Voronoi Diagram Computation for Protein Molecules Using Graphics Hardware

Ku-Jin Kim* Jung-Eun Lee† Nakhoon Baek‡
Kyungpook National University, Republic of Korea

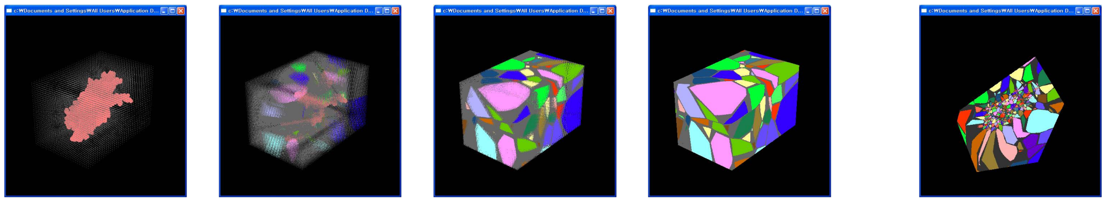
Figure 1: *(a) snapshots from our CUDA-based implementations* *(b) a clipped plane view*

1 Introduction

We present an interactive algorithm to compute Voronoi diagrams for protein molecules. In the research area of biochemistry, a molecule is generally represented as a set of 3D spheres with various radii. In this paper, we propose a method to compute Voronoi diagrams for a set of spheres in the 3D discrete domain. We achieved interactive construction of Voronoi diagrams through our adaptive subdivision scheme and massively parallel processing supported by current graphics hardware.

2 GPU Based Voronoi Diagram Computing

We get the Voronoi diagrams for the given sphere sites. Each sphere site s_i is described as its center point $c_i = (x_i, y_i, z_i)$ and its radius r_i. Let $S = \{s_i | 1 \leq i \leq n\}$ be a set of sphere sites in the 3D space. From a query point $\mathbf{q} = (x, y, z)$, the Euclidean distance to a sphere site s_i is calculated as $dist(\mathbf{q}, s_i) = \|\mathbf{q} - c_i\| - r_i$. The Voronoi region of a sphere site s_p can be formed as : $reg(s_p) = \{\mathbf{p} | dist(\mathbf{p}, s_p) \leq dist(\mathbf{p}, s_i)$, for all $s_i \in S, s_i \neq s_p\}$. When $dist(\mathbf{q}, s_p) = dist(\mathbf{q}, s_q)$, the point $\mathbf{q}$ belongs to both of $reg(s_p)$ and $reg(s_q)$, to finally construct the boundary of those regions.

We are focusing on the discrete domain. Each rectangular voxels in the 3D space is marked to belong to the specific region or to the boundary area, to finally display the 3D Voronoi diagram. We start from the distance between the center point q of the voxel and the nearest sphere site s_p. Though $\mathbf{q} \in reg(s_p)$, we also need to decide whether all the points in the voxel belong to $reg(s_p)$ or not. We present a region decision algorithm for each voxel as follows:

```
Kernel Program
for each voxel,
  with its center $\mathbf{q}$ and half of the diagonal length $d$
step 1.  calculate $dist(\mathbf{q}, s_i)$ from the voxel center point $\mathbf{q}$, for
         all sphere sites $s_i$.
step 2.  get $d_{min} = \min(dist(\mathbf{q}, s_i))$ and its corresponding
         nearest sphere site $s_{min}$.
step 3.  count the number $n$ of the sphere sites whose
         $dist(\mathbf{q}, s_i)$ is less than $d_{min} + 2d$.
step 4.  if $n = 1$, the voxel belongs to $reg(s_{min})$.
         otherwise, it may belong to the boundary of the
         Voronoid regions with respect to the corresponding
         sphere sites.
```

*e-mail: kujinkim@yahoo.com
†e-mail:highshia@nate.com
‡(corresponding author) e-mail:oceancru@gmail.com

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007

The above algorithm should be executed for all voxels, and thus its CPU-based sequential execution may require a considerable amount of time. We achieved its interactive execution with CUDA-based massively parallel implementation. Voxel-related data are stored into the CUDA texture memories and the above voxel-specific algorithm is realized as a CUDA kernel program to finally be executed as CUDA threads. Overall CUDA framework with adaptive refinement can be represented as follows:

```
procedure framework
do
 step1.   subdivide the space into $8 \times 8 \times 8$ voxels, which is the
          current CUDA hardware limit.
 step 2.  invoke kernel program for each voxel
 step 3.  find any voxels belonged to boundary areas
 step 4.  if any, process framework recursively for those voxels.
until user-specified accuracy limit is achieved.
```

3 Experimental Results

As shown in Table 1, we achieved at least 15 times and at most 129 times faster performance in comparison with CPUbased sequential implementations. Our CUDA-based implementation can be used in an interactive manner, while CPU-based ones not.

subdivision	execution time (msec)		Speed ups
	CPU-based	CUDA-based	(times)
$32 \times 32 \times 32$	750	48	15.63
$64 \times 64 \times 64$	6,016	69	87.19
$128 \times 128 \times 128$	47,750	404	118.19
$256 \times 256 \times 256$	381,625	2,939	129.85

CPU: Intel Core2 Duo E8400, 3GHz CPU with 3GB RAM
GPU: nVIDIA GeForce GTX 285 with 1GB Video RAM
with 1468 sphere sites from a protein molecule description file.

Table 1: *Execution times for our implementations*

4 Conclusions

In this paper, we presented an interactive-time algorithm to compute Voronoi diagrams for protein molecules. With respect to the user-specified accuracy limits, our algorithm shows at least 15 times to at most 129 times better performance in comparison to the single-core CPU-based implementations. Based on our work, we expect to develop interactive-time algorithms to compute the volume of the protein and molecular surfaces as further researches.