GPU Particle-Grid Methods:
Molecular Surfaces and Synthetic Density Maps

John Stone

Theoretical and Computational Biophysics Group

Beckman Institute for Advanced Science and Technology

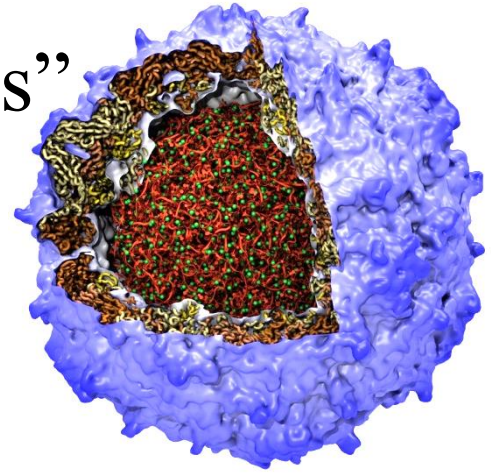
University of Illinois at Urbana-Champaign

http://www.ks.uiuc.edu/Research/gpu/

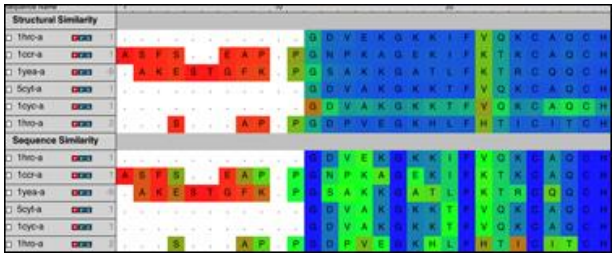
Workshop on GPU Programming for Molecular Modeling , Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, August 3, 2013

VMD – "Visual Molecular Dynamics"

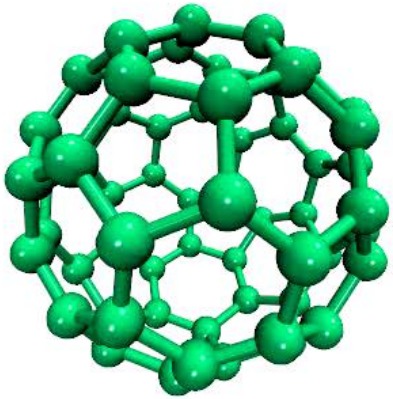
- Visualization and analysis of:
  - molecular dynamics simulations
  - quantum chemistry calculations
  - particle systems and whole cells
  - sequence data
- User extensible w/ scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/



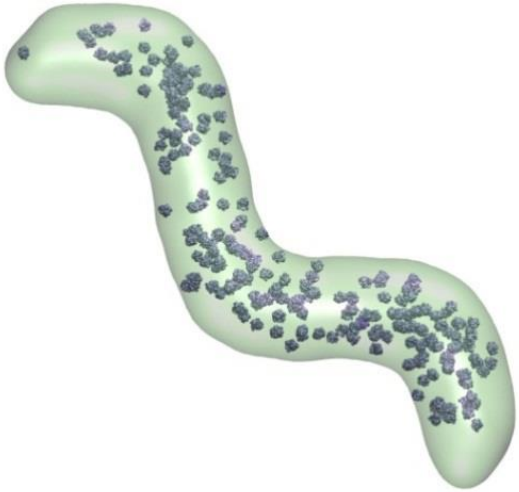
Poliovirus



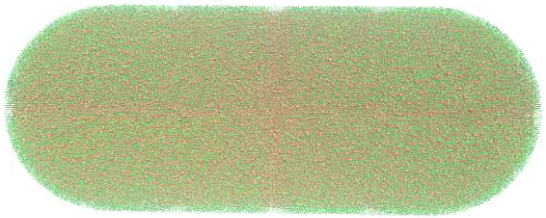
Ribosome Sequences



Electrons in
Vibrating Buckyball



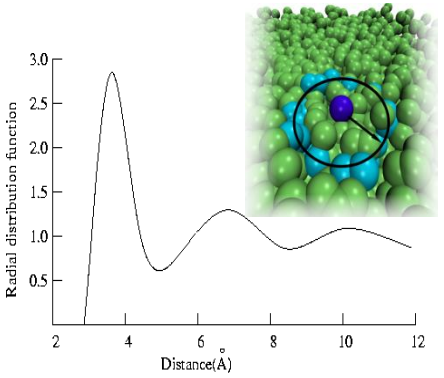
Cellular Tomography,
Cryo-electron Microscopy

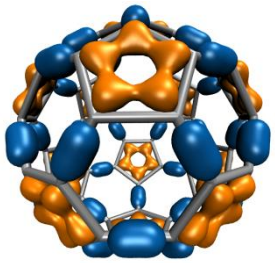


Whole Cell Simulations

GPU Accelerated Trajectory Analysis and Visualization in VMD

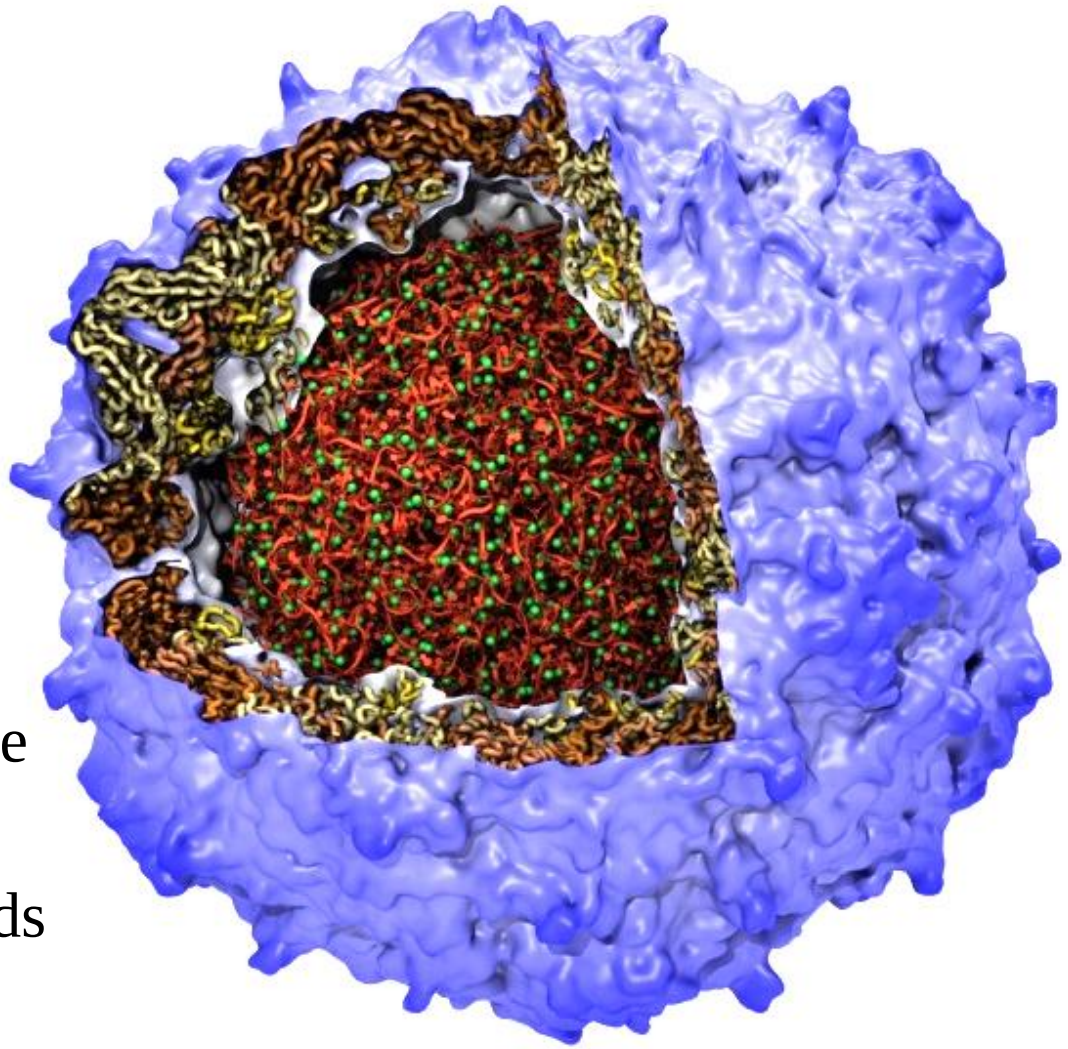
GPU-Accelerated Feature	Peak speedup vs. single CPU core
Molecular orbital display	120x
Radial distribution function	92x
Electrostatic field calculation	44x
Molecular surface display	40x
Ion placement	26x
MDFF density map synthesis	26x
Implicit ligand sampling	25x
Root mean squared fluctuation	25x
Radius of gyration	21x
Close contact determination	20x
Dipole moment calculation	15x





Molecular Surface Visualization

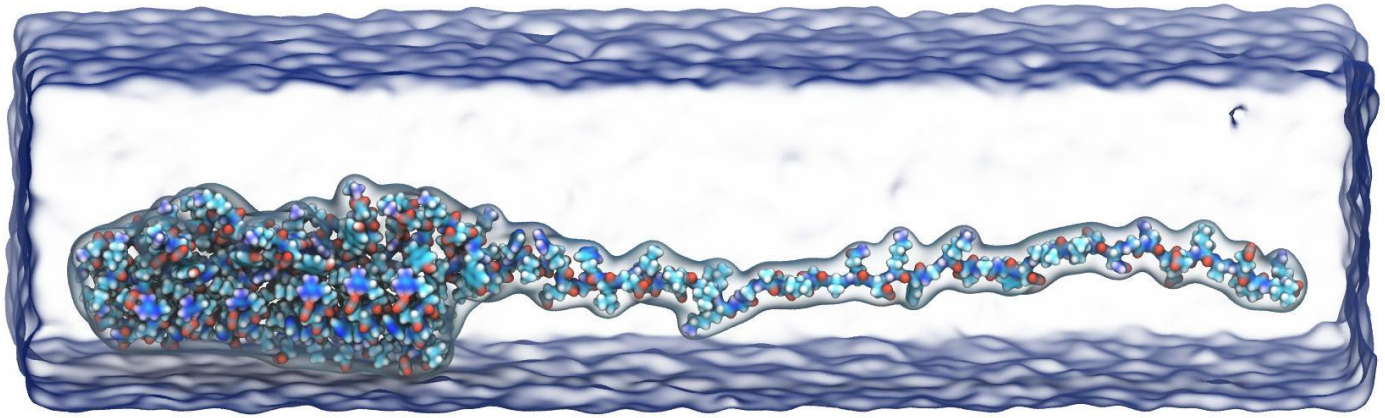
- Large biomolecular complexes are difficult to interpret with atomic detail graphical representations
- Even secondary structure representations become cluttered
- Surface representations are easier to use when greater abstraction is desired, but are computationally costly
- Most surface display methods incapable of animating dynamics of large structures w/ millions of particles



Poliovirus

VMD "QuickSurf" Representation

- Displays continuum of structural detail:
  - All-atom models
  - Coarse-grained models
  - Cellular scale models
  - Multi-scale models: All-atom + CG, Brownian + Whole Cell
  - Smoothly variable between full detail, and reduced resolution representations of very large complexes

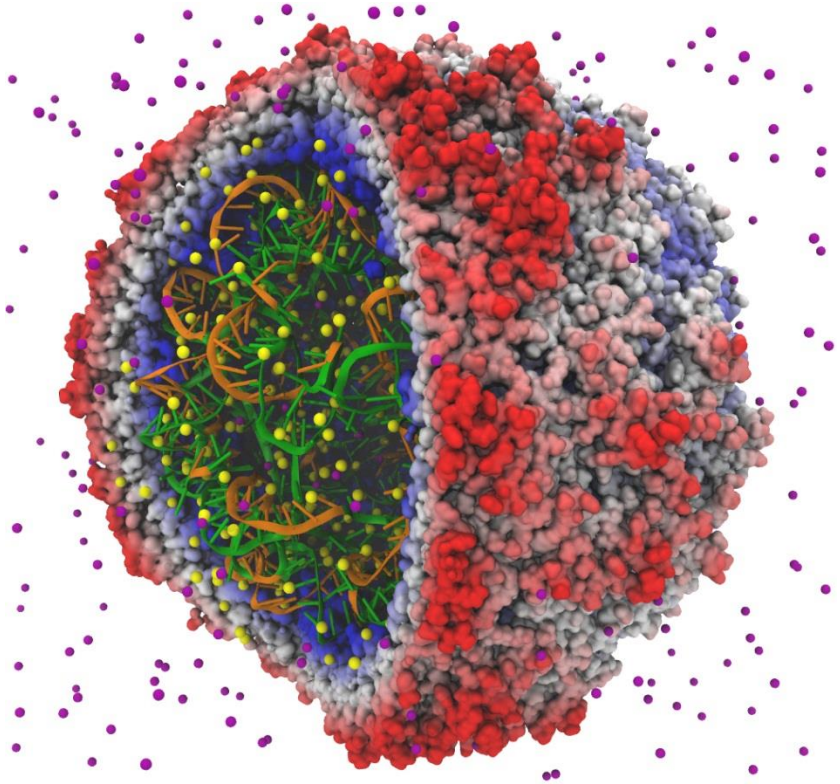


Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.

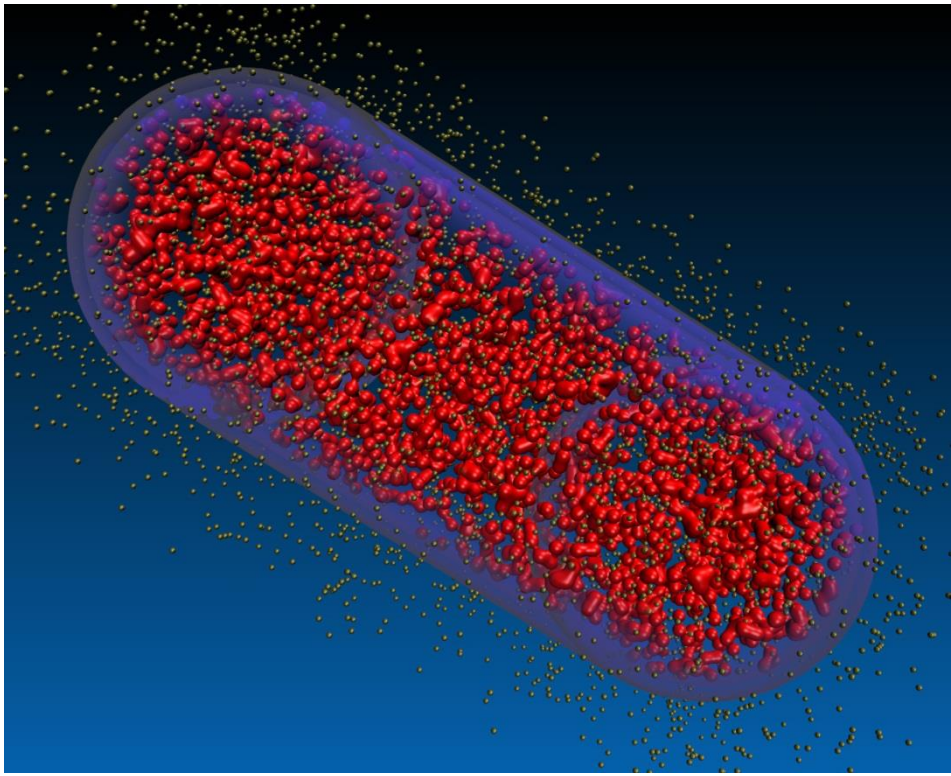
M. Krone, J. E. Stone, T. Ertl, K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012

VMD “QuickSurf” Representation

- Uses multi-core CPUs and GPU acceleration to enable **smooth real-time animation** of MD trajectories
- Linear-time algorithm, scales to millions of particles, as limited by memory capacity

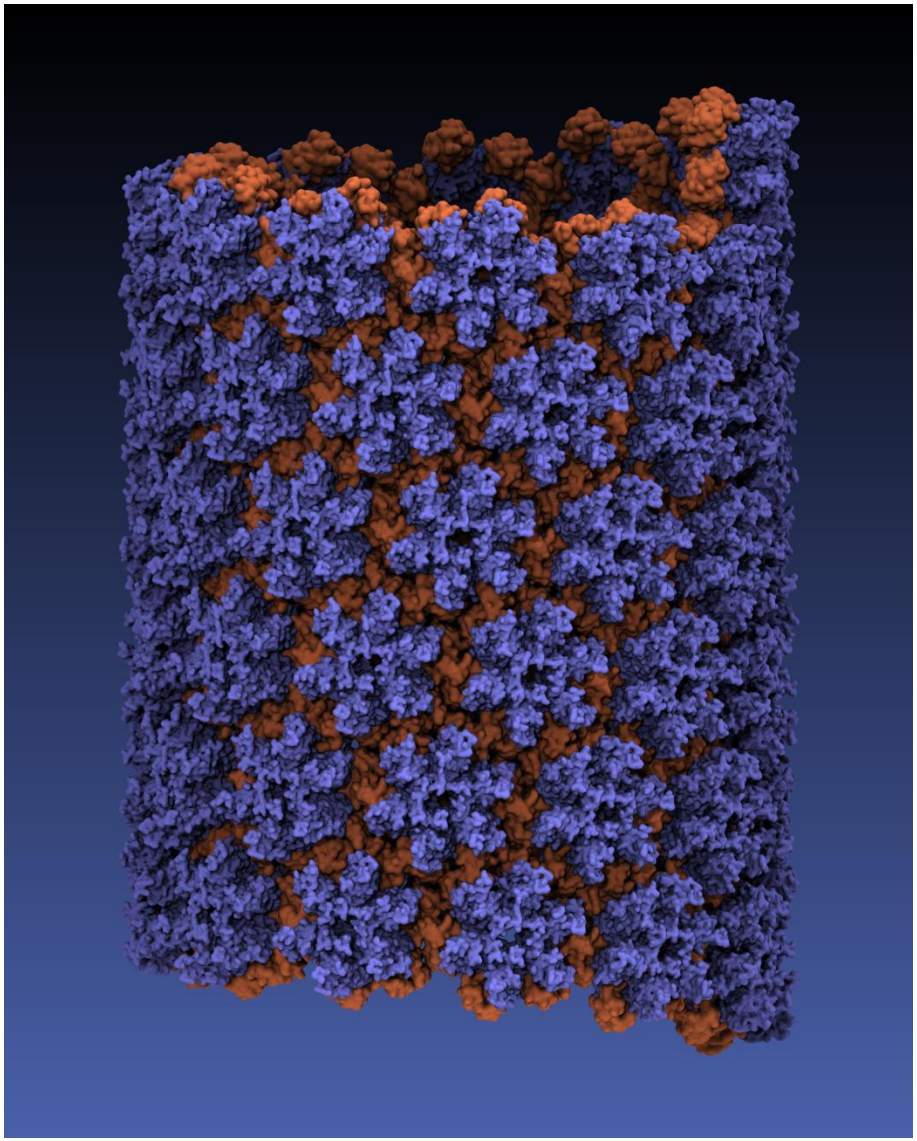


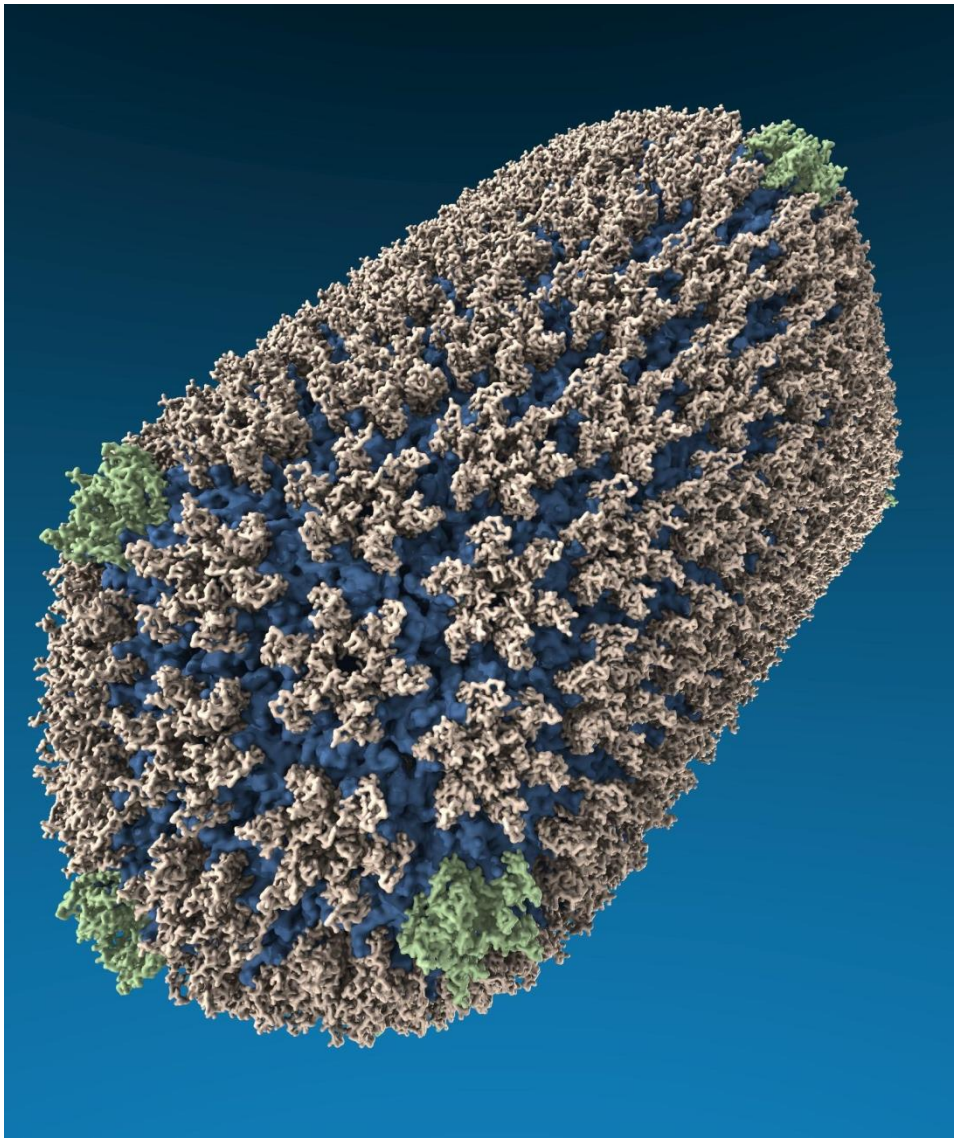
Satellite Tobacco Mosaic Virus



Lattice Cell Simulations

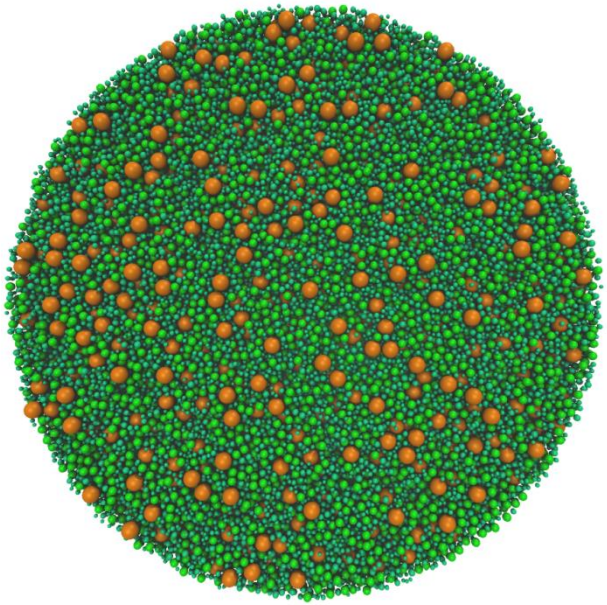
VMD “QuickSurf” Representation

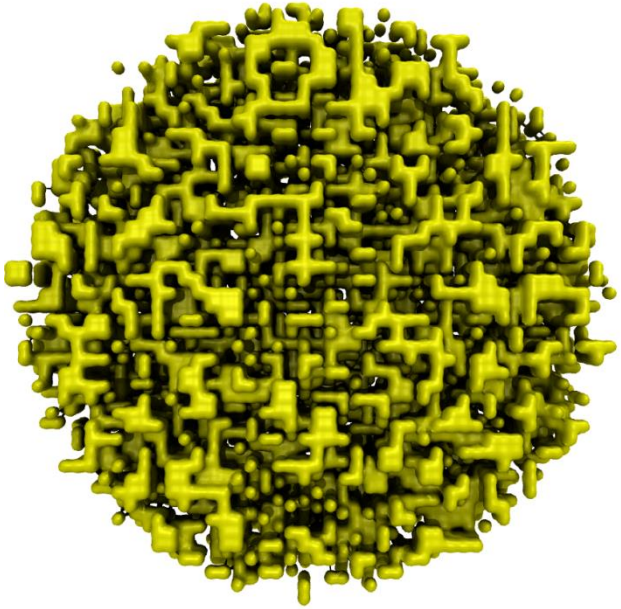


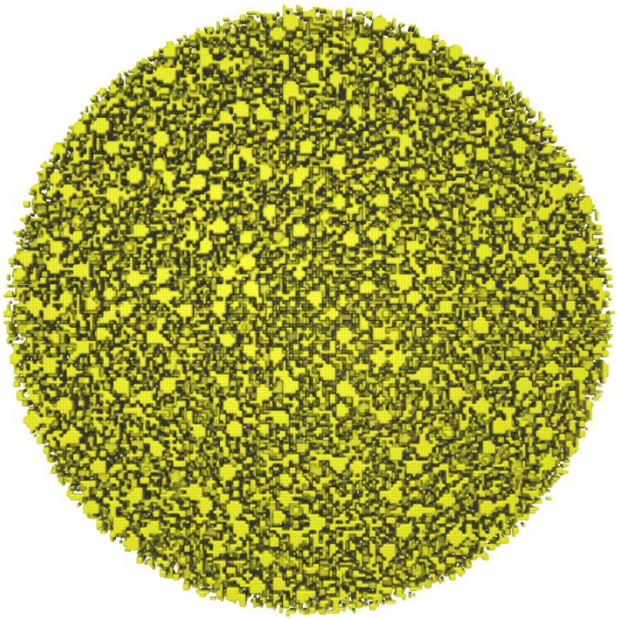


All-atom HIV capsid simulations

QuickSurf Representation of Lattice Cell Models







Continuous particle based model – often 70 to 300 million particles

Discretized lattice models derived from continuous model shown in VMD QuickSurf representation

Lattice Microbes: High-performance stochastic simulation method for the reaction-diffusion master equation

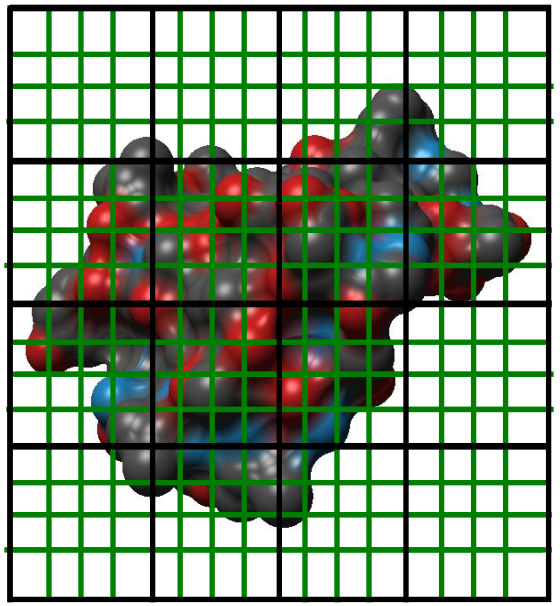
E. Roberts, J. E. Stone, and Z. Luthey-Schulten.
J. Computational Chemistry 34 (3), 245-255, 2013.

QuickSurf Algorithm Overview

- Build spatial acceleration data structures, optimize data for GPU
- Compute 3-D density map, 3-D volumetric texture map:

$$\rho(\vec{r}; \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \sum_{i=1}^{N} e^{\frac{-|\vec{r}-\vec{r}_i|^2}{2\alpha^2}}$$

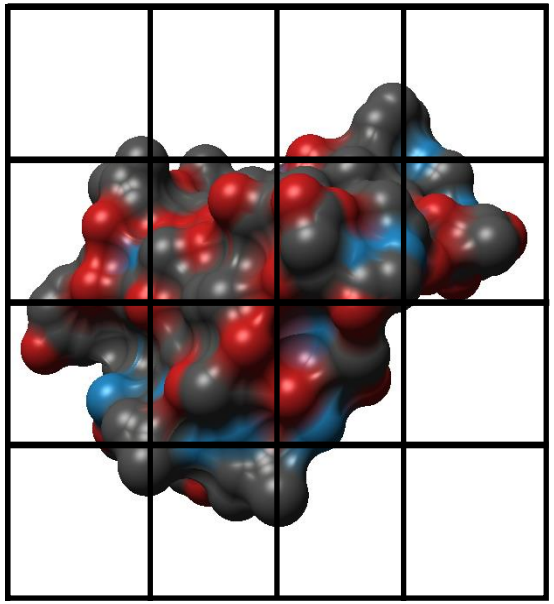
- Extract isosurface for a user-defined density value



3-D density map lattice, spatial acceleration grid, and extracted surface

QuickSurf Particle Sorting, Bead Generation, Spatial Hashing

- Particles sorted into spatial acceleration grid:
  - Selected atoms or residue “beads” converted lattice coordinate system
  - Each particle/bead assigned cell index, sorted w/NVIDIA Thrust template library
- Complication:
  - Thrust allocates GPU mem. on-demand, no recourse if insufficient memory, have to re-gen QuickSurf data structures if caught by surprise!
- Workaround:
  - Pre-allocate guesstimate workspace for Thrust
  - Free the Thrust workspace right before use
  - Newest Thrust allows user-defined allocator code…



Coarse resolution spatial acceleration grid

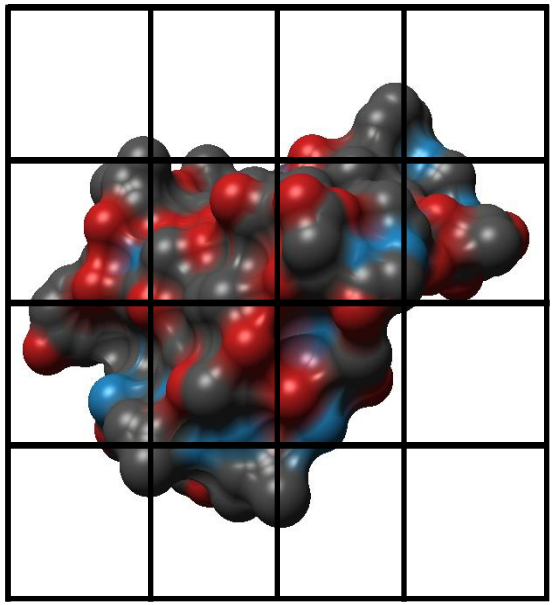
Spatial Hashing Algorithm Steps/Kernels

1) Compute bin index for each atom, store to memory w/ atom index

2) **Sort** list of bin and atom index tuples (1) by bin index **(thrust kernel)**

3) Count atoms in each bin (2) using a **parallel prefix sum, aka *scan***, compute the destination index for each atom, store per-bin starting index and atom count **(thrust kernel)**

4) Write atoms to the output indices computed in (3), and we have completed the data structure

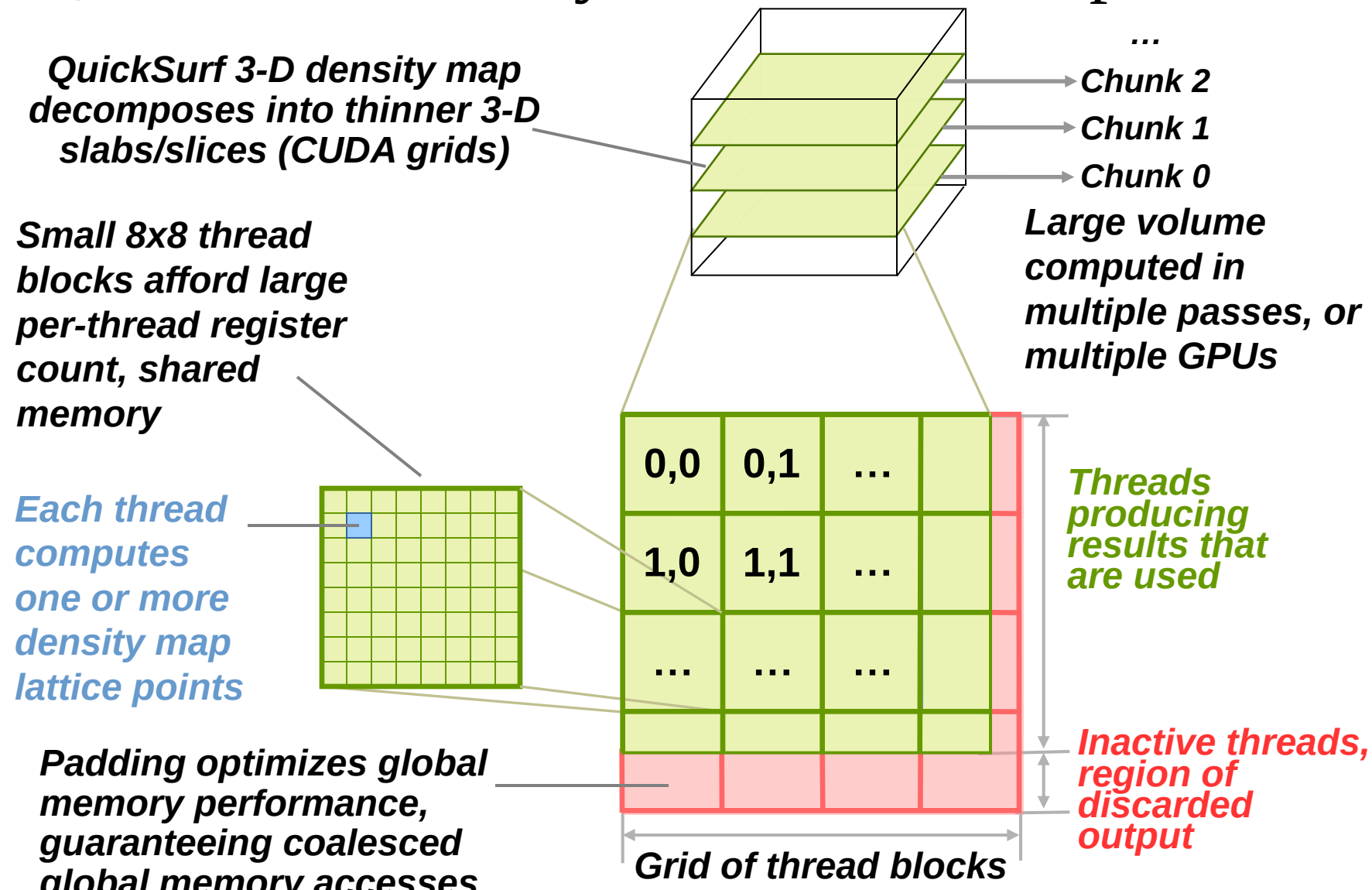


QuickSurf uniform grid spatial subdivision data structure

QuickSurf and Limited GPU Global Memory

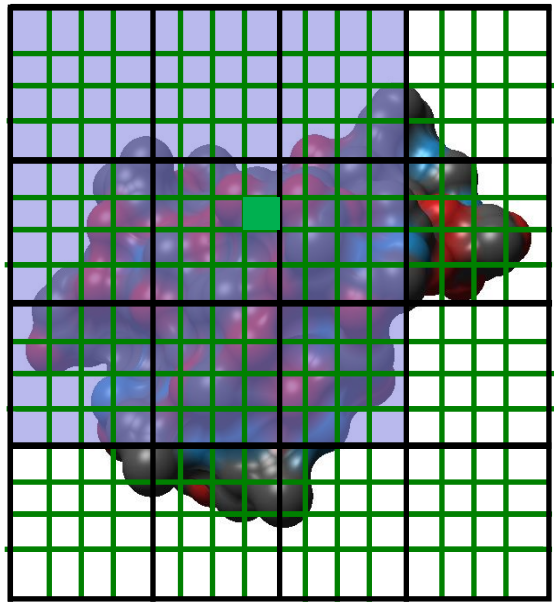
- High resolution molecular surfaces require a fine lattice spacing
- Memory use grows cubically with decreased lattice spacing
- Not typically possible to compute a surface in a single pass, so we loop over sub-volume “chunks” until done…
- Chunks pre-allocated and sized to GPU global mem capacity to prevent unexpected memory allocation failure while animating…
- Complication:
  - Thrust allocates GPU mem. on-demand, no recourse if insufficient memory, have to re-gen QuickSurf data structures if caught by surprise!
- Workaround:
  - Pre-allocate guesstimate workspace for Thrust
  - Free the Thrust workspace right before use
  - Newest Thrust allows user-defined allocator code…

QuickSurf Density Parallel Decomposition



QuickSurf Density Map Algorithm

- Spatial acceleration grid cells are sized to match the cutoff radius for the exponential, beyond which density contributions are negligible
- Density map lattice points computed by summing density contributions from particles in 3x3x3 grid of neighboring spatial acceleration cells
- Volumetric texture map is computed by summing particle colors normalized by their individual density contribution



3-D density map lattice point and the neighboring spatial acceleration cells it references

QuickSurf Density Map Kernel Optimizations

- Compute reciprocals, prefactors, other math on the host CPU prior to kernel launch
- Use of **intN** and **floatN** vector types in CUDA kernels for improved global memory bandwidth
- **Thread coarsening**: one thread computes multiple output densities and colors
- Input data and **register tiling**: share blocks of input, partial distances in regs shared among multiple outputs
- Global memory **(L1 cache) broadcasts**: all threads in the block traverse the same atom/particle at the same time

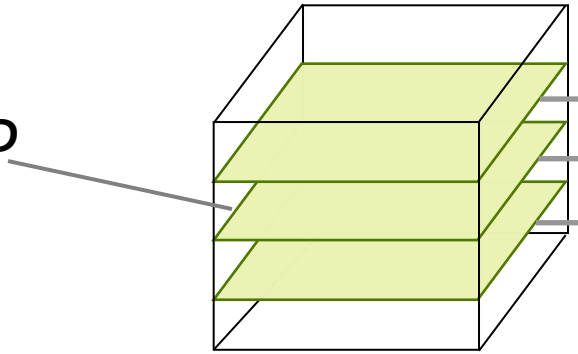
QuickSurf Density Map Kernel Snippet…

```
for (zab=zabmin; zab<=zabmax; zab++) {
  for (yab=yabmin; yab<=yabmax; yab++) {
    for (xab=xabmin; xab<=xabmax; xab++) {
      int abcellidx = zab * acplanesz + yab * acncells.x + xab;
      uint2 atomstartend = cellStartEnd[abcellidx];
      if (atomstartend.x != GRID_CELL_EMPTY) {
        for (unsigned int atomid=atomstartend.x; atomid<atomstartend.y; atomid++) {
          float4 atom = sorted_xyzr[atomid];
          float dx = coorx - atom.x;          float dy = coory - atom.y;        float dz = coorz - atom.z;
          float dxy2 = dx*dx + dy*dy;
          float r21 = (dxy2 + dz*dz) * atom.w;
          densityval1 += exp2f(r21);
          /// Loop unrolling and register tiling benefits begin here……
          float dz2 = dz + gridspacing;
          float r22 = (dxy2 + dz2*dz2) * atom.w;
          densityval2 += exp2f(r22);
          /// More loop unrolling ….
```

QuickSurf Marching Cubes Isosurface Extraction

- Isosurface is extracted from each density map "chunk", and either copied back to the host, or **rendered directly** out of GPU global memory via **CUDA/OpenGL interop**
- All MC memory buffers are pre-allocated to prevent significant overhead when animating a simulation trajectory

QuickSurf 3-D density map decomposes into thinner 3-D slabs/slices (CUDA grids)



…

Chunk 2

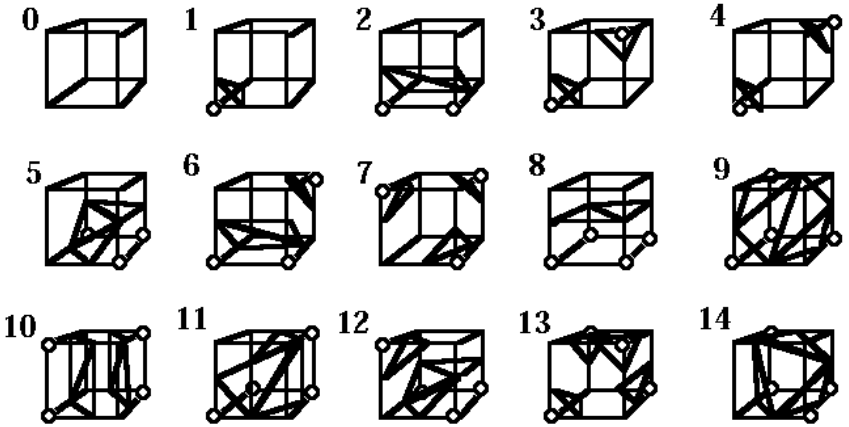
Chunk 1

Chunk 0

Large volume computed in multiple passes

Brief Marching Cubes Isosurface Extraction Overview

- Given a 3-D volume of scalar density values and a requested surface density value, marching cubes computes vertices and triangles that compose the requested surface triangle mesh
- Each MC "cell" (a cube with 8 density values at its vertices) produces a variable number of output vertices depending on how many edges of the cell contain the requested isovalue…
- Use **scan()** to compute the output indices so that each worker thread has **conflict-free output** of vertices/triangles



Brief Marching Cubes Isosurface Extraction Overview

- Once the output vertices have been computed and stored, we compute surface normals and colors for each of the vertices
- Although the separate normals+colors pass reads the density map again, molecular surfaces tend to generate a small percentage of MC cells containing triangles, we avoid wasting interpolation work
- We use CUDA **tex3D()** hardware 3-D texture mapping:
  - Costs double the texture memory and a one copy from GPU global memory to the target texture map with **cudaMemcpy3D()**
  - Still roughly 2x faster than doing color interpolation without the texturing hardware, at least on GT200 and Fermi hardware
  - Kepler has new texture cache memory path that may make it feasible to do our own color interpolation and avoid the use of extra 3-D texture memory and associated copy, with acceptable performance

QuickSurf Marching Cubes Isosurface Extraction

- Our optimized MC implementation computes per-vertex surface normals, colors, and outperforms the NVIDIA SDK sample by a fair margin on Fermi GPUs
- Complications:
  - Even on a 6GB Quadro 7000, GPU global memory is under great strain when working with large molecular complexes, e.g. viruses
  - Marching cubes involves a parallel prefix sum (scan) to compute target indices for writing resulting vertices
  - We use Thrust for scan, has the same memory allocation issue mentioned earlier for the sort, so we use the same workaround
  - The number of output vertices can be huge, but we rarely have sufficient GPU memory for this – we use a fixed size vertex output buffer and hope our heuristics don't fail us

QuickSurf Performance
GeForce GTX 580

Molecular system	Atoms	Resolution	T_{sort}	$T_{density}$	T_{MC}	# vertices	FPS
MscL	111,016	1.0Å	0.005	0.023	0.003	0.7 M	28
STMV capsid	147,976	1.0Å	0.007	0.048	0.009	2.4 M	13.2
Poliovirus capsid	754,200	1.0Å	0.01	0.18	0.05	9.2 M	3.5
STMV w/ water	955,225	1.0Å	0.008	0.189	0.012	2.3 M	4.2
Membrane	2.37 M	2.0Å	0.03	0.17	0.016	5.9 M	3.9
Chromatophore	9.62 M	2.0Å	0.16	0.023	0.06	11.5 M	3.4
Membrane w/ water	22.77 M	4.0Å	4.4	0.68	0.01	1.9 M	0.18

Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.

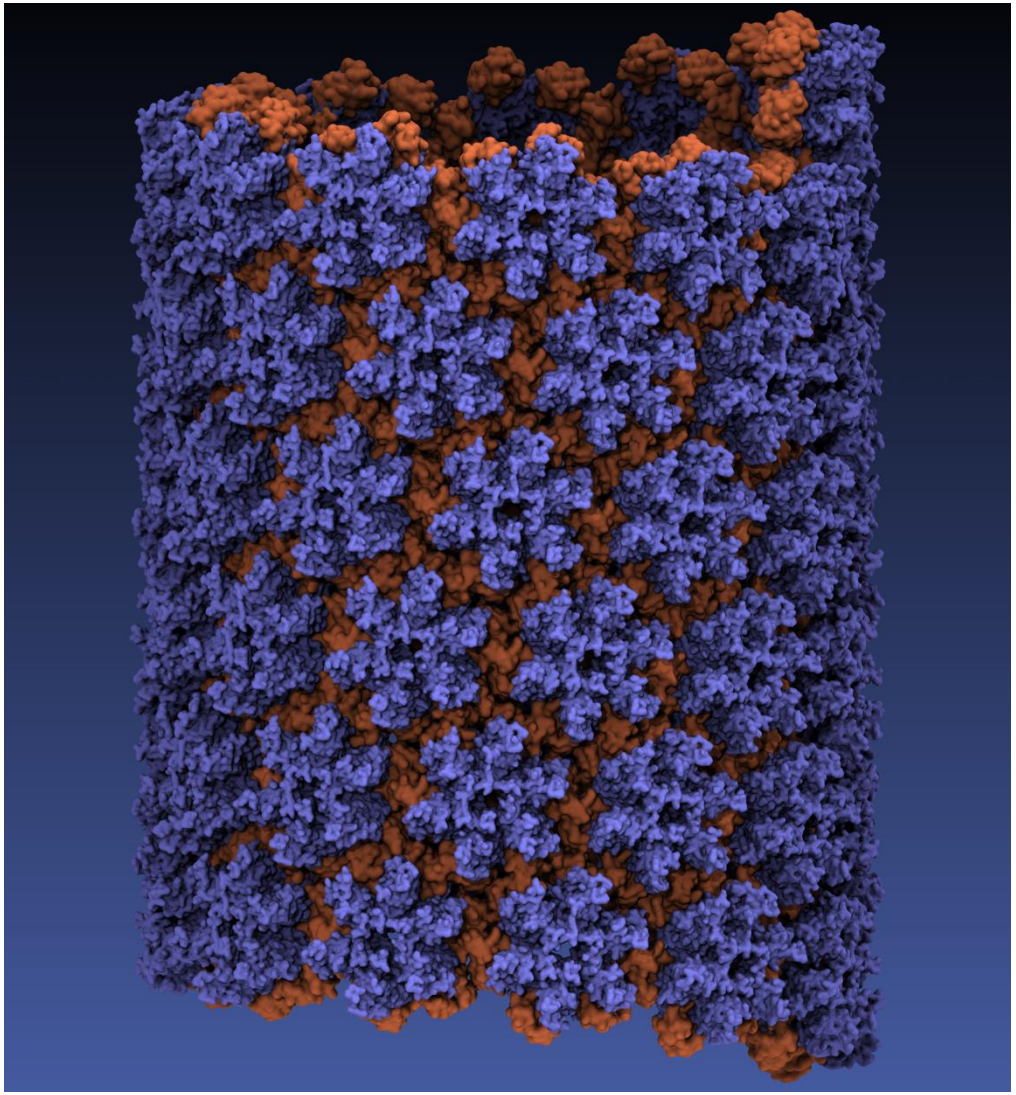
M. Krone, J. E. Stone, T. Ertl, K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012

Extensions and Analysis Uses for QuickSurf Triangle Mesh

- Curved PN triangles:
  - We have performed tests with post-processing the resulting triangle mesh and using curved PN triangles to generate smooth surfaces with a larger grid spacing, for increased performance
  - Initial results demonstrate some potential, but there can be pathological cases where MC generates long skinny triangles, causing unsightly surface creases
- Analysis uses (beyond visualization):
  - Minor modifications to the density map algorithm allow rapid computation of solvent accessible surface area by summing the areas in the resulting triangle mesh
  - Modifications to the density map algorithm will allow it to be used for MDFF (molecular dynamics flexible fitting)
  - Surface triangle mesh can be used as the input for computing the electrostatic potential field for mesh-based algorithms

Challenge: Support Interactive QuickSurf for Large Structures on Mid-Range GPUs

- Structures such as HIV initially needed large (6GB) GPU memory to generate fully-detailed surface renderings
- Goals and approach:
  - **Avoid slow CPU-fallback!**
  - Incrementally change algorithm phases to use more compact data types, while maintaining performance
  - Specialize code for different performance/memory capacity cases



Improving QuickSurf Memory Efficiency

- Both host and GPU memory capacity limitations are a significant concern when rendering surfaces for virus structures such as HIV or for large cellular models which can contain hundreds of millions of particles
- The original QuickSurf implementation used single-precision floating point for output vertex arrays and textures
- Judicious use of reduced-precision numerical representations, cut the overall memory footprint of the entire QuickSurf algorithm to half of the original
  - Data type changes made throughout the entire chain from density map computation through all stages of Marching Cubes

Supporting Multiple Data Types for QuickSurf Density Maps and Marching Cubes Vertex Arrays

- The major algorithm components of QuickSurf are now used for many other purposes:
  - Gaussian density map algorithm now used for MDFF Cryo EM density map fitting methods in addition to QuickSurf
  - Marching Cubes routines also used for Quantum Chemistry visualizations of molecular orbitals
- Rather than simply changing QuickSurf to use a particular internal numerical representation, it is desirable to instead use **CUDA C++ templates** to make type-generic versions of the key objects, kernels, and output vertex arrays
- Accuracy-sensitive algorithms use high-precision data types, performance and memory capacity sensitive cases use quantized or reduced precision approaches

Minimizing the Impact of Generality on QuickSurf Code Complexity

- A critical factor in the simplicity of supporting multiple QuickSurf data types arises from the so-called ***"gather"*** oriented algorithm we employ
  - Internally, all in-register arithmetic is single-precision
  - Data conversions to/from compressed or reduced precision data types are performed on-the-fly as needed
- Small **inlined** type conversion routines are defined for each of the cases we want to support
- Key QuickSurf kernels are genericized using C++ template syntax, and the compiler "connects the dots" to automatically generate type-specific kernels as needed

Example Templated Density Map Kernel

```
template<class DENSITY, class VOLTEX>
__global__ static void
gaussdensity_fast_tex_norm(int natoms,
                           const float4 * RESTRICT sorted_xyzr,
                           const float4 * RESTRICT sorted_color,
                           int3 numvoxels,
                           int3 acncells,
                          float acgridspacing,
                          float invacgridspacing,
                          const uint2 * RESTRICT cellStartEnd,
                          float gridspacing, unsigned int z,
                          DENSITY * RESTRICT densitygrid,
                          VOLTEX * RESTRICT voltexmap,
                         float invisovalue) {
```

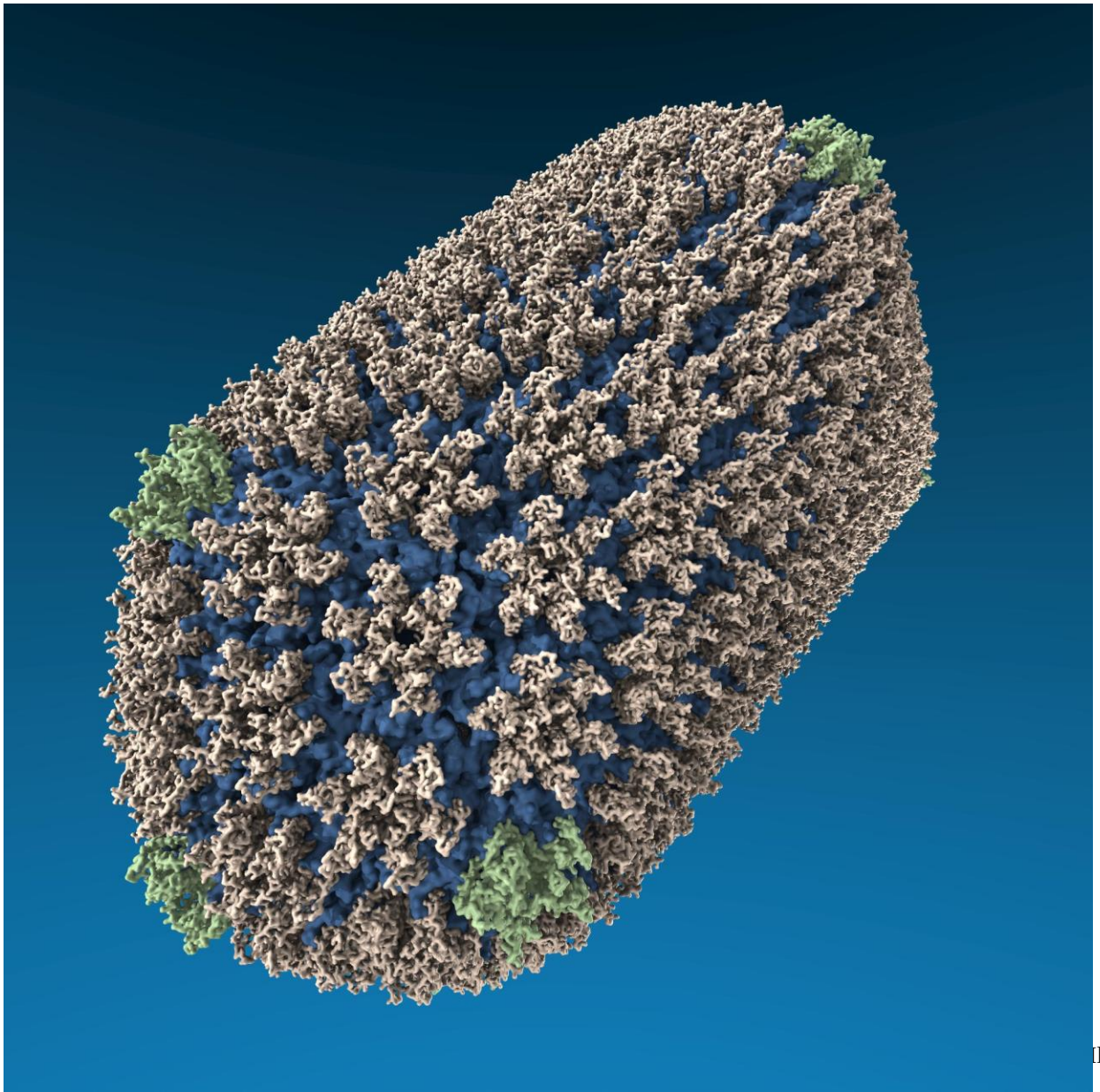
Example Templated Density Map Kernel

```
template<class DENSITY, class VOLTEX>
__global__ static void
gaussdensity_fast_tex_norm( …  ) {

  … Triple-nested and unrolled inner loops here …

  DENSITY densityout;
  VOLTEX texout;
  convert_density(densityout, densityval1);
  densitygrid[outaddr         ] = densityout;
  convert_color(texout, densitycol1);
  voltexmap[outaddr         ] = texout;
```

Net Result of QuickSurf Memory Efficiency Optimizations

- **Halved** overall GPU memory use
- Achieved **1.5x to 2x performance gain**:
  - The "gather" density map algorithm keeps type conversion operations out of the innermost loop
  - Density map global memory writes reduced to half
  - Multiple stages of Marching Cubes operate on smaller input and output data types
  - Same code path supports multiple precisions
- Users now get full GPU-accelerated QuickSurf in many cases that previously triggered CPU-fallback, all platforms (laptop/desk/super) benefit!

High Resolution HIV Surface



Acknowledgements

- Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign
- NCSA Blue Waters Team
- NCSA Innovative Systems Lab
- NVIDIA CUDA Center of Excellence, University of Illinois at Urbana-Champaign
- The CUDA team at NVIDIA
- NIH support: 9P41GM104601

GPU Computing Publications

http://www.ks.uiuc.edu/Research/gpu/

- **Lattice Microbes: High-performance stochastic simulation method for the reaction-diffusion master equation.**
  E. Roberts, J. E. Stone, and Z. Luthey-Schulten.
  J. Computational Chemistry 34 (3), 245-255, 2013**.**
- **Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.** M. Krone, J. E. Stone, T. Ertl, and K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012.
- **Immersive Out-of-Core Visualization of Large-Size and Long-Timescale Molecular Dynamics Trajectories.** J. Stone, K. Vandivort, and K. Schulten. G. Bebis et al. (Eds.): *7th International Symposium on Visual Computing (ISVC 2011)*, LNCS 6939, pp. 1-12, 2011.
- **Fast Analysis of Molecular Dynamics Trajectories with Graphics Processing Units – Radial Distribution Functions.** B. Levine, J. Stone, and A. Kohlmeyer. *J. Comp. Physics*, 230(9):3556-3569, 2011.

GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/

- **Quantifying the Impact of GPUs on Performance and Energy Efficiency in HPC Clusters.** J. Enos, C. Steffen, J. Fullop, M. Showerman, G. Shi, K. Esler, V. Kindratenko, J. Stone, J Phillips. *International Conference on Green Computing*, pp. 317-324, 2010.
- **GPU-accelerated molecular modeling coming of age. J. Stone, D. Hardy, I. Ufimtsev, K. Schulten.** *J. Molecular Graphics and Modeling*, 29:116-125, 2010.
- **OpenCL: A Parallel Programming Standard for Heterogeneous Computing. J. Stone, D. Gohara, G. Shi.** *Computing in Science and Engineering*, 12(3):66-73, 2010.
- **An Asymmetric Distributed Shared Memory Model for Heterogeneous Computing Systems**. I. Gelado, J. Stone, J. Cabezas, S. Patel, N. Navarro, W. Hwu. *ASPLOS '10: Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 347-358, 2010.

GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/

- **GPU Clusters for High Performance Computing**. V. Kindratenko, J. Enos, G. Shi, M. Showerman, G. Arnold, J. Stone, J. Phillips, W. Hwu. *Workshop on Parallel Programming on Accelerator Clusters (PPAC)*, In Proceedings IEEE Cluster 2009, pp. 1-8, Aug. 2009.
- **Long time-scale simulations of in vivo diffusion using GPU hardware**. E. Roberts, J. Stone, L. Sepulveda, W. Hwu, Z. Luthey-Schulten. In *IPDPS'09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Computing*, pp. 1-8, 2009.
- **High Performance Computation and Interactive Display of Molecular Orbitals on GPUs and Multi-core CPUs**. J. Stone, J. Saam, D. Hardy, K. Vandivort, W. Hwu, K. Schulten, *2nd Workshop on General-Purpose Computation on Graphics Pricessing Units (GPGPU-2), ACM International Conference Proceeding Series*, volume 383, pp. 9-18, 2009.
- **Probing Biomolecular Machines with Graphics Processors**. J. Phillips, J. Stone. *Communications of the ACM*, 52(10):34-41, 2009.
- **Multilevel summation of electrostatic potentials using graphics processing units**. D. Hardy, J. Stone, K. Schulten. *J. Parallel Computing*, 35:164-177, 2009.

GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/

- **Adapting a message-driven parallel application to GPU-accelerated clusters**. J. Phillips, J. Stone, K. Schulten. *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, IEEE Press, 2008.
- **GPU acceleration of cutoff pair potentials for molecular modeling applications**. C. Rodrigues, D. Hardy, J. Stone, K. Schulten, and W. Hwu. *Proceedings of the 2008 Conference On Computing Frontiers*, pp. 273-282, 2008.
- **GPU computing**. J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, J. Phillips. *Proceedings of the IEEE*, 96:879-899, 2008.
- **Accelerating molecular modeling applications with graphics processors**. J. Stone, J. Phillips, P. Freddolino, D. Hardy, L. Trabuco, K. Schulten. *J. Comp. Chem.*, 28:2618-2640, 2007.
- **Continuous fluorescence microphotolysis and correlation spectroscopy**. A. Arkhipov, J. Hüve, M. Kahms, R. Peters, K. Schulten. *Biophysical Journal*, 93:4006-4017, 2007.