Biomedical Signal Processing and Control 35 (2017) 8–18

Contents lists available at ScienceDirect

Biomedical Signal Processing and Control

journal homepage: www.elsevier.com/locate/bspc

An efficient and fast GPU-based algorithm for visualizing large volume of 4D data from virtual heart simulations

Shui Yu[a], Shanzhuo Zhang[a], Kuanquan Wang[a,*], Yong Xia[a], Henggui Zhang[b]

[a] *Research Center of Perception and Computing, School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China*
[b] *School of Physics and Astronomy, University of Manchester, Manchester, United Kingdom*

ARTICLE INFO

Article history:
Received 9 March 2016
Received in revised form 10 January 2017
Accepted 28 January 2017

Keywords:
Time-varying visualization
Vector quantization
Ray casting
Cardiac electrophysiology
Virtual heart

ABSTRACT

An efficient and fast visualization algorithm is important for analyzing a large volume of physical data conformed to 3D anatomical geometry that evolves with time (i.e., 4D data). Such 4D visualization helps to study the evolution of cardiac excitation waves in normal and pathological conditions and understand the mechanisms underlying the genesis and maintenance of cardiac arrhythmias. However, due to limited hardware resources, so far we have not found any report about real time methods to visualize a large volume of 4D data of virtual heart simulation data. In this study, we propose a GPU-based method to address this issue, our method consists of two phases, and the first is the data compression phase, implementing an improved hierarchical vector quantization method with N-nearest neighbor searching strategy in GPU, which reduces compression time dramatically. In the second phase, the compressed data is directly decompressed in GPU and rendered with ray casting method. What is more, an adaptive sampling strategy and empty space skipping methods are further used to accelerate the rendering process, resulting in a high rendering speed. The proposed method has been evaluated for the visualization of large time-varying cardiac electrophysiological simulation data by using our simulation datasets and has achieved promising results. For about 27G bytes dataset, our method can render the data with above 35 frames per second (FPS), which exceeds the real-time frame rate for interactive observing. It significantly decreases the time in the compression phase and achieves real time rendering speed with high image quality in the visualization phase, which demonstrates the accuracy and efficiency of our method.

1. Introduction

Virtual physiological heart provides a powerful computational platform to simulate and analyze the underlying mechanisms for normal and abnormal cardiac electrical activities [1]. For example, Multi-scale heart simulation and visualization methods have been used to study pathological mechanisms of cardiac arrhythmias by analyzing the evolution of re-entrant electrical excitation waves in the heart [2]. In these studies, due to large volume of physical data in 3D space, which evolves with time (i.e., 4D data), it is still a challenge to visualize dynamical cardiac excitation waves with high quality in an efficient and interactive way. As the size of the 4D data (such as heart data) may be up to dozens of gigabytes, it is hard to store them in the RAM or the video memory, data transmission between the RAM and the video memory is time consuming, which makes it difficult for a rapid manipulation (i.e., "real-time") of the visualization. Some out-of-core methods, such as volume compression methods [3–6], can solve some of the storing and transferring problems, but most of these methods are still time-consuming, which may take minutes or hours to complete the tasks of compression. Furthermore, such compressed data cannot be decompressed directly in graphics processing unit (GPU), which seriously restricts rendering speed, so they need to make a necessary trade-off between the image quality and the speed.

Visualization methods have been applied in heart models for several years. A 3D human atrial model and the corresponding visualization method were used by Colman et al. [7] to simulate and analyze how re-entrant excitation waves degenerate into persistent and erratic wavelets underlying atrial fibrillation. Aslanidi et al. [8] used 2D and 3D models to investigate how the frequency, wavelength, meandering pattern and stability of the re-entrant spiral waves evolve with time in cardiac tissue. Their works presented insights into underlying possible mechanisms of cardiac arrhythmias, but their 3D visualization results were presented only

[*] Corresponding author.
E-mail addresses: yushi@hit.edu.cn (S. Yu), shanzhuo.zhang@gmail.com (S. Zhang), wangkq@hit.edu.cn (K. Wang), xiayong@hit.edu.cn (Y. Xia), h.zhang-3@manchester.ac.uk (H. Zhang).

http://dx.doi.org/10.1016/j.bspc.2017.01.015
1746-8094/© 2017 Elsevier Ltd. All rights reserved.

as discrete snapshots of animation, which normally took a large amount of computing time, limiting an efficient and interactive visualization of simulations. To our best knowledge, among the intense research of cardiac visualization methods, few of them focused on the issue of 4D visualization problems. For example, Wang et al. [9] presented a simplified LH histogram-based transfer function design method to interactively visualize multi-boundary and electrophysiology simulation data, but this method was based on CPU, which cannot be implemented interactively with simulation courses. Zhang et al. [10] provided a visualization method to display dynamic 4D real-time MDCT images and later expanded the system to 4D cardiac MRI and ultrasound images [11], but the data size were relatively small as compared to the large volume data of whole heart simulations.

With the great advance of graphic hardware technology, GPU-based methods for data visualization have been proposed. Garcia et al. [4] introduced a GPU-based algorithm for reconstructing 3D wavelets using fragment programs and tile-boards to distribute the 3D wavelet coefficients. Wetekam et al. [12] used an FPGA decoder to decompress wavelet coefficients based on Huffman variable bit decoding; Villa et al. [13] proposed a hardware-accelerated pipeline to resample the slice using 3D texture. Lamas-Rodriguez et al. [14] presented a brick-based GPU-accelerated method in which bricks were decompressed and rendered with a specific level of resolution depending on their distances to the eye point. The method was a GPU implementation of Ihm and Park's classic wavelet-based rendering [15]. Transforming domain approaches stated above are kind of pre-defined models [16], and are often computationally economic, but the disadvantage of these approaches is that the compressed data cannot be decompressed or rendered directly in GPU because of the sophisticated decoding process, that is a large waste of the strong computational capability of the hardware.

Vector quantization (VQ) is another strategy to compress volume data. The idea of VQ is to divide the space into small blocks and then represent these blocks with a set of vectors named codewords or dictionary. VQ was first introduced in volume rendering by Paul et al. [17]. Schneider et al. [5] proposed a hierarchical VQ method (HVQ) with three hierarchical levels and a dictionary of 1024 elements. The learning of the codebooks is performed through a modification of the Linde–Buzo–Gray (LBG) algorithm [18]. Fout et al. [19] combined HVQ with a previous domain transformation. Some methods based on sparse coding have been proposed in recent years which can be seen as a generalization of VQ [20], Gobbetti et al. [21] compactly described the small blocks of datasets by sparse linear combinations of prototype blocks stored in an over-complete dictionary, applying the concept of corset [22,23]. VQ-based methods are very popular in the papers of volume rendering since the decompression is extremely fast and can be done directly on GPU. However, a main problem of these methods is the computationally expensive cost in pre-processing massive volumes for obtaining a good codebook.

In recent years, tensor approximation (TA) has been applied to interactively visualize large volume data, such as the hierarchical brick-tensor composition method and the tensor-specific quantization strategy developed by Suter et al. [6,24]. However, the reconstructive costs in GPU are very high and real time reconstruction is feasible only for small blocks. Other methods treat the 4D data as a 3D video [25], which means combining the spatial compression with time compression. For example, Jang et al. [26] used the functional representative approach for time-varying data, encoding the data by utilizing temporal similarities of the evolving datasets. Wang et al. [27] built a hierarchical structure for each time step, then grouped and encoded space-corresponding nodes together. This method is very easy to implement and each frame can be encoded separately, but it cannot exploit the space coherence of the data since the data is separated by time step.

In this paper, we present a solution to tackle the aforementioned challenges, i.e., limited hardware resource to store the GB bytes of volume data, time-consuming compression process, not real-time visualization and poor image quality, for visualization of the 4D cardiac electrophysiological simulation data in a dynamic style, so that the data can be interpreted in an interactive way (for example users can explore the 4D object from any viewing angles, and visualize the inside volume by simply rotating or changing the opacity of any parts of the target). The solution consists of two main aspects: one is to compress the data rapidly and the other is to develop a method to render the compressed data directly in GPU without loss of image quality and rendering speed.

Specifically, in the compression phase, a N-nearest searching method and a GPU-based compression architecture were proposed to speed up the VQ and encoding processes. In the rendering phase, some techniques were employed to accelerate the rendering process. Firstly, we expanded the hierarchical data structure to skip the empty blocks in the rendering stage. Secondly, an adaptive sampling strategy was presented to find a valid block as fast as possible, avoiding the redundant computations. Third, an advanced feature, texture object in CUDA (Compute Unified Device Architecture, a parallel computing platform and programming model which enables dramatic increases in computing performance by harnessing the power of the GPU) [28], was employed to control the rendering flow flexibly. We also tried to render the compressed data with the ray casting method and tri-linear interpolation by taking advantage of these methods.

2. Methods

2.1. Overview

For real-time rendering of time-varying volumetric data sets, it is essential to implement a compressed representation of the data. In fact, due to the bandwidth at various stages of the pipeline, it is often not feasible to interactively load, decode and render data at each time step in GPU in real time. To overcome this problem, we propose a new method with architecture shown in Fig. 1. First, the volume is decomposed into hierarchical structure, which consists of two sets of vectors and values representing the information of all sub-blocks. Second, the vector sets are quantized into a set of codebooks and indexed volumes, which are saved in hard disk. These two steps are conducted offline.

At running time, compressed data, such as codebooks and indexed volumes are loaded from hard disk to GPU memory, and in turn are mapped as 2D or 3D textures. These textures are managed as texture objects; texture object is an advanced feature in CUDA. The main program running in CPU transfers the IDs of the needed textures to CUDA kernel function, and the corresponding codebooks and indexed volumes are selected to be used as the current working sets. Codewords and mean values are then fetched from the current working set through the coordinates needed in the CUDA kernel function. After the process of ray casting, all the results are written to frame buffer and then output by OPENGL as images.

In the following sections, we will first introduce how to generate the simulation data from the model of virtual heart, and then describe the novel methods we have used in a sequential manner.

2.2. Electrophysiological Simulation Data

The data used in this paper consist of two different parts. One represents the anatomic structure of human ventricles, and the other represents the electrophysiological simulation data of action potentials through the human ventricles.