderverld, "Adaptive histogram equalization and its variations," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987.

[25] D. H. Douglas and T. K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[26] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, pp. 244–256, 1972.

[27] GPU4VISION, "GPU4VISION," 2010. [Online]. Available: http://www.gpu4vision.org

[28] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum, 26(1)*, pp. 80–113, 2007.

[29] NVIDIA, "NVIDIA CUDA SDK code samples." [Online]. Available: http://developer.download.nvidia.com/compute/cuda/sdk/website/samples.html

[30] OpenCV, "OpenCV computer vision library." [Online]. Available: http://opencv.willowgarage.com/wiki/

[31] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures," *In Concurrency and Computation: Practice and Experience, Euro-Par 2009, best papers issue*, pp. 863–874, 2009.