Computing and Informatics, Vol. 33, 2014, 707–720

PARALLEL PROCESSING IN WEB-BASED INTERACTIVE ECHOCARDIOGRAPHY SIMULATORS

Adam Piórkowski

Department of Geoinfomatics and Applied Computer Science
AGH University of Science and Technology
al. Mickiewicza 30, 30-059 Cracow, Poland
e-mail: pioro@agh.edu.pl

Abstract. Medical simulation is a new method of education in medicine. It allows training medical students or practitioners without the need to involve patients and makes them familiar with various kinds of examinations, especially related to medical imaging. Simulators that visualize examinations or operations require large computing power to keep time constraints of output presentation. A common approach to this problem is to use graphics processing units (GPU), but the code is not portable. The method of parallelization of processing is more important in component environments, to allow calculating projections in real time. In this paper parallelization issues in the ultrasound view simulation based on provided computer tomography images are analyzed. The proposed domain decomposition for this problem leads to significant reduction in simulation time and allows obtaining an animated visualization for currently available personal computers with multicore processors. The use of a component environment makes the solution portable and makes it possible to implement a web-based application that is the basis for eTraining. The method for creating animation in real time for such solutions is also analyzed.

Keywords: Parallel processing, domain decomposition, server-side processing, simulation, ray-casting, interactive application, echocardiography, eTraining

Mathematics Subject Classification 2010: 68-04

1 INTRODUCTION

Modern methods of education have been equipped with new technology to facilitate learning. Widely used simulators allow to approximate the conditions of the medical examinations or operations and to get close into the real situation [1]. We distinguish between two main applications:

1. a long-term simulation – which goal is to perform, demonstrate or confirm the scientist research theses and
2. a real-time simulation of data showing the effect (usually of an operation or examination) – allows the student to become familiar with the operation technique without participation of the patient.

Recent methods allow the use of simulation via the Internet [2, 3]. This results in eLearning and eTraining systems [4, 5]. This paper presents an example for an echocardiography eTraining solution including the system design and methods of parallelization that allow compliance with the time constraints associated with smooth animation delivering a proper frame rate.

2 SIMULATION IN ECHOCARDIOGRAPHY

Extensive professional experience enables a physician to apply echocardiography efficiently. Transthoracic echocardiography (TTE) is simple, and does not cause discomfort to the patient. Training this technique is also easy for medical students. This is opposite to transesophageal echocardiography (TEE) examination. This examination allows much more accurate diagnosis of heart, but it is unpleasant for the patient. Another problem is in spatial orientation and mastery of instrument manipulation. In this case, there are very helpful simulators that allow physicians to prepare preliminary examination training without involving the patient [6, 7, 8]. Such simulators should implement simulation in a smooth manner. The process of simulation is time-consuming, so the use of parallel processing techniques may allow to calculate output images in a short span of time. Non-real-time simulators make a view in a long time [9] or use clusters [10, 11]. Other real-time ultrasound simulators use a GPU power for processing [12, 13, 14].

2.1 The CT2TEE – the First Web-Based Real-Time Simulator

One of the few solutions designed to TEE training is the CT2TEE project [8]. A similar solution, Virtual TEE Simulator [4], offers a limited number of on-line predefinied echocardiography views, and therefore is rather an eLearning than an eTraining tool.

The CT2TEE simulator generates TEE projections based on a model developed from computer tomography (CT) images [10, 13, 15]. Using prepared data, the simulator tracks the movement of a virtual probe along the esophagus. Generated

images include a projection as seen from the current position of a probe head. The projection takes into account the transformation of CT image brightness levels (according to an X-ray attenuation coefficient, based on the Hounsfield scale) to the brightness levels of ultrasonography (ultrasound reflections) [16]. Additionally, some characteristics of ultrasound effects, such as an acoustic shadow or coaxial noise are implemented. The standalone version of CT2TEE simulator allows real-time examination simulations.

The first version of CT2TEE simulator used a static model of the body [8]. The new version allows simulating a motion along with heart beat [17], therefore the model can contain a large amount of data.

2.2 Principles of Simulator Operations

CT2TEE simulator generates views corresponding to the TEE head views during a real examination. The user can specify several parameters such as:

- p – depth of probe insertion to determine position of probe,
- $\alpha(X, Y)$ – angle of rotation (range of $0 : 360$ degrees),
- $\beta(Y, Z)$ – angle of deflection (range of $-30 : 30$ degrees),
- $\gamma(X, Z)$ – plane angulation (range of $0 : 180$ degrees),
- d – scan depth.

All these variables $(p, \alpha, \beta, \gamma, d)$ create a space of control parameters for the virtual TEE probe.

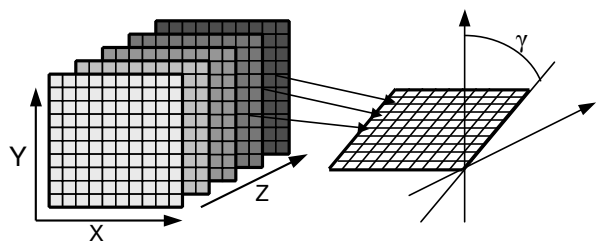

Figure 1. Creating a projection

There are several methods of ultrasound simulating classified by Zhu and Salcudean in [18]. One of the methods, used in the CT2TEE, assumes two stages – the first is to calculate a current projection of CT (Figure 1) [19], the second – to simulate an ultrasound beam. To calculate the current projection a superposition of rotations (α, β, γ) should be determined for each point of data (x', y', z') for this

projection, as shown in Equation (1).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (1)$$

The standard approach is to calculate the projection using matrix multiplication. It performs several operations that can be omitted (e.g. scalar multiplications by zero, iterations). For effective numeric calculations Equation (1) can be decomposed into simple operations that speed up processing:

```
xp=D11*x+D12*y+D13*z;
yp=D21*x+D22*y+D23*z;
zp=D31*x+D32*y+D33*z;
```

$$\begin{bmatrix} D11 & D12 & D13 \\ D21 & D22 & D23 \\ D31 & D32 & D33 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix} \quad (2)$$

where $D11 - D33$ are pre-calculated (calculated once) values (see Equation (2)) for a given projection.
Processing the independent calculations for each point of the projection (standard size of 512×512 pixels) allows to divide operations into domains (domain decomposition, described in Section 3). To smooth the edges, the trilinear interpolation is used [19].

The next step is to simulate the ultrasound beam using projection from CT. In the case of real examination, it is important to precisely represent the patient's organ structures. For the purpose of the simulator, the simpler approach (reduced accuracy) is satisfactory. The phenomenon of wave refraction and diffraction is not taken into account, as the speed of sound in various structures of the heart is very similar [17]. It can be assumed in the case of echocardiography. The simulator uses a ray-casting technique that allows to achieve the effect of the attenuation and the acoustic shadow (Figure 2). More accurate techniques, such as wave field modeling [9] or ray-tracing [20] are much more time consuming.

2.3 Principles of Web-Based Version of the CT2TEE Simulator

The web-based version of the simulator provides an interactive graphical interface for the clients, but the projections are downloaded from the CT2TEE Internet server. This version of the simulator is available on the Internet at the website `www.ct2tee.agh.edu.pl` [21].

The model for simulation consists of CT slices, and it cannot be transmitted to clients because it takes up a big amount of data. It is loaded into server memory at the start of CT2TEE application. For each web request a new projection is calculated. This process is described in Figure 3 [22].

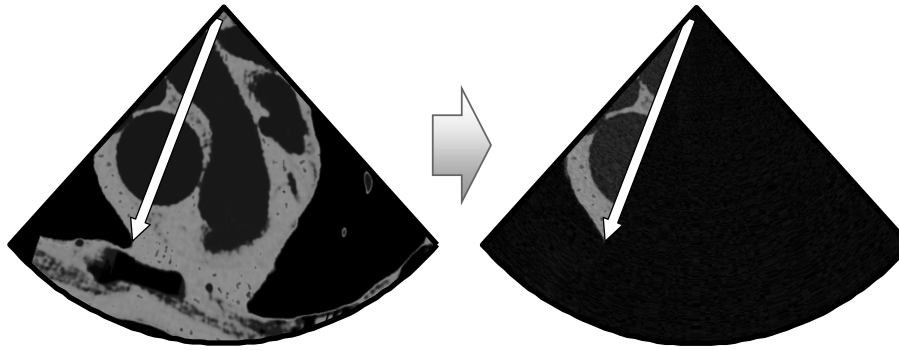

Figure 2. Using ray-casting technique – from a projection (left), to simulated US (right)

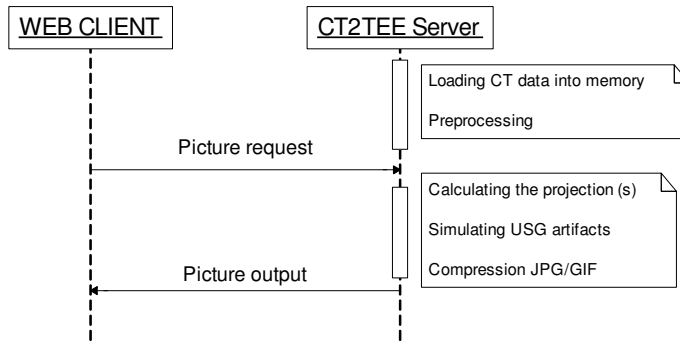

Figure 3. Diagram for a request processing by CT2TEE web server

It was considered that the interface reaction times (100 ms for the standalone version and 200 ms for the web-based version) are sufficient. To achieve a smooth animation effect without flicker, the time of projection calculation should be shorter than 100 ms and a reserve time for data transmission via Internet or intranet should be guaranteed. Although there is the possibility to use GPU processing for web requests [23], the component environments deliver the portability of the code, while the code for the graphics card is rather associated with a specific device.

3 IMPLEMENTATION OF PARALLEL PROCESSING

The time-consuming calculations are the important issues of medical informatics and bioinformatics [24]. Nowadays some real-time computational problems, which previously required large processing power of super or distributed computing, can be solved with highly parallel algorithms running on Graphical Processing Units (GPU). However, to apply techniques, the problems have to be separated to a number of short operations (they create so-called processing pipeline) that are exe-

cuted in single instruction, multiple data (SIMD) manner. The visualization of large (512^3 voxels or larger) medical volumetric data is solved in this way commonly [25, 26]. In the same manner the ultrasound simulation is parallelized, e.g., Shams et al used SIMD instruction set [10].

Daoud and Lacefield propose one-dimensional decomposition of the 3D simulation along z-axis, that minimizes inter-node communication [11]. There are methods presented for parallel processing of single scan line or multiple scan lines. Gjerald et al. propose division of the extraction and sampling of scatterers into a large number of independent processes [14].

To implement the parallel processing a division of calculations into a number of parallel tasks is needed. The domain decomposition technique can be used. The first approach is to divide input data into two orthogonal spaces separated by a vertical line [28]. It is possible to calculate the projection and ray-casting for these halves independently by two threads. This case is illustrated in Figure 4.

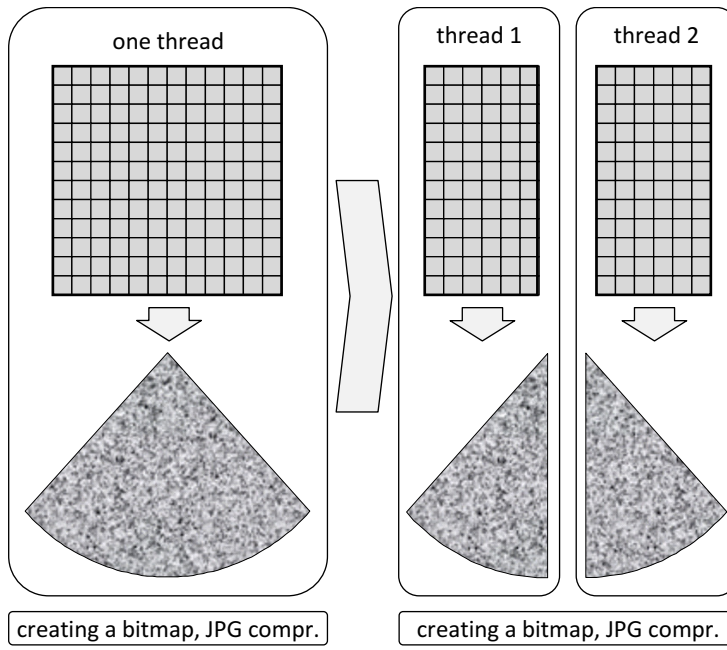

Figure 4. Dividing simulation processing into two threads

The general case of division into many (N) parts in this way is not possible because calculating parts of projection are rectangles and parts of ray-casting are pieces of a circle (slices) – ranges of data for these two steps of simulation processing do not match/overlap. Parallelization of calculating projection can use domain decomposition technique for Cartesian system of data [29]. The ray-casting method uses the polar coordinate system; therefore, decomposition should be processed in

the other way – according to polar coordinate system. The proper approach is to implement parallelization in two stages – the first stage is to parallelize projection calculations and the second – to parallelize ray-casting (Figure 5). Joining data between stages (synchronization) is required.

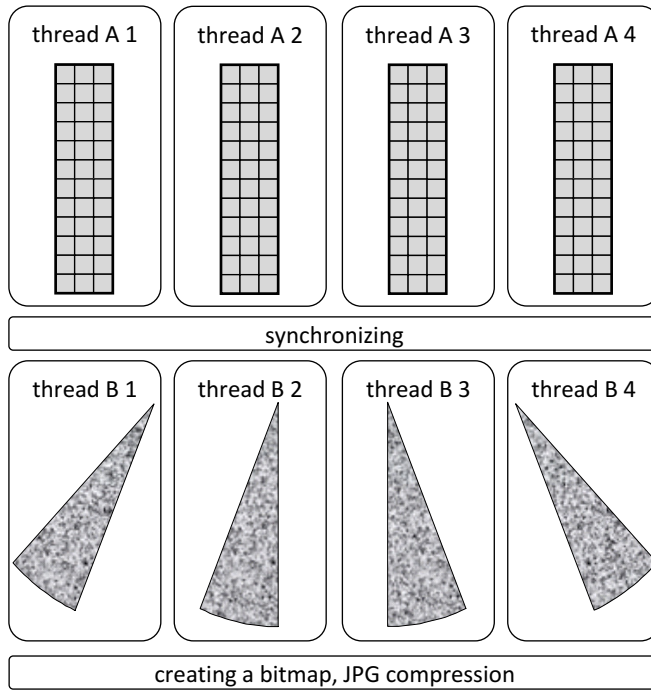

Figure 5. Dividing simulation processing into more than two threads

The parallel computing technique is characteristic of a programming environment. Most parallel implementations, including medical applications, are based on parallelization of loops [27]. The CT2TEE project is written in Csharp language and compiled for the .NET Framework 2.0. The latest versions of the .NET Framework (4.0, 4.5) provide a new extension for parallel programming – the Parallel class, that delivers a mechanism of parallel loops (Parallel.For). Unfortunately, Parallel.For is implemented only for integers. Ray-casting is implemented as a loop of floating point numbers, which is related to the angle of a ray, measured in radians. Precision of loop iteration is determined by a view magnification and is variable. Therefore, the best way to implement parallelization for .NET Framework 2.0 is to base on the method presented in [30]. A sample code is shown below.

Part of code for parallelizing of projection calculating

```
[...]
int inclLowerBound = 0;
```

```
int exclUpperBound = size_x;
int size = exclusiveUpperBound - inclusiveLowerBound;
int range = size / number_of_processors;

for (int p = 0; p < number_of_processors; p++)
{
    int start = p * range + inclLowerBound;
    int end = (p == number_of_processors - 1)
           ? exclUpperBound : start + range;
    ThreadPool.QueueUserWorkItem(              // (A1)
        delegate
        {
            int x, y;
            for (x = start; x < end; x++)      // (A2)
                for (y = 0; y < size_y; y++)
[...]
```

The presented code implements a projection calculation, using a pool of threads (A1). The domain decomposition is realized in the line (A2), where for each thread a part of a domain (columns from *start* to *end*) is assigned. The step is of 1 point. In case of ray-casting the loops should use floating point iterators (angle of a ray), therefore the code should be modified as shown on the listing below. The altered part of the code (floating point number iteration) is marked as the line (B1).

Part of code for parallelizing of ray-casting

```
delegate
{
    double angle;
    [...]
    for (angle = start; angle < end; angle += 0.001)  // (B1)
    [...]
```

4 PARALLEL PROCESSING PERFORMANCE

To assess the performance of the proposed solutions, special tests were carried out. A high performance server was used: IBM Blade HS 21, CPU: 2.0 GHz, Intel Xeon (8 cores), RAM 16 GB. There were Windows 2003 Server 64 bit and .NET Framework 2.0 installed on the server. The time measurement techniques (System.Diagnostics) allowed to provide sufficient measurement accuracy of 1 ms. The processing time values were very stable due to the use of a special server hardware and equipment, previously described.

The results of the tests are shown in Table 1. Times of view generation are presented in the form of a chart (Figure 6). The next chart (Figure 7) shows the speedup (SU) of parallelization [31, 32], calculated according to Equation (3).

$$SU_N = \frac{t_{serial}}{t_N} \tag{3}$$

To check a parallelization opportunities (and quality of the solution), the proportion of a program that can be executed in the parallel way is estimated according to Equation (4).

$$P_{estim} = \frac{\frac{1}{SU} - 1}{\frac{1}{N} - 1} \tag{4}$$

The P_{estim} values are very similar – small differences are caused by measurement error.

CPU number	t_{min} [ms]	SpeedUp (SU)	P_{estim}
(S)erial	83	1.00	–
1	84	0.99	–
2	48	1.73	0.84
3	36	2.31	0.85
4	30	2.77	0.85
5	27	3.07	0.84
6	24	3.46	0.85
7	22	3.77	0.86
8	20	4.15	0.87

Table 1. The performance of parallelization

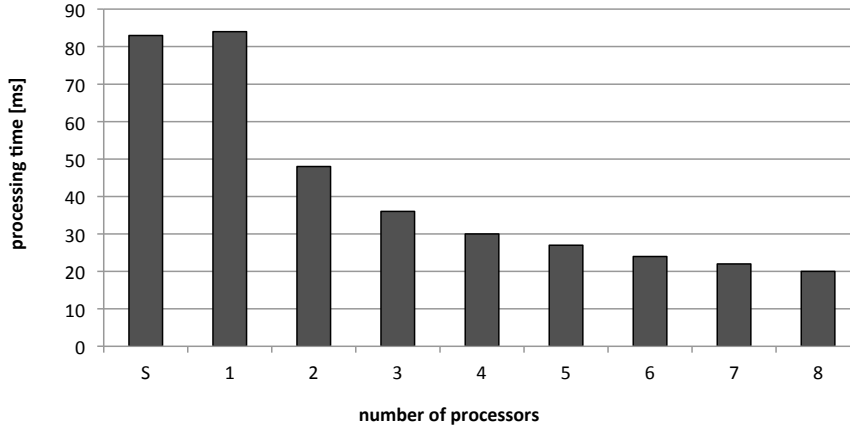

Figure 6. Processing times for parallelized simulation

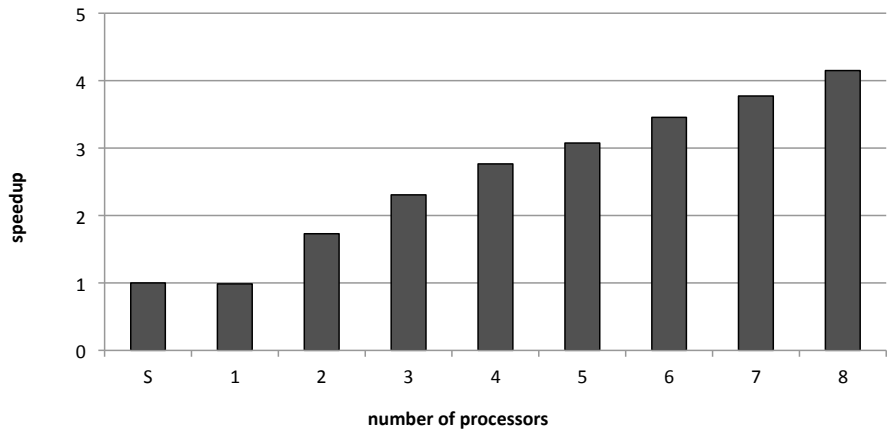

Figure 7. Speedups for parallelized simulation

5 PARALLEL PROCESSING FOR ANIMATED PROJECTIONS

The dynamic version of CT2TEE web application should provide an animated view of heart projection. In the case of web-based version it can be implemented in two ways:

- as a client-side application, that downloads separate frames and controls viewing (in Flash, JavaScript, etc.),
- as a web site, that downloads an animated picture (GIF).

In the first case the client-side application downloads subsequent frames in a proposed order:

- frame 1 – (show: 1),
- frame 3 – (show cycle: 3, 1),
- frame 2 – (show cycle: 3, 1),
- frame 4 – (show cycle: 1, 2, 3, 4).

This method provides faster response time for the first image, but multiple network requests make full animation is achieved after a long time.

In the second case, the web server can prepare an animated projection on client demand. This case is easier to implement on the client side and is more reliable, but on the other hand – the number of colors is reduced to 256 (e.g. grayscale palette, color layers unavailable [33]) and a request takes more time. To achieve a smooth animation effect, the calculations should be optimized.

The simplest way to implement an animated view is to generate four projections in the described way (Figure 5) and to join them. It can take $4 \times 20\,\text{ms} = 80\,\text{ms}$ (using presented environment – 8 CPUs).

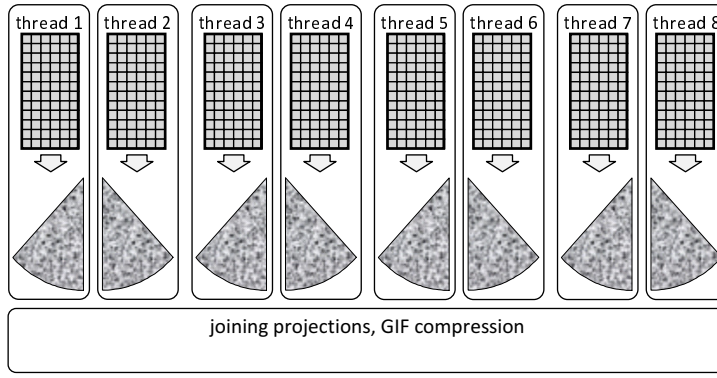

Figure 8. Parallel processing of animated projection (GIF format)

The other way is to divide computations of each frame into a couple of CPUs, and process a concurrent simulation of four frames (Figure 8). This method takes 47 ms for the simulation and is faster. Additional time is required for the frames compression to GIF format (6 ms per frame, 24 ms for the whole animation). It can be optimized by using a separate GIF compression for each frame (Figure 9), and is possible because the animated GIF format (proposed by Netscape) consists of a header and blocks of independent GIF images as subsequent frames.

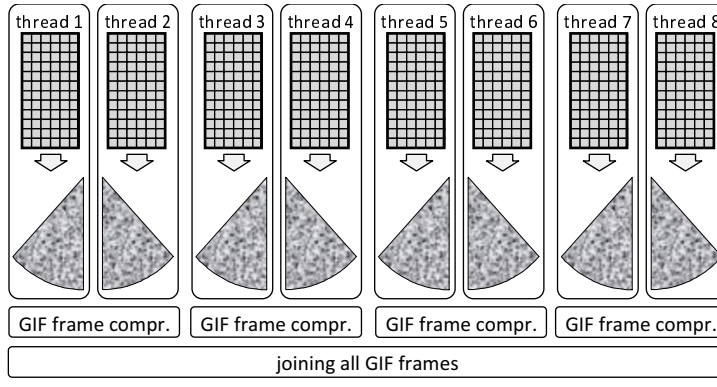

Figure 9. Optimized parallel processing of animated projection (GIF format)

6 CONCLUSIONS

This paper shows advantages of parallelization of ultrasound simulation. Projection generation times in a standalone or web-based CT2TEE application with the static model were highly reduced from 84 ms to 20 ms when using eight processors. It

makes the simulation very smooth in the case of standalone application. In the case of web-based application the times of image delivery are highly reduced and smooth animation is enabled. The parallel part of the code execution was estimated as 85 %. The optimized methods of animated GIF images are presented.

The future work involves an implementation of transthoracic echocardiography simulator (CT2TTE). The next problem is to optimize web application for concurrent requests and to reduce resource consumption. Another issue is to check the relation between request times and throughput in the case of CT2TEE web server [22]. A further consideration of the current work will be performance analysis and parallelization of ray-tracing techniques in echocardiography simulations.

REFERENCES

[1] Kitowski, J.—Alda, W.—Boryczko, K.—Bubak, M.—Dzwinel, W.—Funika, W.—Moscinski, J.—Nikolow, D.—Pogoda, M.—Slota, R.—Wcislo, R.: On Computational and Computer Methods for Simulation Problems in Model Systems. Proceedings of the Third International Conference on Parallel Processing and Applied Mathematics (PPAM '99), pp. 473–488.

[2] Izworski, A.—Koleszynska, J.—Tadeusiewicz, R.—Bulka, J.—Wochlik, I.: GIGISIM (Glucose-Insulin and Glycemic Index Web Simulator) – The Online System Supporting Diabetes Therapy. Proceedings of the IASTED International Conference on Telehealth, July 19–21, Banff, AB, Canada 2005, pp. 80–83.

[3] Izworski, A.—Koleszynska, J.—Tadeusiewicz, R.: Educational Simulators – Compliance with the Requirements of Diabetes Patients and Diabetes Therapy Guidelines. Proceedings of ICEIS 2007, Portugal 2007, pp. 319–322.

[4] Jerath, A.—Vegas, A.—Meineri, M.—Silversides, C.—Feindel, C.—Beattie, S.: An Interactive Online 3D Model of the Heart Assists in Learning Standard Transesophageal Echocardiography Views. Canadian Journal of Anesthesia, Vol. 58, 2011, No. 1, pp. 14–21.

[5] Piórkowski A.—Werewka, J.: A Concept of eTraining Platform for Cardiology Learning based on SOA Paradigm. Proceedings of 14th International Conference on Enterprise Information Systems (ICEIS 2012), pp. 261–264.

[6] Weidenbach, M.—Drachsler, H.—Wild, F.—Kreutter, S.—Razek, V.—Grunst, G.—Ender, J.—Berlage, T.—Janousek, J.: EchoComTEE – A Simulator for Transoesophageal Echocardiography. Anaesthesia, Vol. 62, 2007, pp. 347–53.

[7] Bose, R.—Matyal, R.—Panzica, P.—Karthik, S.—Subramaniam, B.—Pawlowski, J.—Mitchell, J.—Mahmood, F.: Transesophageal Echocardiography Simulator: A New Learning Tool. J. Cardiothorac Vasc Anesth, Vol. 23, 2009, pp. 544–548.

[8] Kempny, A.—Piórkowski, A.: CT2TEE – A Novel, Internet-Based Simulator of Transoesophageal Echocardiography in Congenital Heart Disease. Kardiologia Polska, Vol. 68, 2010, pp. 374–379.

[9] Jensen, J. A.—Fox, P. D.—Wilhjelm, J. E.—Taylor, L. K.: Simulation of Non-Linear Ultrasound Fields. Proceedings of IEEE Ultrasonics Symposium 2002, Vol. 2, pp. 1733–1736.

[10] Shams, R.—Hartley, R.—Navab, N.: Real-Time Simulation of Medical Ultrasound from CT Images. Medical Image Computing and Computer-Assisted Intervention (MICCAI 2008), Springer, Berlin Heidelberg 2008, pp. 734–741.

[11] Daoud, M. I.—Lacefield, J. C.: Parallel Three-Dimensional Simulation of Ultrasound Imaging. In Proceedings of 22nd International Symposium on High Performance Computing Systems and Applications, IEEE 2008, pp. 146–152.

[12] Kutter, O.—Shams, R.—Navab, N.: Visualization and GPU-Accelerated Simulation of Medical Ultrasound from CT Images. Computer Methods and Programs in Biomedicine, Vol. 94, 2009, pp. 250–266.

[13] Reichl, T.—Passenger, J.—Acosta, O.—Salvado, O: Ultrasound Goes GPU: Real-Time Simulation Using CUDA. Med. Imaging, 2009: Visualization, Image-Guided Procedures, and Modeling, Vol. 7261, 2009.

[14] Gjerald, S. U.—Brekken, R.—Hergum, T.—D'hooge, J.: Real-Time Ultrasound Simulation Using the GPU. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, Vol. 59, 2012, No. 5, pp. 885–892.

[15] Skurski, A.—Borzecki, M.—Balcerzak, B.—Kaminski, M.—Napieralski, A.—Kasprzak, J. D.—Lipiec, P.: Image Processing Methods for Diagnostic and Simulation Applications in Cardiology. International Journal of Microelectronics and Computer Science, Vol. 3, 2012, No. 4, pp. 146–151.

[16] Szostek, K.—Piórkowski, A.—Kempny, A.—Banyś, R.—Gackowski, A.: Using Computed Tomography Images for a Heart Modeling. Journal of Medical Informatics and Technologies, Vol. 19, 2012, pp. 75–84.

[17] Piórkowski, A.—Kempny, A.: The Transesophageal Echocardiography Simulator Based on Computed Tomography Images. IEEE Transactions on Biomedical Engineering, Vol. 60, 2013, No. 2, pp. 292–299.

[18] Zhu, M.—Salcudean, S. E.: Real-Time Image-Based B-Mode Ultrasound Image Simulation of Needles Using Tensor-Product Interpolation. Transactions on Medical Imaging, Vol. 30, 2011, No. 7, pp. 1391–1400.

[19] Piórkowski, A.—Jajesnica, L.—Szostek, K.: Creating 3D Web-Based Viewing Services for DICOM Images. In: Kwiecien, A., Gaj, P., Stera, P. (Eds.): 16th Conference on Computer Networks (CN 2009), Poland 2009, CCIS, Vol. 39, Springer 2009.

[20] Szostek, K.—Lesniak, A.: Parallelization of the Seismic Ray Trace Algorithm. Parallel Processing and Applied Mathematics (LNCS, Vol. 7204), Springer 2012, pp. 411–418.

[21] CT2TEE Project Home Page – Web-Based Transoesophageal Echocardiography Simulator. Availaible on: `http://www.ct2tee.agh.edu.pl`.

[22] Piórkowski, A.—Kempny, A.—Hajduk, A.—Strzelczyk, J.: Load Balancing for Heterogeneous Web Servers. 17th Conference on Computer Networks (CN 2010), CCIS 79, Springer 2010, pp. 189–198.

[23] Szostek, K.—Piórkowski, A.: OpenGL in Multi-User Web-Based Applications. Innovations in Computing Sciences and Software Engineering, Springer, 2010, pp. 379–383.

[24] Orzechowski, P.—Boryczko, K.: Parallel Approach for Visual Clustering of Protein Databases. Computing and Informatics, Vol. 29, 2010, pp. 1221–1231.

[25] Hachaj, T.—Ogiela, M. R.: Visualization of Perfusion Abnormalities with GPU-Based Volume Rendering. Computers and Graphics, Vol. 36, 2012, No. 3, pp. 163–169.

[26] Hachaj, T.—Ogiela, M. R.: Framework for Cognitive Analysis of Dynamic Perfusion Computed Tomography With Visualization of Large Volumetric Data. Journal of Electronic Imaging, Vol. 21, 2012, No. 4, 043017 (Nov 30, 2012), doi: 10.1117/1.JEI.21.4.043017.

[27] Schellmann, M.—Gorlatch, S.—Meilander, D.—Kosters, T.—Schafers, K.—Wubbeling, F.—Burger, M.: Parallel Medical Image Reconstruction: From Graphics Processing Units (GPU) to Grids. The Journal of Supercomputing, Vol. 57, 2011, No. 2, pp. 151–160.

[28] Emmanuel, M.—Ramesh Babu, D. R.—Jagdale, J.—Game, P.—Potdar, G. P.: Parallel Approach for Content Based Medical Image Retrieval System. Journal of Computer Science, Vol. 6, 2010, No. 11, pp. 1258–1262.

[29] Ino, F.—Kawasaki, Y.—Tashiro, T.: A Parallel Implementation of 2-D/3-D Image Registration for Computer-Assisted Surgery. International Journal of Bioinformatics Research and Applications, Vol. 2, 2006, No. 4, pp. 341–358.

[30] Toub, S.: Patterns of Parallel Programming. Understanding and Applying Parallel Patterns with the .Net Framework 4 and Visual Csharp. Parallel Computing Platform Microsoft Corporation. Version February 16, 2010.

[31] Amdahl, G.: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. AFIPS Conference Proceedings, Vol. 30, 1967, pp. 483–485.

[32] Rodgers, D. P.: Improvements in Multiprocessor System Design. ACM SIGARCH Computer Architecture News archive, New York, ACM, Vol. 13, ISSN 0163-5964, 1985, pp. 225–231.

[33] Piórkowski, A.: Construction of Educational Color Layer for Echocardiography Simulator. Journal of Medical Informatics and Technologies, Vol. 16, 2010, pp. 167–171.

Adam Piórkowski received his Ph. D. degree in computer science in 2005 from the AGH University of Science and Technology. Currently he works for the Department of Geoinformatics and Applied Computer Science at the Faculty of Geology, Geophysics and Environmental Protection. His research interests include real-time systems, scheduling, databases, component technologies and medical imaging.