DYNAMIC PARTICLE SYSTEM ON THE GPU

Mark Kim, Guoning Chen and Charles Hansen

Introduction

Extracting isosurfaces represented as high quality meshes from three-dimensional scalar fields is needed for many important applications, particularly visualization and numerical simulation. One recent advance for extracting high quality meshes for isosurface computation is based on a dynamic particle system. Unfortunately, this state-of-the-art particle placement technique requires a significant amount of time to produce a satisfactory mesh. To address this issue, we utilize the parallelism property in the particle placement and combine it with the CUDA implementation, a parallel programming technique on the GPU, to significantly improve the performance. We have applied our GPU based particle placement to a number of data from bioengineering where particle system is frequently used to generate isosurface meshes for simulations. Our results show comparable quality to the meshes generated using conventional CPU based particle system with at least ten fold speed up for most data.

Methods

Initially, a distance field and a sizing field are pre-computed to represent the isosurface as an implicit function, F, and to encode the distance between points on F, respectively. Next, particles are seeded on the isosurface based on the results of marching cubes. Then, the particles are processed sequentially: determine neighbors, compute energy and velocity, and update position. A particle only moves if the new position has lower energy. Once every particle has been processed, the density of the particles are checked to delete or add particles. To speed-up the system, we bin the space to reduce the neighborhood search.

Although the GPU has more processing power than the CPU, it also has limitations. In particular, the GPU is a massively parallel system with many hardware threads. Unfortunately these hardware threads do not handle divergence well, where control statements (if, switch, do, for, while) may cause threads to follow different execution paths which serializes the computations. We chose to parallelize the particle system as two levels. First, bins are run concurrently. Second, the energy and velocity computations are parallelized as well.

Results

First, the quality of a mesh is determined by calculating the radius ratio between the inscribed and the circumscribed circles of the triangles on the mesh. For the ribcage data set, the mean radius ratio of the CPU version is 0.912863 while the mean radius ratio of the GPU version is 0.914975. This means the CPU mesh and GPU mesh have very good triangles, i.e. equilateral triangles. Further, the GPU version produces a mesh of similar quality to the CPU version. Second, although both meshes are of similar quality, the GPU version is 44x faster producing a mesh with approximately 500,000 particles than the CPU version, where the CPU takes 19750.2 seconds while the GPU takes 445.49 seconds. Further, as the particle count increases, so does the speed-up, from 23x for approximately 320k particles to 44x speed-up for approximately 530k particles.

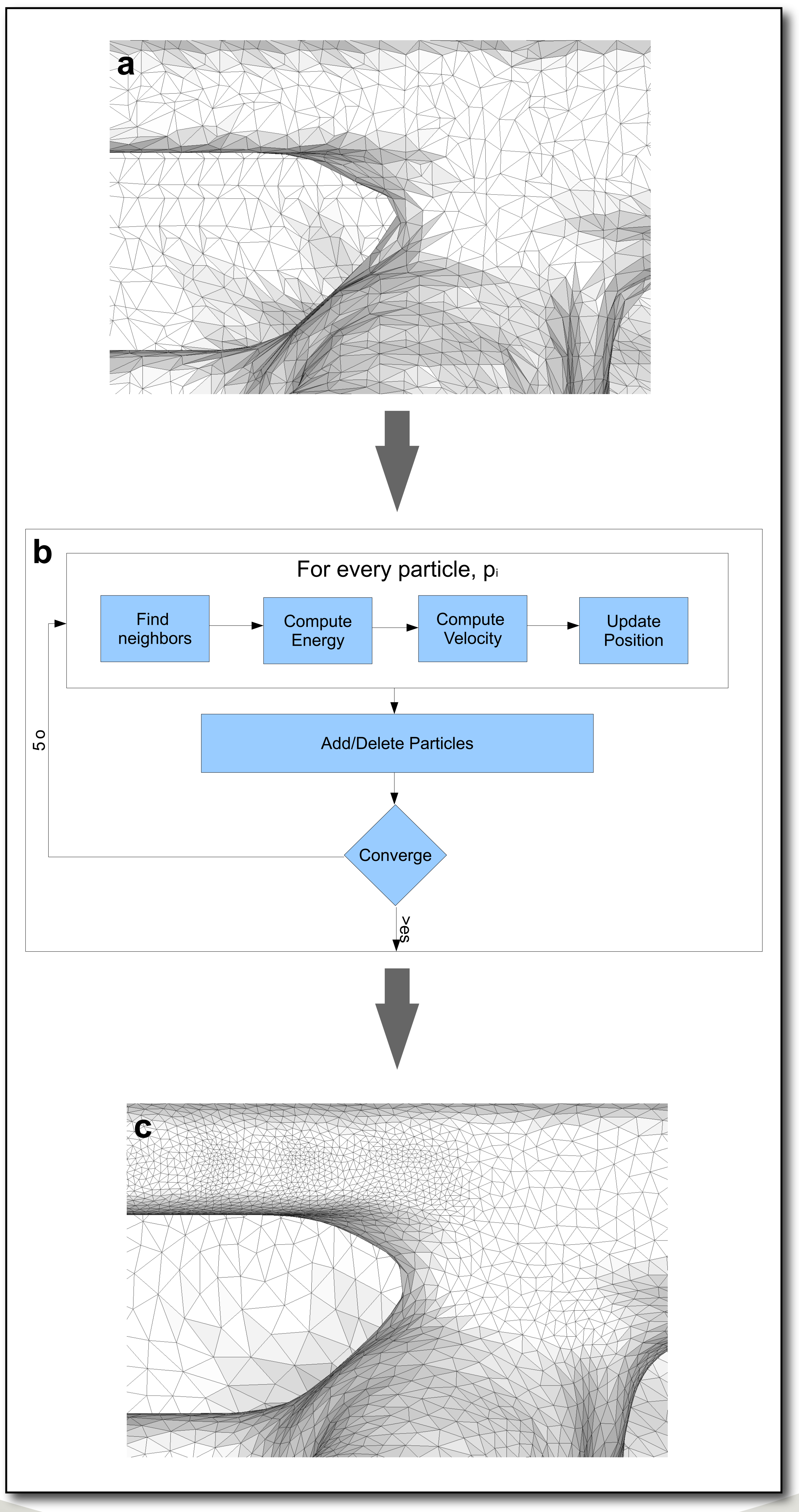
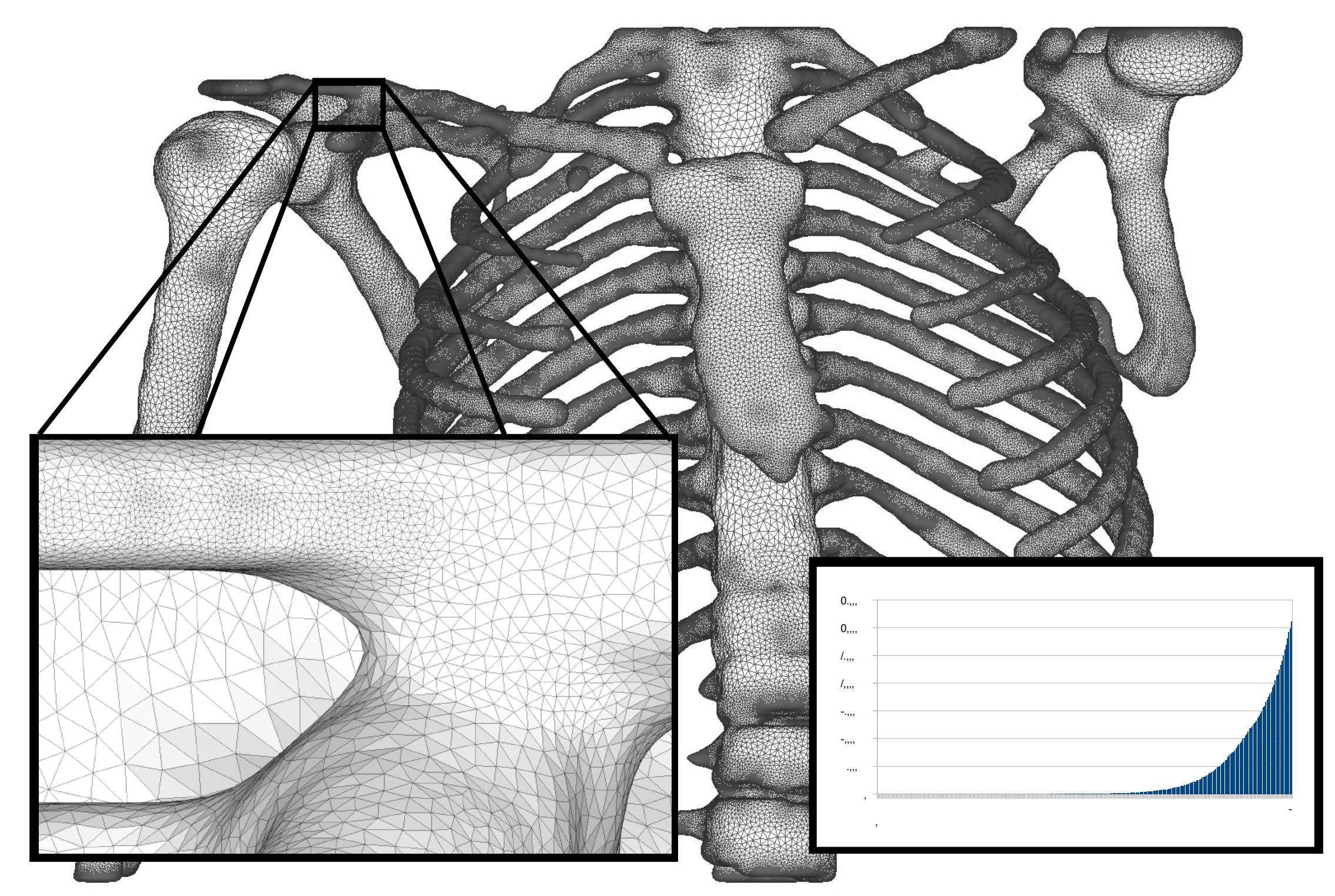
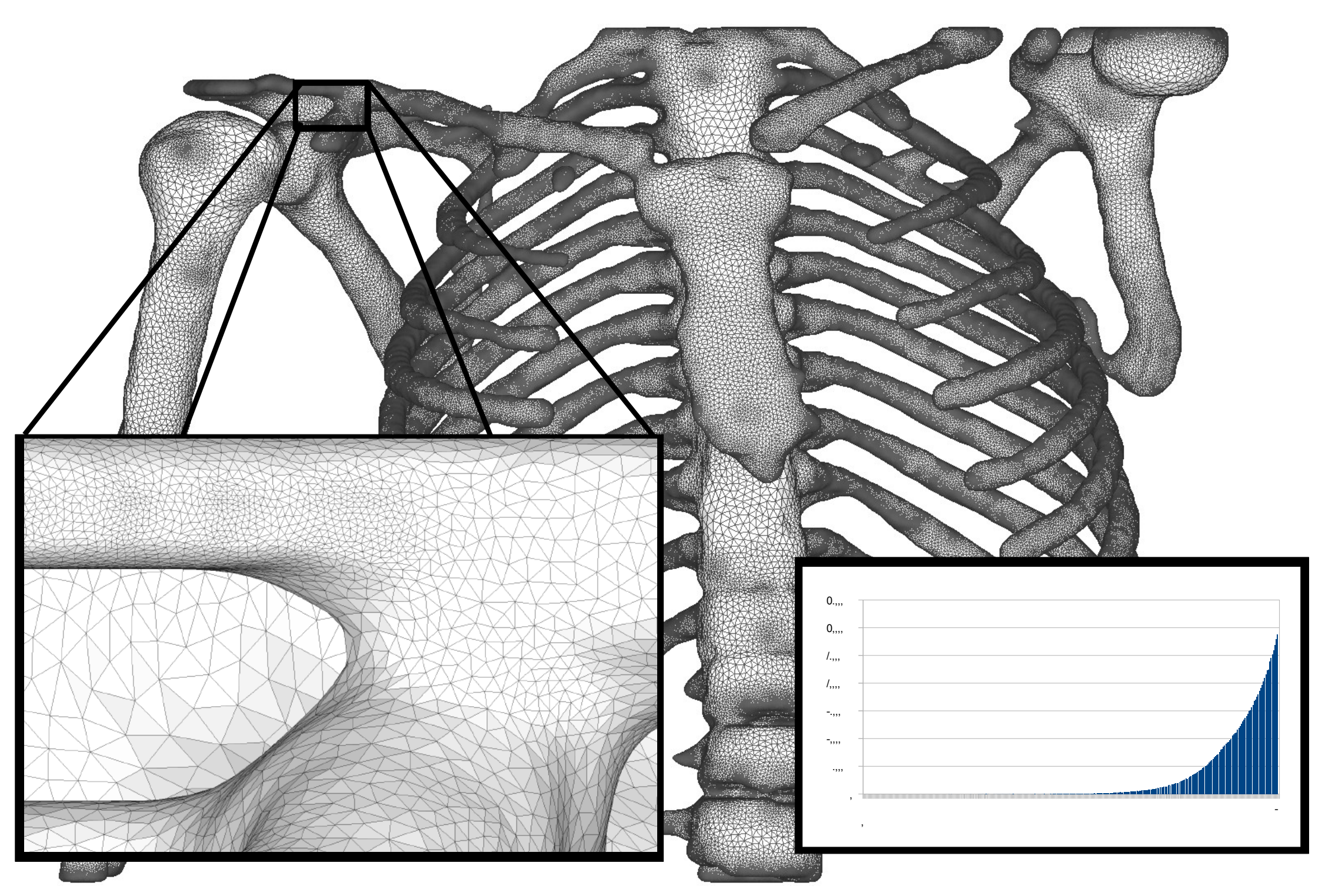


Figure 1. Particle System overview. (a) The system is initialized by placing particles on the surface. (b) Move all the particles until the energy reaches an ideal state. (c) The finished mesh.



(a) A ribcage data set with zoomed image and histogram of the radius ratio. This mesh was generated on the GPU.



(b) A ribcage data set with zoomed image and histogram of the radius ratio. This mesh was generated on the CPU.

Figure 2. Images of ribcage data set, GPU and CPU, respectively. Further, embedded is a zoomed in area for each image and the histogram for the data sets. The visual quality of the CPU implementation compared to the GPU implementation is very similar across the data sets. The histograms show that both the CPU and GPU systems are dominated by well-shaped triangles.

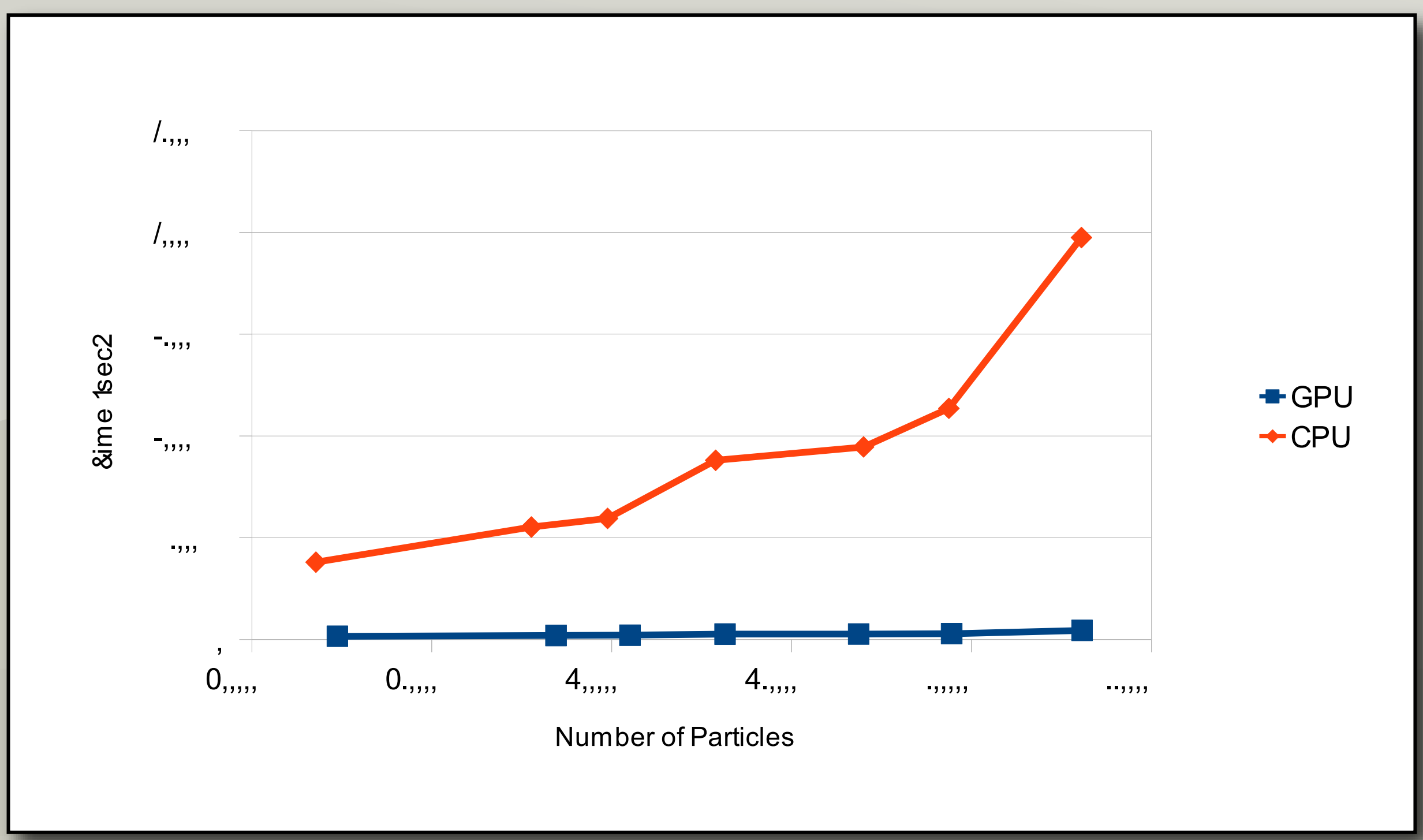


Figure 3. Timing results, in seconds, as the number of particles increase. The GPU times are in blue while the CPU times are in red.