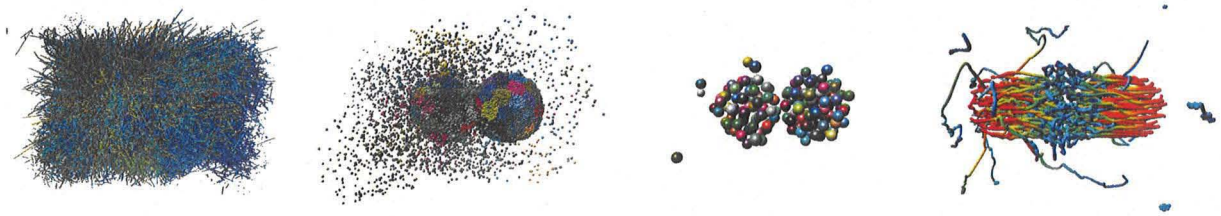
First publ. in: Proceedings of the 2011 IEEE Pacific Visualization Symposium, 2011 : PacificVIS '11 ; Hongkong, China, 1 - 4 March 2011 / ed. by Guiseppe Di Battista; Jean-Daniel Fekete, ... - Piscataway, NJ : IEEE Service Center, 2011. - pp. 51-58. - ISBN 978-1-61284-935-5

http://dx.doi.org/10.1109/PACIFICVIS.2011.5742372

Loose Capacity-Constrained Representatives for the Qualitative Visual Analysis in Molecular Dynamics

Steffen Frey[†], Thomas Schlömer[‡], Sebastian Grottel[†], Carsten Dachsbacher[§], Oliver Deussen[‡] and Thomas Ertl[†]

[†]Visualization Research Center Universität Stuttgart (VISUS) [‡]University of Konstanz [§]Karlsruhe Institute of Technology



(a) Path lines of 81672 molecules over 600 time steps

(b) Grouped molecules for a single time step

(c) LCCVD representatives for the groups in (b)

(d) Path lines of the representatives for all time steps

Figure 1: Showing all points of a large and time-dependent data set at once usually results in expensive (non-interactive) rendering, high storage requirement, and heavy occlusion. LCCVD allows to drastically reduce the amount of points very quickly using a GPU-friendly algorithm which preserves the basic structure of the data set.

Abstract

Molecular dynamics is a widely used simulation technique to investigate material properties and structural changes under external forces. The availability of more powerful clusters and algorithms continues to increase the spatial and temporal extents of the simulation domain. This poses a particular challenge for the visualization of the underlying processes which might consist of millions of particles and thousands of time steps. Some application domains have developed special visual metaphors to only represent the relevant information of such data sets but these approaches typically require detailed domain knowledge that might not always be available or applicable.

We propose a general technique that replaces the huge amount of simulated particles by a smaller set of representatives that are used for the visualization instead. The representatives capture the characteristics of the underlying particle density and exhibit coherency over time. We introduce loose capacity-constrained Voronoi diagrams for the generation of these representatives by means of a GPU-friendly, parallel algorithm. This way we achieve visualizations that reflect the particle distribution and geometric structure of the original data very faithfully. We evaluate our approach using real-world data sets from the application domains of material science, thermodynamics and dynamical systems theory.

Keywords: particle-based visualization, molecular dynamics, clustering, time-dependent data.

Index Terms: Computer Graphics [I.3.1]: Parallel processing Computer Graphics [I.3.8]: Applications

1 Introduction

Many applications in visualization and simulation are based on particles. Molecular dynamics (MD) is a prominent example: it uses particles to represent individual atoms or molecules, and models their interaction with each other. This allows to study effects on the micro scale, such as thermodynamical behavior of the nucleation during phase transition, or the atomistic behavior of solid material under external forces, e.g. deformation and destruction of a block of metal during laser ablation. A realistic simulation typically requires a huge amount of particles and many time steps to fully capture the underlying processes. In such a case, data size quickly becomes problematic for visualization: a huge number of particles, often rendered as spheres, can result in high occlusion and poor perceptibility due to an overloaded image. While the simulation and a *quantitative analysis* may require the full data set, a *qualitative visual evaluation* usually benefits from a clearer representation using a reduced number of graphical primitives. This is especially the case for the visual analysis of time-dependent data. As a consequence, some application domains have created special visual metaphors (e.g. the cartoon representation for proteins [23]) which abstracts from the individual atoms. The extraction of relevant features, however, typically requires detailed knowledge of domain experts, and may require manual parameter tuning.

In order to generate meaningful visualizations of arbitrary and large time-dependent particle data sets, we propose to generate a set of representatives from such large collections of particles merely by analyzing the data set itself. Our representatives reproduce the density distribution of the underlying input particles very faithfully and thus the characteristic structure of the data. Our approach utilizes *loose capacity constraints* based on capacity-constrained Voronoi diagrams (CCVDs) of the finite space constituted by the particles. Loosening the capacity constraint allows the centroids of the Voronoi diagrams to adapt well to the incoming data set.

In particular, our paper makes the following contributions:

- Loose capacity-constrained Voronoi diagrams (LCCVD) which allow a controllable assignment of particles to representatives.
- A massively parallel algorithm for computing the time critical operations of CCVDs and LCCVDs on the GPU.
- A quality metric which captures how well particles are covered by representatives.

The remainder of this paper is structured as follows: In Sec. 2 we discuss related work and CCVDs. Sec. 3 gives a description of our LCCVD-based method. Sec. 4 explains the parallel algorithm for the performance-critical part of our method. Sec. 5 introduces the quality metric, which is used in Sec. 6 together with qualitative visual analysis to demonstrate the effectiveness of our approach.

2 Related Work

Capacity-Constrained Voronoi Diagrams Capacity-constrained Voronoi diagrams, briefly CCVDs, have been described by Aurenhammer et al. [2]. CCVDs are Voronoi diagrams where each region's generator point—typically called a *site*—has a predetermined *capacity* which can be understood as the area of a site's Voronoi region weighted with an underlying density function. In discrete spaces, the density function can be represented by a finite set of points which is analogue to the set of particles in our application scenario. An algorithm for the computation of CCVDs has been presented by Balzer et al. [4, 5] who were especially interested in the case where each site coincides with the centroid of its Voronoi region (centroidal CCVD), and where each site has equal capacity. For discrete spaces, this means that each site is assigned the same number of points from the underlying space. To maintain this *strict* capacity constraint, Balzer et al. presented an iterative optimization technique which swaps the assignment of points to sites based on a specified distance function, such that the sum of squared distances from sites to their points converges to a local minimum. This swapping operation is performed by sequentially processing each combination of site pairs which, however, yields an algorithm of quadratic complexity.

Clustering Algorithms Our approach bears similarities to some methods from the field of clustering although the goals of both approaches differ significantly. Applied to our context of spatial point data sets, clustering means the segmentation of a set of points into subsets (clusters) according to proximity. Usually, there are no guaranteed constraints restricting cluster sizes so that an arbitrary number of points could be represented by a single centroid. This means that the original point data set is not guaranteed to be faithfully represented by these centroids at all. For a data set consisting of groups with varying numbers of points (similar to Fig. 2), a standard clustering would detect one cluster for each of these groups and each cluster would simply be represented by its centroid. This way, information about size and shape of the point groups would be lost.

A comprehensive overview on clustering techniques is given by Kolatch [19]. The clustering algorithm most related to our proposed technique is the widely used k-means [13] algorithm. Starting with an initial seed of cluster centroids (sites), k-means iteratively assigns points to its nearest cluster centroid, and then computes a new centroid for each cluster by computing the mean position of all points. The results strongly depend on the initial seeds. More importantly, the number of points assigned to each cluster may differ significantly leading to centroid configurations that do not represent the underlying point density appropriately (Fig.2(a)). There have been attempts to balance k-means [6, 7, 9], but the imposed restrictions either cannot be guaranteed or cannot be chosen freely. Preliminary fixing the number of sites, however, at least determines the average number of points assigned to each site.

Clutter Reduction Techniques Our method uses a set of representatives replacing the original point data set and can thus be considered a clutter reduction technique in the sense of Ellis and Dix [14]. Utilizing their taxonomy, our method can be categorized as an appearance-oriented clustering technique where clustering describes "a different representation of the group of individual lines or points." An alternative technique is based on statistical sampling [11] where representatives are simply picked randomly among the full set of points. We will compare our results to this approach in the evaluation section. In contrast, more general reduction techniques [18, 24] or approaches aiming at surface reconstruction [12, 26] are not geared towards density function adaptation.

Molecular Dynamics Visualization There exists a great variety of visualization tools for particle data sets which differ in focus, performance, and features. The most wide-spread tools for MD visualization are Chimera [10], PyMOL [22], and VMD [27]. Generic visualization packages, such as AVS [3] or Amira [1], also provide special modules for molecular visualization. However, these tools work in the context of bio-chemistry and often lack support for direct particle-based visualization (e.g. with spheres) beyond several tens of thousands of atoms. For larger data sets, they apply visual metaphors from the application domain [23] with less graphical primitives for faster rendering and better perceptibility.

Beyond the context of bio-chemistry, visualizations have to revert to particle-based rendering, which has been recently optimized for data sets with opaque spherical particles up to tens of millions of particles [16], and for transparent data sets from astronomy even up to billions of particles using level-of-detail techniques [15]. However, these visualizations still suffer from cluttered images and lack feasible aid in analyzing time-dependent data. This is usually remedied by applying feature extraction and tracking [28] typically tailored to very specific applications, such as schematic views for nucleation processes [17], mixing layers in hydrodynamics [21], or extraction and visualization of solvent molecules moving paths in proximity of active sizes of proteins [8]. Thus, they cannot be applied directly to arbitrary particle data.

3 LCCVD

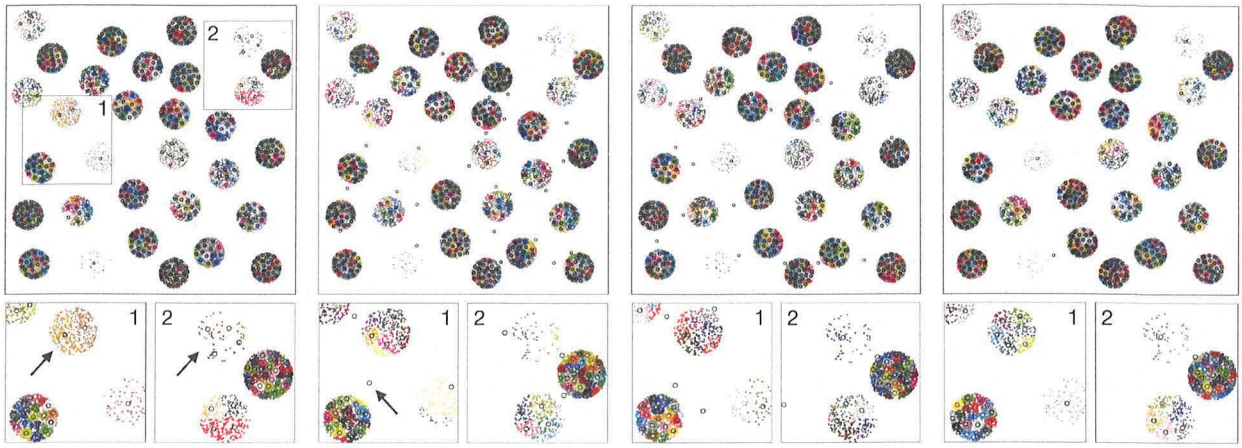
Before introducing our method, consider the case shown in Fig. 2 which demonstrates the shortcomings of existing methods for our application scenario. Here, an inhomogeneous set of particles (*points* from hereon) is to be represented by a smaller set of representatives (sites). Applying strict capacity constraints as proposed by Balzer et al. [4, 5] may result in sites being located inappropriately in-between accumulations of points, making them poor representatives for their sets of associated points. K-means clustering does not share this problem but instead does not allow to draw any conclusions about the underlying point density. This is emphasized by the closeup images where the top groups of points are represented by either too few or too many sites.

We propose *loose* capacity-constrained Voronoi diagrams (LCCVD). Loose capacity constraints mean that the number of points assigned to each site is not fixed, but may reside within an interval $[c_{min}, c_{max}]$. In the following, this is also given in terms of the *capacity looseness* l which translates to the interval by $c_{min|max} = \max(m \cdot (1 \pm l), 1)$, m denoting the average number of points per site. A typical value is $l = 0.2$ such that the capacity interval allows a 20% deviation from m. As such, LCCVD can be seen as a hybrid between the CCVD-based method and the k-means approach, allowing the adjustment of the constraints.

In the remainder of this section, we discuss the basic approach behind LCCVD (see Fig. 3). First, we determine an initial assignment of points to sites (Sec. 3.1). We then exchange points between sites until convergence (Sec. 3.2). When it is acceptable to spend more time on the computation to achieve better results, we perform a step called temporary c_{min} relaxation (Sec. 3.3) which temporarily ignores the minimum constraint to allow an even better adaptation of sites to points (Fig. 2(d)). This is followed by the next phase of point exchange. For time-dependent data sets, the whole procedure is performed for each time step, using the results from the previous time step as initialization to exploit coherency.

3.1 Site Initialization

We use the input point set to determine the initial site positions. For static data sets (and for the first step of a time-dependent series), sites are initially placed at the locations of randomly chosen points. To fulfill its minimum capacity constraint, each site then searches for the c_{min} nearest points that have not yet been assigned to another site. The remaining points are then assigned to the closest site which has not yet reached its maximum capacity constraint



(a) No Constraints (K-Means) $l = \infty$ (b) Strict Constraints (CCVD) $l = 0$ (c) Loose Constraints (LCCVD) $l = 0.15$ (d) Loose Constraints (LCCVD) $l = 0.15$ and c_{min}-relaxation

Figure 2: Two-dimensional point dataset represented by sites using LCCVD. Sites are depicted by black circles while points are shown as colored dots. Different colors depict that points belong to different sites.

c_{max}. The necessary nearest-neighbor queries are efficiently performed using a kD-tree, removing points which have been assigned to sites, or sites which have reached c_{max}, respectively. For subsequent steps of a time-dependent data set, the assignments of points to sites are passed on from the previous time step, and sites are updated using the mean position of the newly assigned points.

3.2 Point Exchange using Loose Capacity Constraints

During the point exchange phase, every site exchanges points with all other sites until convergence. In the strict CCVD-based method this is accomplished by pairwise testing sites for potential point swaps: two points are swapped between a pair of sites only if the sum of squared distances between points and sites decreases. A site is relocated to the new mean position of its points at the end of each swapping process. Using our loose capacity constraints, a point can also simply be re-assigned without substituting it with another point as long as the constraint interval $[c_{min}, c_{max}]$ of a site is not violated. This way we allow points to "switch" to better sites where the strict CCVD-method would have intervened. This user-defined capacity interval allows to span the whole range from the pure distance-based k-means approach ($c_{min} = 0$ and $c_{max} = \infty$) to the strict capacity-constrained approach ($c_{min} = c_{max}$).

The point exchange phase is by far the most expensive part of LCCVD. To this end we introduce an optimized parallel algorithm suitable for GPUs in Sec. 4. We discuss which pairs of sites to consider for the swapping operations (Sec. 4.1 & 4.2), and how to determine point swapping pairs (Sec. 4.3) such that the requirements of an efficient GPU implementation are met.

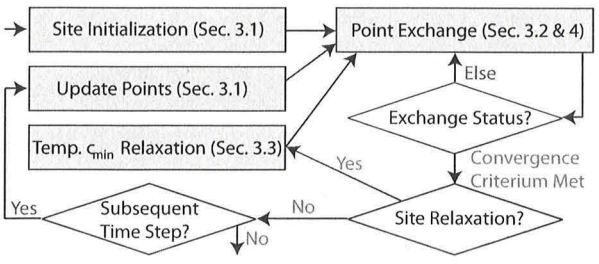


Figure 3: Computation steps and control flow of LCCVD.

Note that data sets from MD simulations usually employ periodic boundary conditions which have to be considered. LCCVD handles periodic boundary conditions when calculating distances or centroids by virtually shifting the data set's bounding box such that the currently considered particle is in its center. Subsequent calculations can then be done in a non-periodic manner.

3.3 Temporary Minimum Constraint Relaxation

One problem from CCVD is partly inherited by our LCCVD approach: sites may get positioned between adjacent point clouds (Fig. 2(c)), making this site a bad representative. This is due to the minimum constraint c_{min} which can prevent that points are removed from sites between two such clouds. Points cannot be swapped to another site either, since other points are even further away. We denote these problematic sites *bad sites* in the following.

We found that temporarily relaxing the minimum constraint for bad sites largely resolves this problem which is why we interpose an optional correction step after each exchange phase (cf. Fig. 3). During this correction step, we perform the following substeps:

1. **Identify bad sites:** A site s is considered a bad site when it is at its minimum capacity and its farthest point p is much closer to any other site s_{other}: $|s_{other} - p|/|s - p| < z$. In our experiments across all our data sets, $z = 0.85$ proved to reliably detect bad sites with only a small amount of false positives.

2. **Assign points of bad site to closest sites:** Release points to closer sites while constantly updating the site position (temporarily violating c_{min} for the bad site). After this step, the bad site only represents the points it is closest to.

3. **Bad site takes on points from nearby sites:** Identify the nearest sites and insert them into a priority queue based on their distance to the bad site. Take the first site from the queue and—as long as its minimum constraint is not violated—reassign points from it to the bad site in the order of proximity. Proceed with the next site from the queue until the bad site has reached its minimum capacity.

We initiate this correction step after each exchange phase (cf. Fig. 3), but no more than five times per time step (this proved to be a good tradeoff between speed and quality). This avoids infinite loops since it is not always possible to resolve a bad site without

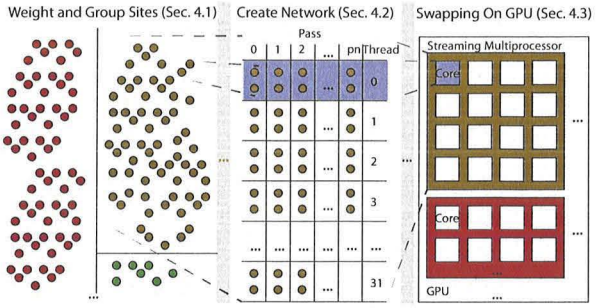


Figure 4: Overview on the LCCVD parallel point exchange. Selected connections between the three steps are indicated by dashed lines.

permanently violating c_{min}. Overall, the temporary relaxation of the minimum constraint allows to reduce the number of bad sites without losing the flexibility of loose capacity constraints.

4 LCCVD Parallel Point Exchange

Exchanging points between sites is the computationally most demanding part of our method and thus its parallelization is crucial for the overall performance. One key observation is that point swaps occur primarily between neighboring sites. This allows us to restrict swapping operations to groups of adjacent sites (Sec. 4.1). These groups can then be processed in parallel by different multiprocessors on a GPU (cf. Fig. 4). To ensure that close sites which do not belong to the same group also get the chance to exchange points, we regroup sites over time—still grouping nearby sites—such that all sites are able to at least once exchange points with all sites in their proximity. Swapping with more distant sites occurs indirectly by successively handing over points from site to site. However, not only full site groups, but also the swapping of points between sites *within* a site group is parallelized in order to fully utilize a GPU (Sec. 4.2). For this purpose, we employ a sorting network that determines the optimal ordering of swapping operations between sites which are then processed by separate GPU threads on a streaming multiprocessor (Sec. 4.3).

4.1 Partitioning Sites into Groups

To determine groups of adjacent sites we enumerate the sites such that the enumeration indices reflect their spatial proximity. For this purpose, we employ a kD-tree based on the set of input points P, because it roughly reflects the points spatial distribution: in densely populated regions, kD-tree nodes (i.e. the centroids of their bounding boxes) are close, in less dense regions they are farther apart. To determine the index i of a site $s \in S$, we search for its enclosing kD-tree node, by traversing the tree (starting from the root with $i = 0$): whenever we descend to the left child, i remains unchanged, whenever we descend to the right child, i is increased by $P \cdot 2^{-h}$ where h denotes the level of the tree ($h = 0$ for the root node). The traversal is stopped as soon as we reach a node that contains $m = |P|/|S|$ points or less. Subsequently, sites are sorted according to i using the in-place GPU radix sorting algorithm of Satish et al. [25] and finally partitioned into groups of consecutive sites.

In order to avoid that sites always belong to the same group, we displace the kD-tree splitting planes in each iteration by applying an offset of a certain direction and magnitude. For the displacement directions, we alternate between the main axial directions and the diagonal directions, while the displacement magnitude for each site—according to our experiments—should be roughly half the extent of the site group's bounding box (as determined without any displacements). Since it is impractical to displace the whole tree with all different group extents along all directions, we consolidate similar displacement magnitudes.

```
for n in sortingNetworkPasses:
  (site0, site1) = swappingNetwork(n, threadId)
  for point0, point1 in sitePoints0, sitePoints1:
    // select point swapping candidates
    weight0 = dist(point0, site0) - dist(point0, site1)
    weight1 = dist(point1, site1) - dist(point1, site0)
    update((maxWeight0, maxPoint0), (weight0, point0))
    update((maxWeight1, maxPoint1), (weight1, point1))
    // take free slots instead of bad candidates
    if maxWeight0 < 0 && freeSlotAvailable(sitePoints0):
      maxWeight0 = 0
      point0 = getFreeSlot(sitePoints0)
    end if
    if maxWeight1 < 0 && freeSlotAvailable(sitePoints1):
      maxWeight1 = 0
      point1 = getFreeSlot(sitePoints1)
    end if
    // swap points if swapping condition is met
    if maxWeight0 + maxWeight1 > 0:
      swap maxPoint0 and maxPoint1
      maxWeight0 = maxWeight1 = -∞
      update site positions
    else
      fill free slots ahead of free slot index when required
    end if
  end for
  (synchronize threads)
end for
```

Listing 1: Pseudo-code for the LCCVD swapping kernel. The distance of a free slot to any site is defined to be zero.

4.2 Swapping Network Construction

While site groups are distributed over the streaming multiprocessors (SM) of a GPU, the swapping operations between sites (within a group) are executed in parallel by each SM. Prior to the swapping algorithm, we create a swapping network that determines which pairs of sites should be processed by a single GPU thread. The network schedules which site pairs are to be processed in parallel, and which are to be processed successively. It needs to ensure that no site is processed in more than one thread at a time to avoid read-write conflicts (or expensive atomic operations). Maximizing the utilization of threads by distributing pairs as evenly as possible is yet another goal. In principle all pairs of sites would have to be considered, but we can significantly prune the set of pairs beforehand by excluding sites which are guaranteed not to swap points according to the following criteria:

1. **Bounding Sphere:** The distance between two sites is larger than the sum of the distances to their farthermost points.

2. **Stability:** Both sites have not exchanged points for N iterations where N denotes the number of displacement directions times the number of displacement magnitude groups.

The bounding sphere criterion is particularly beneficial when sites are roughly at their final position but not yet stable. The stability criterion has a strong impact during the final steps of the optimization when many sites have already reached stable positions.

4.3 Swapping Algorithm

In our GPU implementation, points are stored in an array with c_{max} elements or *slots*. Slots are placeholders for points from the data set such that each site can have at most c_{max} points. Some of these slots are *free slots* in the case that the number of points is smaller than c_{max}. Free slots can be used to re-assign a point to another site without the requirement to take another point in return. In order to determine which points to swap between a pair of sites, the original

CCVD-based method [4, 5] uses a max-heap data structure. Since this is impractical for GPU implementations, our algorithm only keeps track of the point with the largest squared distance to its site. Listing 1 gives pseudo-code for our swapping kernel.

Initially, all points are located at the beginning of the array while free slots are located at the end. The *free slot index* indicates the slot from which no points are stored in the remaining array. A site is able to trade free slots for points as long as the free slot index does not point to the end of the array. When a point is exchanged for a free slot, the free slot is stored in the former location of the point. These free slots form holes and are fixed the next time the algorithm iterates over the array: either the free slot forming the hole is used to store a point of another site, or it is swapped internally with the point just before the current free slot index. The free slot index is subsequently decremented until it marks the first free slot.

5 Point Distribution Preservation Quality Metric

For the numerical comparison between different methods for generating representatives, we propose a metric that captures the quality of the representation of a set of points P by a set of sites S. It is independent of any method-specific information and operates solely on the sets P and S, i.e. without additional knowledge (e.g. assignments of points to sites). As opposed to conventional density estimation and subsequent distance calculation, it is simple and fast to compute, and directly determines the points that each site covers. This is critical for analysis and code debugging purposes.

One basic assumption is that the set of sites offers a good representation for the set of points if their distribution proportionally follows the distribution of the points, i.e. each site should roughly represent an equal amount of points. Since $|S|$ sites have to represent $|P|$ points, each site should represent an average $m = |P|/|S|$ points. This is also reasonable from the user's point of view who expects each representative to be of equal importance. Another assumption is that sites are good representatives for points in their proximity, and less so for more distant points. These assumptions lead to the following metric: a site only covers (i.e. represents) a point if it is amongst its m-nearest points. How well it is covered is determined by means of its distance to the site. If it is inside a certain radius of importance, it counts as fully covered. Beyond this radius, the influence of the site decreases quadratically, such that points that are farther away are only slightly covered by the site.

Note that a metric based on these two assumptions alone is punishing inappropriate site positions—either directly or indirectly. When a certain subset of points is represented by too few sites (as with the k-means example in Fig. 2(a)), the coverage value m will lead to some points not being covered at all. On the other hand, when a certain subset of points is represented by too many sites, the punishment occurs indirectly as there will be a severe lack of sites in other regions. In addition, bad sites (Sec. 3.3, Fig. 2(b) and (c)) are only able to exert a small influence on their surrounding points as most of these will be located outside the site's radius.

We can subsume these assumptions from the perspective of a single point p by computing its coverage quality q_p as:

$$q_p = \min\left(\sum_{s\in N(p,S)} \frac{1}{(|p-s|/r)^2},\ 1\right), \text{ with } r = v\sqrt[3]{\frac{1}{4\sqrt{2}|S|}} \text{ in 3D.}$$

$N(p,S)$ denotes all sites which have p as one of their m nearest points. The radius r is derived from the solution of circle packing [20] such that each site covers as much space as possible without yielding overlapping spheres. In order to approximately adjust to the bounding box domain, r is scaled with the average side length v of the bounding box volume enclosing all points.

The total quality q is given by a normalized sum:

$$q = \frac{1}{|P|}\sum_{p\in|P|} q_p.$$

Sites m	Comp. Time		Norm. Radius α	
Constant	[5]	our	[5]	our
1024 4096	237.9s	129.7s	0.7628	0.7543
2048 4096	451.9s	152.3s	0.7481	0.7451
4096 4096	991.1s	175.6s	0.7470	0.7454
8192 4096	2413.3s	241.6s	0.7455	0.7588
16384 4096	6361.8s	525.7s	0.7367	0.7382
8192 8192	8319.1s	1258.0s	0.7576	0.7588
24576 1500	6720.4s	125.3s	0.7072	0.7035

Sites m	Comp. Time		Cap. Error δ_c	
ρ	[5]	our	[5]	our
1024 4096	214.6s	231.3s	0.00349	0.00346
2048 4096	421.9s	235.2s	0.00291	0.00318
4096 4096	876.6s	338.4s	0.00263	0.00304
8192 4096	1927.0s	542.4s	0.00245	0.00259
16384 4096	4911.7s	857.3s	0.00239	0.00246
8192 8192	6543.7s	2836.5s	0.00204	0.00220
24576 1500	2734.5s	158.7s	0.00333	0.00327

Table 1: Computation times and quality metrics for varying numbers of sites and points per site m using a constant and a non-constant two-dimensional density function. All results were obtained by averaging runs from 10 sets of sites obtained via rejection sampling.

Note that q_p ranges from 0 (unrepresented by surrounding sites) to 1 (fully covered). A site completely covers a point in its radius r, while its influence quadratically decreases beyond that. Full coverage of a point may still be achieved through other adjacent sites.

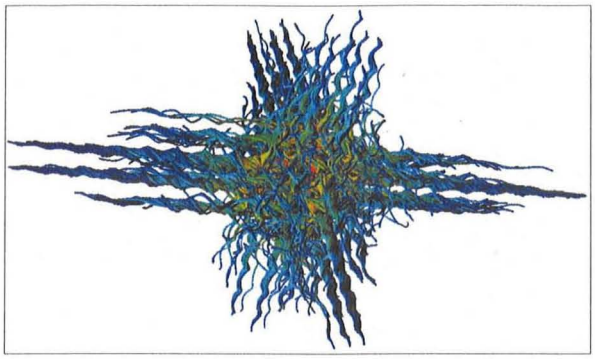
6 Evaluation

This section presents the evaluation of our approach in different scenarios. First, we compare the performance of our method for computing strict CCVDs to the original CPU-based method by Balzer et al. [4, 5] to show the advantages of our parallel algorithm. We then present LCCVD results, including comments from application domain experts, for real-world data sets using 3D molecular dynamics simulation and flow data. All measurements were done using a NVIDIA GTX 480 and an Intel Core i7. The partitioning of sites into groups and the swapping algorithm were implemented in CUDA using a block size of 128 threads and a group size of 128 sites. The remaining computational steps of LCCVD were executed on the CPU using OpenMP.

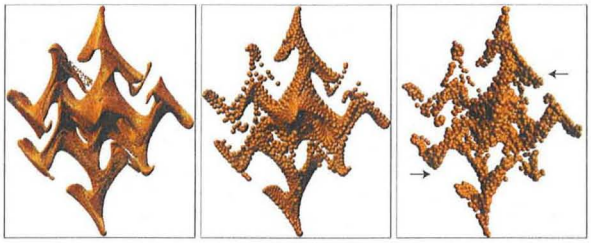
6.1 CCVD of 2D Point Distributions

In their original work, Balzer et al. [5] generated initial 2D point data sets by rejection sampling a given density function. One density function was simply a constant, while an exemplary non-constant density function was chosen as $\rho = e^{(-20x^2-20y^2)} + 0.2\sin^2(\pi x)\sin^2(\pi y)$. Table 1 lists timings and quality results for the quality metrics *normalized radius* α [20] which should be around 0.75, and *capacity error* δ_c [5] which should be close to zero. Both metrics underline that our improved parallel algorithm does not sacrifice the quality of the resulting site distributions.

As expected, our parallel approach becomes more and more beneficial as the number of sites increases. It does not slow down as drastically as the original implementation for large numbers of sites. This is mainly due to the fact that the higher utilization of the GPU cushions the increased computation costs (see Sec. 6.2 for a detailed discussion). Even greater impact can be observed for the number of points per site m: for small m the parallel nature of our approach shows its strength much more clearly and we achieve timings which are faster by an order of magnitude. We attribute this dependency on m to the less sophisticated selection of point swapping candidates which is unavoidable to meet the requirements of an efficient GPU implementation as described in Sec. 4.3.

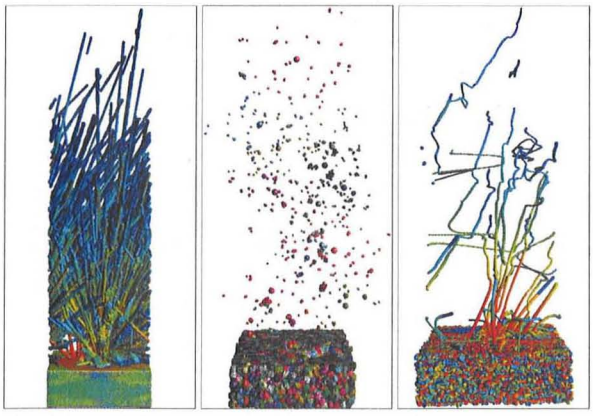


(a) Sites path lines $l = 0.2$, $m = 8192$, $t = (0,400)$



(b) Points, $t = 90$ (c) Sites, $m = 512$, $l = 0.2$, $t = 90$ (d) Sites, $m = 512$, random, $t = 90$

Figure 5: Arnold-Beltrami-Childress flow. *(a)* Data set represented by a set of sites over numerous time steps. *(b)* Full data set for a single time step t = 90. *(c)* Reduced version for t = 90 sites which represents m = 512 points on average. It fully preserves the basic structure and allows better insight into the data set, e.g. the point density at the left is much lower than in the middle or on the right. *(d)* Reduced version based on random sampling exhibits an irregular structure that does not preserve densities and results in the loss of smaller features (e.g. thin structures on the bottom left and top right indicated by arrows).



(a) Points path lines, $t = (0,400)$ (b) Sites, $l = 0.2$, $m = 125$, $t = 400$ (c) Sites path lines, $l = 0.2$, $m = 125$, $t = (0,400)$

Figure 6: Laser ablation from a block of solid aluminum. *(a)* Extracting path lines using the full data set. *(b)* Reduced version for a single time step. *(c)* Extracting path lines from the reduced version. In *(c)* the structure of the molecule movement as well as the amount of molecules being expelled from the block is visualized more clearly.

6.2 LCCVD of 3D Molecular Dynamics Data Sets

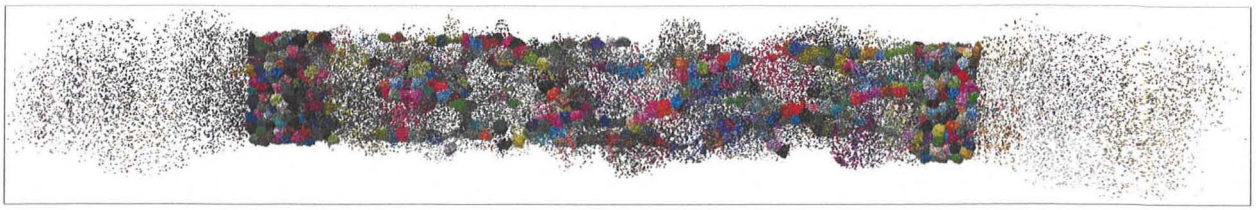
In the application domains of thermodynamics, physics, and material science a direct, particle-based visualization is commonly used to visualize the individual time steps of a simulation. Typically, every particle representing a molecule or atom is rendered as a small sphere. If time series data needs to be visualized, either animations are used to depict the evolution of particles over time, or path lines are rendered for small subsets of particles. Using a set of representatives (sites) instead of the original particles (points) not only reduces both storage requirements and rendering time, but also improves comprehensibility. We demonstrate the effectiveness of our approach by using a data set from particle tracing for vector field visualization—Arnold-Beltrami-Childress (ABC, with $A = \sqrt{3}$, $B = \sqrt{2}$, $C = 1$ and $T = -8$) shown in Fig. 5—and by means of three molecular simulation data sets: laser ablation from a block of solid aluminum (Fig. 6), compressed argon surrounded by vacuum (Fig. 7), and two colliding liquid droplets (methane and ethane) (Fig. 8). Particle numbers and the amount of time steps per data set are listed in Table 2.

Visualizing sites instead of points has numerous benefits apart from rendering speed and storage requirements. For example, Fig. 5(a) illustrates the structure of the flow of the ABC data set. It can be seen that sites move smoothly over time as long as there are no rapid, incoherent movements in the data set. Fig. 5(c) shows that the density in different regions of the data set can be estimated much better with a set of site representatives than with rendering points directly (Fig. 5(b)). It also demonstrates that the basic structure of the data set is preserved even when using a drastically reduced amount of points. Random sampling (Fig. 5(d)) does not preserve the structure of data set as well and results the in the loss of many small features. Occlusion problems are illustrated in the laser ablation example in Fig. 6(a). When rendering all points of all time steps at once, most of the important information remains hidden due to the extensive mutual occlusion of the molecules. Furthermore, it is almost impossible to estimate the amount of molecules being expelled. The reduced version in which one site represents $m = 125$ points illustrates this much better (Fig. 6(c)).

One domain expert concluded that "since attributes like angle, velocity, or cluster size distribution highly depend on the applied laser, a quick way to grasp the ablation process qualitatively (e.g. the opening-angle of the evolving gas plume as can be seen from the reduced trajectories) and without major data post-processing is very useful." Another expert mentioned, more concretely, that "while the basic vertical movement [of the expelled particles] is captured by appropriate color coding, the diagonal movement (which is inherent to the data) is better visible in Fig. 6(c) than in Fig. 6(a)." Similarly, one expert found that for the visualization of the collision scenario "Fig. 1(a) is not useful, and that Figures 1(b) and 1(c) miss the temporal information. Fig. 1(d), however, faithfully captures the main trend of the collision where distant path lines also capture the left droplet's instability." He also stated an analogue visualization could be "very useful for lab-on-a-chip systems where one could estimate where the desired flow is disturbed, i.e. where one would have to adjust the channel structure for undisturbed substance transportation."

In the following examples, we analyze the effect of the looseness parameter l more deeply. First, we show key problems of strict and unrestricted point constraints as occurring with the CCVD and the k-means approach, respectively, and demonstrate that they can be resolved using LCCVD. We then present detailed timings and quality measures based on our quality metric introduced in Sec. 5.

Fig. 7 illustrates that a strict capacity constraint ($l = 0$) potentially forces sites to represent points from two or more dense clusters. This leads to sites floating in-between clusters of points such that they are located where no associated points are (e.g. the purple



(a) Sites, $l = 0.2$, $m = 500$, $t = 70$

(b) Closeup for $l = 0$, $m = 50$ with inappropriate sites highlighted

(c) Closeup for $l = 0.2$, $m = 50$

Figure 7: Argon in vacuum. *(a)* Overview over the reduced version. *(b)* Strict capacity constraints (l = 0) force inappropriate site locations between dense point groups, falsely creating the impression of occupied space. *(c)* Using loose constraints (l = 0.2) largely remedies this problem.

site on the left, or the green site on the right in Fig. 7(b)). Loosening the capacity constraint using a value of $l = 0.2$ and temporarily relaxing the minimum constraint as described in Sec. 3.3 largely avoids these issues (see Fig. 7(c)).

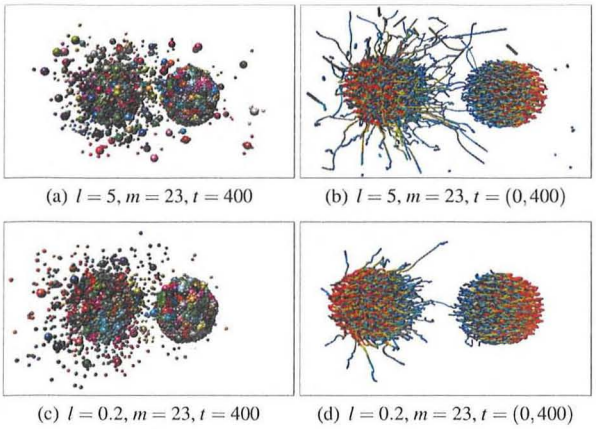
However, too loose constraints may lead to an over- or under-representation of the point density, i.e. regions where a site either represents a too small or too big portion of the data set. Fig. 8 demonstrates overrepresentation for the methane-ethane collision data set with a very loose constraint of $l = 5$ (Fig. 8(a)). Thus, when displaying sites only, the surrounding of the droplets appears much more dense than it actually is. This way, path lines generated with a very loose constraint (Fig. 8(b)) give the impression of a much larger amount of points being spread (Fig. 8(d)).

These observations from the example data sets are underlined by our quality metric for which we present detailed results in Table 2. For each data set, we list both the quality q and the associated computation times while varying the capacity looseness l from $l = 5$ (almost unconstrained) to $l = 0$ (strictly constrained). For better comparability, we omitted the temporary relaxation of the minimum constraint as described in Sec. 3.3 for this test series. Across all data sets, the best results were obtained by applying a loose capacity constraint of $l \approx 0.2$ despite the variations due to different data sets or site configurations; $l = 0.1$ delivered nearly as good results and might be favorable if stricter bounds are required. Note that a difference in the quality metric of 0.001 is equal to the difference of a thousand points being completely covered or uncovered in a data set of a million points. Smaller quality values thus either indicate poorly located sites, or an inappropriate amount of sites covering a particular part of the data set. As demonstrated in the examples, these cases typically occur in regions with a rapid change in point density. In turn, we measure negligible differences for our example data sets for regions of approximately constant density. Quality results for a statistical sampling-based approach (provided for comparison) are typically around ≈ 0.62.

The timing results in Table 2 underline that the computation time for LCCVD strongly depends on the amount of points per site m. The main reason for this is that the GPU load decreases with a decreasing number of sites. For example, a GTX 480 features 15 SMs, each of which can execute two warps concurrently. As each warp processes point swapping operations in groups of 128 sites, any number of sites below $15 \cdot 2 \cdot 128 = 3840$ is not able to fully utilize the GPU. In order to hide latencies, the actual number of sites

Capacity Looseness					
$l = 5.0$	$l = 0.5$	$l = 0.3$	$l = 0.2$	$l = 0.1$	$l = 0$
Argon in vacuum, 2000000 points, 100 time steps, periodic					
4000 sites, $m = 500$ points per site (random sampling: .61829)					
.02709	.00549	.00159	.00006	**.86951**	.01683
11162.6s	17157.7s	16371.1s	15289.4s	14469.7s	14028.0s
20000 sites, $m = 100$ points per site (random sampling: .62090)					
.03593	.00806	.00182	**.86262**	.00040	.02100
9005.7s	7476.3	7374.2s	7306.4s	7012.6s	6412.8s
40000 sites, $m = 50$ points per site (random sampling: .62269)					
.04639	0.00957	.00183	**.85439**	.00073	.02050
9003.1s	10038.3s	10159.0s	9986.1s	9846.4s	9572.3s
Laser Ablation, 562500 points, 400 time steps, periodic					
1125 sites, $m = 500$ points per site (random sampling: .63140)					
.00723	.00234	0.00057	**.89489**	.00136	.02374
10545.7s	10066.8s	9419.4s	8753.3s	9460.2s	4559.7s
11250 sites, $m = 50$ points per site (random sampling: .63249)					
.03189	.00698	.00190	**.89575**	.00026	.03704
4497.3s	4529.9s	3801.6s	4476.6s	4438.1s	5954.0s
Methane-Ethane Collision, 81672 points, 1782 time steps, periodic					
3403 sites, $m = 24$ points per site (random sampling: .61536)					
.17199	.019660	.00369	**.85184**	.00149	.07216
1586.8	1566.4	1575.2	1575.0	1984.2	1476.4
1992 sites, $m = 41$ points per site (random sampling: .61234)					
.17063	.02065	.00299	.00007	**.84938**	.02328
1894.3s	1847.6s	2228.1s	1865.8s	1837.3s	1633.8s
ABC, 2097152 points, 400 time steps					
16384 sites, $m = 128$ points per site (random sampling: .62235)					
.02554	.01221	.00335	**.85882**	.00018	.01205
4002.4s	4195.0s	4178.9s	4135.7s	3974.0s	4217.0s
4096 sites, $m = 512$ points per site (random sampling: .62183)					
.06424	.02968	.01034	.00307	**.86803**	0.00810
6566.4s	8393.2s	8180.8s	7886.8s	7027.9s	5124.9s

Table 2: Performance of LCCVD for different data sets, loose constraints l and no c_{min} relaxation. The top rows depicts the best (meaning largest) quality results in bold while the other results are given as the difference to this reference value. The bottom rows give the computation times in seconds. Additionally, quality results for random sampling are provided for comparison.



(a) $l = 5, m = 23, t = 400$ (b) $l = 5, m = 23, t = (0,400)$

(c) $l = 0.2, m = 23, t = 400$ (d) $l = 0.2, m = 23, t = (0,400)$

Figure 8: Collision of methane and ethane. *(a, b)* Very loose or no capacity constraints lead to points that are highly overrepresented by sites which gives the false impression of a substantial amount of particles in the outer regions. *(c, d)* A loose constraint of l = 0.2 yields a much more genuine result.

should even be significantly higher since warps may be paused or stalled. In contrast to the number of points per site m, the capacity looseness l only has minor influence on the runtime.

Lastly, we measured the effect of the temporary c_{min} relaxation compared to the best quality values listed in Table 2. In general, the technique is most beneficial for data sets which induce the generation of bad sites—e.g. due to multiple groups of points of varying density—as discussed in Sec. 3.3. This particularly applies to the argon in vacuum data set (Fig. 7). In this scenario, the quality q can approximately be improved by 0.01 for $m = 50$. At the same time, however, the execution time is almost tripled to 28409s. For data sets with significantly less bad sites, e.g. the laser ablation data set, the quality improvement is only about .001 on average at roughly twice the execution time. The coverage quality of the methane-ethane collision data set with 3403 and 1992 sites increases by .00153 and .00055 respectively, while the runtime roughly doubles. Furthermore, we observed that the processing time as well as the quality value achieved for a single time step is largely independent of whether it has been computed as part of a time series or individually. In some cases, however, time steps which are part of a series are processed significantly faster if the changes between two subsequent time steps are rather small. In such a case, the site-to-point assignment of the new time step only requires minor adjustments compared to the previous step in the series.

7 Conclusion and Future Work

We presented a novel technique for particle-based visualizations that uses a set of representatives instead of a large number of particles. To obtain these representatives, we introduced loose capacity-constrained Voronoi diagrams and presented a fast, parallel method for their computation. We demonstrated that the representatives faithfully capture the underlying particle density and exhibit coherent movement for time-dependent simulations. Using these representatives, we are able to generate sparse yet concise renderings with spheres and path lines in the context of different application domains. For future work, we plan to compare our metric results to traditional density based techniques. We also would like to investigate the usage of LCCVD to build hierarchical structures from large point data sets (e.g. for LOD techniques).

Acknowledgments

The authors thank Filip Sadlo and Marcel Hlawatsch for their support with the ABC data set. This work is partially funded by Deutsche Forschungsgemeinschaft as part of SFB 716 project D.3 and the Cluster of Excellence in Simulation Technology.

References

[1] Amira. http://www.amiravis.com/.

[2] F. Aurenhammer, F. Hoffmann, and B. Aronov. Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20:61–76, 1998.

[3] AVS. http://www.avs.com.

[4] M. Balzer and D. Heck. Capacity-constrained Voronoi diagrams in finite spaces. In *Proceedings of the Symposium on Voronoi Diagrams in Science and Engineering*, pages 44–56, 2008.

[5] M. Balzer, T. Schlömer, and O. Deussen. Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 28(3):86:1–8, 2009.

[6] A. Banerjee and J. Ghosh. On scaling up balanced clustering algorithms. In *In Proceedings of the SIAM International Conference on Data Mining*, pages 333–349, 2002.

[7] S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. 2008.

[8] K. Bidmon, G. Reina, F. Bös, J. Pleiss, and T. Ertl. Time-Based Haptic Analysis of Protein Dynamics. In *Proceedings of World Haptics Conference (WHC 2007)*, pages 537–542, 2007.

[9] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, Microsoft Research, 2000.

[10] UCSF Chimera. http://www.cgl.ucsf.edu/chimera/.

[11] A. Dix and G. Ellis. by chance enhancing interaction with large data sets through statistical sampling. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 167–176. ACM, 2002.

[12] X. Du and Y. Zhuo. A point cloud data reduction method based on curvature. In *IEEE 10th International Conference on Computer-Aided Industrial Design*, pages 914–918, 2009.

[13] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. Wiley New York, 1973.

[14] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1216–1223, 2007.

[15] R. Fraedrich, J. Schneider, and R. Westermann. Exploring the millennium run – scalable rendering of large-scale cosmological datasets. *IEEE Trans. on Vis. and Comp. Graph.*, 15:1251–1258, 2009.

[16] S. Grottel, G. Reina, C. Dachsbacher, and T. Ertl. Coherent Culling and Shading for Large Molecular Dynamics Visualization. In *Eurographics/IEEE Symposium on Visualization*, 2010.

[17] S. Grottel, G. Reina, J. Vrabec, and T. Ertl. Visual Verification and Analysis of Cluster Detection for Molecular Dynamics. In *Proceedings of IEEE Visualization '07*, pages 1624–1631, 2007.

[18] G. Guo, H. Wang, D. Bell, and Q. Wu. Data reduction based on spatial partitioning. In *Computational Science - ICCS 2001*, volume 2074, pages 245–252. 2001.

[19] E. Kolatch. Clustering algorithms for spatial databases: A survey, 2001.

[20] A. Lagae and P. Dutré. A comparison of methods for generating Poisson disk distributions. *CG Forum*, 27(1):114–129, 2008.

[21] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1053–1060, 2006.

[22] PyMOL. http://pymol.sourceforge.net/.

[23] J. S. Richardson. The anatomy and taxonomy of protein structure. *Advances in protein chemistry*, 34:167–339, 1981.

[24] Y. Sang, Z. Yi, and J. Zhou. Spatial point-data reduction using pulse coupled neural network. *Neural Process. Lett.*, 32(1):11–29, 2010.

[25] N. Satish, M. Harris, and M. Garland. Designing efficient sorting algorithms for manycore GPUs. In *IEEE International Symposium on Parallel & Distributed Processing*, pages 1–10, 2009.

[26] W. Song, S. Cai, B. Yang, W. Cui, and Y. Wang. A reduction method of three-dimensional point cloud. pages 1–4, 2009.

[27] Visual Molecular Dynamics. http://www.ks.uiuc.edu/Research/vmd/.

[28] T. Weinkauf, H. Theisel, J. Sahner, and H.-C. Hege. UFEA: Unified Feature Extraction Architecture. In *Proceedings of TopoInVis 2009*.