Visual Verification and Analysis of Cluster Detection for Molecular Dynamics

S. Grottel, G. Reina, J. Vrabec and T. Ertl

Abstract— A current research topic in molecular thermodynamics is the condensation of vapor to liquid and the investigation of this process at the molecular level. Condensation is found in many physical phenomena, e.g. the formation of atmospheric clouds or the processes inside steam turbines, where a detailed knowledge of the dynamics of condensation processes will help to optimize energy efficiency and avoid problems with droplets of macroscopic size. The key properties of these processes are the nucleation rate and the critical cluster size. For the calculation of these properties it is essential to make use of a meaningful definition of molecular clusters, which currently is a not completely resolved issue.
In this paper a framework capable of interactively visualizing molecular datasets of such nucleation simulations is presented, with an emphasis on the detected molecular clusters. To check the quality of the results of the cluster detection, our framework introduces the concept of *flow groups* to highlight potential cluster evolution over time which is not detected by the employed algorithm. To confirm the findings of the visual analysis, we coupled the rendering view with a schematic view of the clusters' evolution. This allows to rapidly assess the quality of the molecular cluster detection algorithm and to identify locations in the simulation data in space as well as in time where the cluster detection fails. Thus, thermodynamics researchers can eliminate weaknesses in their cluster detection algorithms. Several examples for the effective and efficient usage of our tool are presented.

Index Terms— Cluster detection analysis, molecular dynamics visualization, time-dependent scattered data, glyph visualization, out-of-core techniques, evolution graph view

1 Introduction

The nucleation of vapor into liquid phase is still a research topic for thermodynamics researchers and is being investigated at the molecular level with the help of distributed simulation runs. Starting with datasets at a homogeneous state point in the vapor, either the pressure is increased or the temperature is decreased so that a metastable state point in the two-phase region is reached. The liquid phase emerges through spontaneous density fluctuations in the vapor which lead to the formation of molecular clusters. These predecessors of liquid droplets reach a macroscopic size through aggregation of individual molecules or coalescence with other clusters. The process is driven by the tendency of physical systems to minimize its Helmholtz energy. Such condensation is found in many physical phenomena, e.g. the formation of atmospheric clouds or the processes inside steam turbines, where the expansion of the water vapor to metastable state points is desirable for maximum energy efficiency. However, water droplets of macroscopic size damage turbine blades, so that detailed knowledge of the dynamics of condensation processes is of great interest [9].

The key properties of these processes are the nucleation rate and the critical cluster size. The critical cluster size designates clusters with a certain number of monomers that have the same probability of growth as of decay; smaller clusters will more likely evaporate and bigger clusters will more likely continue to grow. The nucleation rate quantifies the number of emerging clusters beyond the critical size per volume and time.

Commonly relied-on basic principles, e.g. the classical nucleation theory [35] for the dynamics of the condensation process, do not yield satisfactory results when predicting nucleation rates. The classical nucleation theory works well in some cases, but fails in many others [9] and predicts values which deviate several orders of magnitude from experimental findings [8]. The situation is even worse if mixtures of several substances are concerned, e.g. in process industry. Further problems arise if it is also considered that there is no perfect thermal equilibrium between the vapor and the short-lived clusters of the emerging liquid [28].

Molecular dynamics simulation is a powerful tool that helps to overcome the shortcomings of the classical approach as it allows for detailed insight into the processes that are triggered on the molecular level. This also holds for other phase transitions like melting and freezing [13]. Recently, adequate models for the intermolecular interactions have become available [36, 29], which contain all the relevant thermodynamic properties of real fluids. Thus, it can be expected that these models can predict nucleation rates much more accurately. However, simulations of nucleation processes must use large numbers of molecules to get meaningful results while yielding cluster sizes up to thousands of molecules and more. Furthermore, low nucleation rates can only be detected with very large systems, as they indicate a rare event in large volumes. Large systems are also needed to more closely approximate real-world conditions in most technical condensation processes and thus to ensure comparability and verifiability against real-world measurements. With the advent of commodity hardware PC clusters, molecular dynamics simulation of large systems has become more cost-effective and can also be utilized by researchers on-site. The resulting datasets contain several hundreds of thousands molecules and trajectories of thousands of time steps. We will refer to such a time step as a *configuration* in this paper since this is the term used in molecular thermodynamics[1]. Any software used for visualization or analysis needs to be capable of handling such large datasets.

It is, however, even more important to verify the correctness of a simulation and the employed clustering criterion than just having access to interactive visualization. For the calculation of the nucleation rate and the critical cluster size, it is essential to make use of a meaningful definition for molecular clusters because only if the clusters are correctly detected, the corresponding metrics will yield correct results. One of our main concerns therefore is the verification of the clustering results with the aid of interactive visualization in addition to the comparison of the resulting nucleation rates with experimentally acquired values. By observing the detected clusters and their interaction

- *S. Grottel, Institute for Visualization and Interactive Systems, Universität Stuttgart, E-mail: grottel@vis.uni-stuttgart.de.*
- *G. Reina, Institute for Visualization and Interactive Systems, Universität Stuttgart, E-mail: reina@vis.uni-stuttgart.de.*
- *J. Vrabec, Institute of Thermodynamics and Thermal Process Engineering, Universität Stuttgart, E-mail: vrabec@itt.uni-stuttgart.de.*
- *T. Ertl, Institute for Visualization and Interactive Systems, Universität Stuttgart, E-mail: ertl@vis.uni-stuttgart.de.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

with the surrounding monomers, problems (and bugs) in the employed molecule cluster detection algorithm can be isolated. This becomes even more important when observing how the clusters evolve over time and how they interact with other clusters. Our software provides several visualization modes emphasizing these aspects. The molecular clusters can be represented by single ellipsoid glyphs to enhance the perceptibility of the cluster. The monomers can be filtered to only show those interacting with the clusters. The flow of molecules between clusters can be abstractly represented by *flow groups* that are presented in this paper.

These visualization modes only present qualitative information about the dataset, but are not sufficient to obtain quantitative values in the identified regions in space and time where the cluster detection possibly failed. Therefore, we coupled the molecule view with a schematic graph view representing information about the evolution of the detected molecular clusters over time as well as the molecule flow between these clusters. To reduce clutter when displaying the molecule flow, we use synchronized selection in both views and filter the graph display based on that selection. A user can thus easily get more information about the detected clusters, their evolution over time and their interaction with each other and with the surrounding monomers, allowing a fast analysis of the found situation. The flow groups representing the molecular flow between two clusters can indicate potential classification errors. In the visualization of the molecular clusters the flow groups appear as arrows of defined sizes and positions, so that they can easily be compared to molecule cluster ellipsoids. In the schematic view the flow is visualized as lines connecting the different molecule clusters, which helps tracking the potential classification problems over time. Using both views, the user can benefit from an effective and efficient workflow for the assessment of the molecular cluster detection results and for the identification of potential weak spots in these algorithms.

The remainder of this document is structured as follows: In Section 2 we summarize related work. The point-based visualization of the molecule datasets will be described in Section 3. The subsections will, in addition, describe the data structure internally used and the representations of molecular clusters and monomers used to emphasize the molecule cluster evolution and problems of the molecular cluster detection algorithms. Section 4 presents the schematic view for the molecular cluster evolution. In Section 5, we present the results of our system. These include rendering performance measurements and the exemplary analysis of different molecular cluster detection algorithms. We conclude with an outlook on future extensions in Section 6.

2 Related Work

Scientists who make use of molecular datasets for structural analysis or simulation runs face a plethora of tools of varying complexity for the visualization of such data. Among the most widely known applications are Pymol [7], Chimera [25], Rasmol/Protein Explorer [22], Molscript [20], and VMD [16]. Some of these have not seen updates in several years, for example Molscript and Rasmol, while the more popular ones are still under development. More recently, a number of simple java-based viewers have been created for integration into web sites of scientific databases, but these tools are not suited for large datasets or time-dependent data at all because of performance and memory issues. Popular commercial applications like Amira [2] are nowadays offered with specialized modules for molecular visualization as well. In most of these tools, the user has a good set of options for displaying the datasets and can choose from many different drawing styles for molecules with varying information density. Chimera, for example, also offers the means for performing analyses and measurements on the data. This flexibility, however, usually comes at the cost of suboptimal performance for large or time-dependent datasets. Carefully optimized viewers like TexMol [3], BioBrowser [11], or the tool presented in [23] offer far better interactivity and performance since they rely heavily on GPU-optimized, mainly point-based, rendering techniques. Such point-based rendering of large datasets has been quite an active area of research in the last few years. Many algorithms deal with the point-based rendering of extremely large geometry, or point set surfaces, first using the CPU [4, 39], and more recently using programmable graphics hardware [27, 24, 34]; others concentrate on particle-based data itself, as seen in [14]. Since the rendering of simple points neglects additional attributes and since isotropic representations lack the conveyance of particle orientation, more complex primitives have been introduced, for example ellipsoids [19, 10] or *dipole glyphs* [26]. Current programs like Qutemol [33] not only offer good rendering performance, but also enhance the perceptibility using ambient occlusion and edge cueing. However, Qutemol lacks support for time-dependent datasets. A singular approach using a mixed environment of a graphics workstation for rendering and a PC cluster for scene optimization has been presented in [30], but this also offers only few rendering styles. Furthermore, all of these optimized tools lack the advanced features of VMD and Chimera as well as any support for time-dependent data. VMD and Chimera can display time-dependent datasets, but have to load the whole dataset into system memory. Many of the simulation datasets available to us are several gigabytes in size, on the one hand because of the high number of configurations they contain (several hundreds to thousands usually) and on the other hand due to the high number of molecules contained (tens to hundreds of thousands), and thus cannot be loaded completely into memory on 32 bit-systems. There is no tool that we know of supporting the tracking and visualization of specific aspects related to the nucleation process. None of these tools includes criteria for detecting clusters or allows the interpretation of the cluster membership of loaded molecules to offer visualization methods that help to interpret the nucleation process in gases.

The visualization of flow has been tackled recently with mostly texture-based, dense representations. The texture-based approach was introduced with LIC [5] and has seen many improvements and hardware-accelerated implementations [18] since, even on 3D surfaces [37] and for 3D fields [38, 17, 32]. Since an integral part of the information we need to visualize consists of the droplets themselves, a dense representation could not meet our expectations as it would clutter the rendering too much. Therefore, we worked on creating a simple, sparse visualization which emphasizes the aspects of molecular cluster detection, stability of the detection results, and interactions between clusters and between clusters and monomers, which are the aspects the thermodynamics researchers are most interested in.

Furthermore, we do not know of any tool providing special functionality for clustering algorithm verification, coupled with molecular visualization. There are tools helping to analyze other problems, like the system presented in [21], which visualizes turbulences of the mixing layer between two fluids of different densities. Their application presents a three dimensional rendering of surfaces of the mixing layer in combination with a schematic view of features (bubbles) of the mixing turbulences, which they refer to as merge-split graph, visualizing important events in the features' evolution like births, merges, splits and deaths. Our system with a coupled molecule rendering and a schematic view of the evolution of the detected molecular clusters is similar to this approach.

3 Molecule Visualization Techniques

The goal of our work was to aid the thermodynamics researchers in analyzing the molecular dynamics datasets of their nucleation simulations, in particular the detected molecular clusters and thus the quality of the employed molecular cluster detection algorithm. To achieve this goal, our software needs to be able to render large molecule datasets with long trajectories at interactive frame rates, allowing an exploration of the datasets to search for possible clustering problems. We do not only have to render the molecules but also derived data, like flow groups and ellipsoids representing molecular clusters. While some of these derived data can be calculated for each configuration independently, the calculation of others need information from past and future configurations as well. Therefore, we have to perform a preprocessing step on the datasets before visualizing them, which is presented in Section 3.1 along with the data structures we create. Our time-dependent visualization interpolates between the configurations to achieve a visually continuous output in order not to distract the user by abrupt changes. When the playback is paused, time snaps to the nearest configuration to present unaltered simulation data.

The direct point-based rendering of the molecule dataset will be discussed in Section 3.2. To further emphasize the molecular clusters we use ellipsoids representing the clusters instead of rendering the molecules forming the cluster. The evolution of clusters is visualized using color coding, which allows a qualitative classification of the clusters. This approach is presented in Section 3.3. However, to make a valid evaluation of the cluster detection algorithm, it is insufficient to only review the molecular clusters. The interaction between clustered molecules and the monomers is also an important issue and is discussed in Section 3.4.

3.1 Data Structure

To calculate all derived data features which require information from more than just the current configuration, we employ a preprocessing step. In addition, this gives us the opportunity to create a more suitable data structure for the point-based rendering, which we use to visualize the molecules, as well as supporting different input file formats. The preprocessing is done by a separate program which generates a proprietary binary file storing all data, including additional information like the molecular cluster ellipsoids, flow groups, values for filtering monomers, and file offsets for seeking within the trajectory. Although each dataset has the same number of molecules in every configuration, these file offset tables are necessary for seeking because the sizes of the configurations in one file are not constant due to the varying number of clusters and flow groups.

The molecule positions can be stored directly, or they can be placed in a positional hierarchy using relative quantized coordinates as presented in [14]. In addition, we use this hierarchy to classify the molecules to allow for interactive switching between the different visualization modes. The direct children of the root node are used for this classification. Hence, we have one node holding all monomers as children, one node holding all molecular clusters, and so on. Molecular clusters are represented by inner nodes of this hierarchy, which store all information about the ellipsoidal shape of the molecular cluster, and holding all contained molecules as children.

While the PCIe graphics bus is fast enough to transfer the uncompressed float data, this is not true for the transfer of the data from hard disks. Our datasets have trajectories of several thousand configurations and are thus several gigabytes in size. Therefore, we still allow the use of the positional hierarchy of quantized relative coordinates because this gives us an acceptable compression rate of the dataset without significantly increasing the workload on the CPU. The decompression can be partially performed on the GPU by uploading the quantized coordinates of the molecules and the full precision position of the parent nodes to the graphics card. Quantizing the position using bytes reduces the file size to 70%, because the storage required for additional attributes like quaternions, radii, and colors is not affected. These are quantized to bytes independently. However, using single bytes yields a relative positional error for the individual molecules of about 3%, which is not acceptable. To reduce this error, a much larger hierarchy would be needed, with each leaf node only storing the data of a small group of molecules, which is also not acceptable since this would reduce rendering performance as only very small vertex arrays could be used. Utilizing shorts instead of bytes results in an compression to 80%, but only introduces a relative error of about 0.01%, which is sufficiently small. The effective space saving seems moderate, however it only applies to positional information, which only makes up 25% of a dataset (or 33.3% when no quaternions are present). These factors are applicable to all of our datasets (e.g. a 14.2 GB dataset with float coordinates is compressed to 9.86 GB using bytes or 11.3 GB using shorts). We currently do not use additional compression like *zlib* because this would increase the workload on the CPU too much.

The use of the hierarchy, however, causes problems when interpolating the positions of the molecules. Either the hierarchy must be constant over the complete trajectory (similar to the approach in [15]) or the interpolation of the positions must take place between the relative coordinates of different hierarchies. When considering cyclic boundary conditions resulting in molecules traversing the whole simulation domain between two consecutive configurations, it becomes obvious that a single hierarchy is not applicable to our datasets. Another problem is that we implicitly encode cluster membership through the hierarchy, representing clusters by inner nodes. Therefore, we use one hierarchy per configuration and adjust the coordinates of two consecutive configurations at loading time. In memory each configuration stores start and end positions in relative, and optionally quantized, coordinates in its single hierarchy. So these configurations no longer represent the simulation state at a discrete point in time, but the interpolated simulation states over a short period of time. This must be taken into consideration by the preprocessor when constructing the hierarchy, and some special cases exist when molecules cross the boundary of the simulated area since usually cyclic boundary conditions are used.

The large file sizes of the datasets are the reason why many visualization tools mentioned in Section 2 fail to load the datasets and to render the molecules at interactive frame rates. We implemented an *out-of-core* data streaming to overcome this problem. Our software calculates the memory footprint of a single configuration of the dataset to load. Because only the number of molecular clusters and flow groups change over time, an approximated value can be calculated by examining the first and the last configuration of the dataset. Considering this memory footprint and the amount of memory available, several buffers are created, used to store the current configuration and to prefetch configurations which will be needed next when the trajectory is played back as animation, controlled by a priority queue and processed by a second thread. The offset information for all configurations in our file format allows the user to play back the trajectory as animation forward and reverse at any speed, and to jump to any configuration interactively.

The two derived data values calculated in the preprocessor which either need information from future or past configurations, or which affect past configurations, are the *cluster time distance* and the *flow groups*. To define these, we first need some definitions of the elements of our datasets: $\mathbf{T} = \{0, ..., t_{max}\} \subset \mathbb{N}_0$ is the time line of the dataset with the discrete configuration times $t_i \in \mathbf{T}$, and $\mathbf{M}$ is the set of all molecules in the dataset with the trajectory of molecule $m_i \in \mathbf{M} : t \mapsto (x_i(t), y_i(t), z_i(t))^T$

Molecular clusters are identified by a natural number. The value zero is used to represent molecules that are not part of a cluster. The cluster membership of a molecule m_i can be defined as:

$$c : \mathbf{M} \times \mathbf{T} \to \mathbb{N}_0, (m_i, t) \mapsto c(m_i, t) \tag{1}$$

A molecular cluster can then be defined as the time-dependent set of the contained molecules:

$$\mathbf{S}_j(t) := \{m_i \in \mathbf{M} | c(m_i, t) = j \wedge j \neq 0\} \tag{2}$$

The cluster time distance is a signed value for each molecule, representing the distance in time to the next cluster:

$$ctd(m_i, t) = \begin{cases} t_c - t & \text{if } \exists t_c \forall t_x : |t_c - t| \leq |t_x - t| \\ & \text{with } t_c, t_x \in \{t_j \in \mathbf{T} | c(m_i, t_j) \neq 0\} \\ nan & \text{else} \end{cases} \tag{3}$$

The symbolic constant *nan* is used to indicate that a molecule is never part of a cluster and that the set used in the first case therefore would be empty. Since this set needs to know the cluster membership information of the molecule from all configurations, it is obvious that these values cannot be calculated in a single pass, without rewriting past configurations. The same is true for the flow groups. A flow group is a set of molecules $\mathbf{M}_f$ moving together from one cluster to another:

$$\begin{aligned} \mathbf{M}_f(e, v, t_e, t_v) := \{ & m \in \mathbf{M} | c(m, t_e) = e \wedge c(m, t_v) = v \\ & \wedge \forall t \in (t_e, t_v) : c(m, t) = 0\} \\ & \text{with } e \neq v \text{ , } t_e < t_v \end{aligned} \tag{4}$$

So the flow group is defined by the configuration time t_e of its emergence, the ID e of the cluster it is leaving, the time t_v of its disappearance, and the ID v of the cluster it is joining. We have developed this concept in cooperation with our thermodynamics partners to investigate the interexchange of molecules between clusters. We applied this approach in the context of this paper to evaluate the stability of different molecular clustering criteria. The preprocessor cannot detect such a group until the molecules are joining with a new cluster, which takes place in the configuration when the flow group vanishes.

3.2 Molecular Glyphs

One of the core components of our software is the interactive high-quality rendering of the time-dependent molecular datasets, allowing the user to explore the dataset in space and time. Previous works introduce GPU-based glyphs created by raycasting an implicit surface, represented by only few parameters (position, color, size, radii, orientation quaternion), uploaded as an attributed point to the graphics hardware. As basic objects we use hardware shaders for spheres, dipole-glyphs, which are also used to represent unpolarized two-center molecules, and ellipsoids. Since raycasting of such implicit surfaces, including the calculation of optimal point sprite boundaries, in *vertex* and *fragment shaders* is no new technique, we refer to [26] and [19] for detailed information. Of course, all three shaders calculate correct perspective distortion and are enabled for stereo rendering. To be able to mix these objects with each other, we adapted the shader code to write OpenGL-conforming depth values by projection using the built-in matrices. The color of the glyphs is normally used to identify the different types of molecules and their clustering (see Figure 1). The user specifies two colors for each molecule type, one for monomers and one for clustered molecules. Section 3.4 will show some alternative mappings.

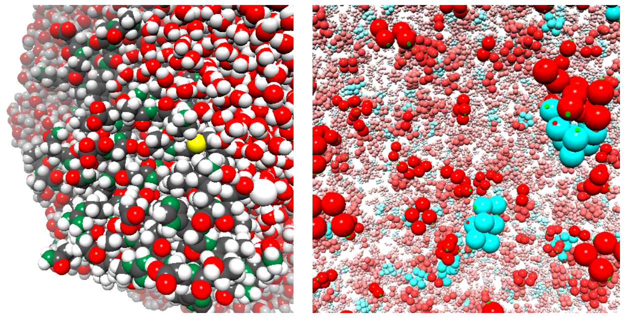
Fig. 1. The left image shows the TEM-1 dataset using spheres, color-coding the atom type. The right image shows the R-152a dataset using dipole glyphs and color-coding whether the molecule is a monomer (red) or part of a cluster (blue). Molecules farther away from the viewer are faded out to enhance the depth perception.

Our molecule rendering is a combination of these glyphs with the out-of-core data streaming mechanism described in Section 3.1. The framework can also be used in virtual environments, using stereo output devices like multiple projectors powered by a graphics cluster. To further enhance the perception, we provide depth cues by fading the glyphs according to their distance to the camera (shown in the right image of Figure 1).

However, like our screen shots in this paper might show, viewing all molecules results in heavily cluttered images. Effects and situations inside the simulated area can hardly be observed. This visualization mode is only good for qualitatively examining the dataset, and in later configurations for identifying regions in space and time where interesting effects take place, but the images are insufficient for judging the molecular cluster detection algorithm or identifying problems of this algorithm. To overcome this problem, the data must be filtered and the molecular clusters as features of interest must be emphasized.

3.3 Visual Representation of Molecular Clusters

The first step to emphasize the molecular clusters is to change their visual representation from multiple molecule glyphs to a single graphical object with roughly the same shape. Since the molecular clusters represent droplets, the surface tension will bring them into a nearly spherical shape as soon as the cluster reaches a sufficiently large size. So representing the clusters as ellipsoids is an obvious choice. Using shaded ellipsoids, the user can perceive the position and extent of the molecular clusters more easily, even if a large number of monomers occlude portions of the ellipsoid as shown in Figure 2. The ellipsoids can also be shown as a transparent overlay over the monomers contained in a cluster (see Figure 7, right image).

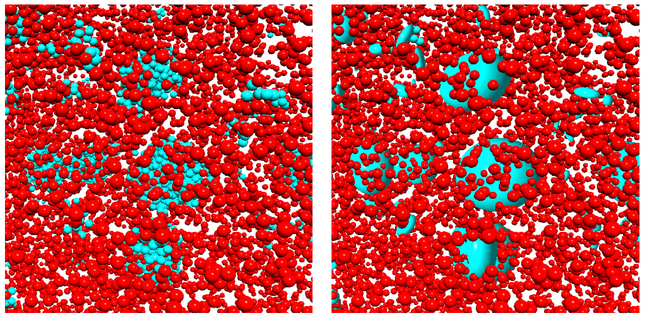
Fig. 2. Molecular clusters are normally represented using a simple color coding (left). Using ellipsoids allows to easier perceive the shape of the clusters (right).

The ellipsoid parameters are calculated similarly to [31]. The position $e_j(t) = (\bar{x}(t), \bar{y}(t), \bar{z}(t))^T$ of the center of the ellipsoid is the average position of all molecules forming the cluster. The main axes are determined using the eigenvectors of the covariance matrix. The eigenvalues and eigenvectors of the covariance matrix are then calculated and sorted according to the eigenvalues. The normalized eigenvector of the largest eigenvalue is used as first main axis. The other two main axes are then calculated using the eigenvector of the second largest eigenvalue and cross products. The radii of the ellipsoid are given by the projection of the contained molecule positions onto the main axes. The resulting ellipsoid will miss some border molecules, but this number is negligible since we can safely assume roughly spherical shape, as mentioned earlier.

Interpolating between two ellipsoids is not straightforward. First, the ellipsoid is symmetric, so two axes can be flipped without the shape changing. Second, and this is the more critical aspect, interpolation can either minimize the scaling or minimize the rotation of the ellipsoid. The first approach will interpolate between the radii of the ellipsoid, which results in the biggest radius remaining the biggest radius. The orientation quaternions are then directly interpolated using *SLERP* [6]. The second approach will select pairs of axes such that the rotation is minimized. The orientation quaternion of the targeted configuration must then be recalculated, which can be performed at loading time. However, the differences between these two interpolation methods are only visible if very unstable clusters are used, and then both approaches result in rather strange animations, either spinning or wobbling. So we decided to interpolate the quaternions directly, without any recalculation of these values, keeping the order of the radii unchanged and accepting the spinning effect in rare situations.

The default coloring of the ellipsoid corresponds to the user-defined color used when the molecules are rendered individually to obtain visual coherence. There are two alternative coloring schemes available: the first one uses a palette of clearly distinguishable colors from which a color is selected using the ID of the cluster modulo the number of entries in this color table. This color coding allows to easily check the tracking of clusters over time when the monomers are filtered out. The second alternative coloring scheme emphasizes the evolution of the molecular cluster. Three user-defined colors represent the major evolution tendencies of the clusters. The default colors are light yellow for clusters keeping their size, light green for growing and light red

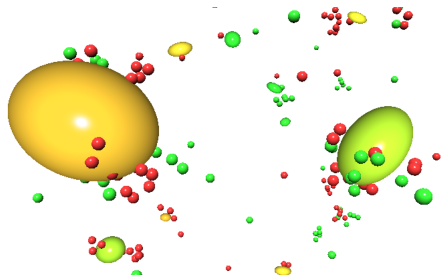

Fig. 3. Molecular cluster ellipsoids with color coded evolution and filtered monomers, color-coded whether they are joining or leaving clusters. Clusters with colors between yellow and red are decaying (left cluster), while clusters with colors between yellow and green are growing (right cluster).

for decaying clusters. These colors are interpolated according to the real evolution (e.g. slow-growing clusters are colored in light greenish yellow; see Figure 3).

3.4 Visual Representation of Monomers

The default visual representation of monomers is the point-based rendering of the molecules as either spheres or dipole glyphs. However, showing all monomers is not practicable because it will generate heavily cluttered images, as already mentioned in Section 3.2. The first approach is a filtering of the monomers according to their importance for the evolution of a molecular cluster. As basis for this filtering we introduced the cluster time distance described in Section 3.1. Only monomers are rendered whose absolute value of the cluster time distance is smaller than a user defined threshold: $ctd(m_i,t) < \varepsilon_{filter}$. To further emphasize the contribution of the monomers to the evolution of molecular clusters, the molecules are color-coded according to the sign of their cluster time distance, showing molecules joining a cluster in green, and molecules leaving a cluster in red, or in other user-defined colors (See Figure 3). One interesting situation is given by monomers starting as red, leaving molecules and then switching their color to green. If this occurs with rather small filtering thresholds, it can indicate cluster detection instabilities.

It can be easily understood that it is hard to perceive these situations, because the changes of the monomers' colors happen rather quickly. To overcome this problem we used the information of the filtered monomers to produce a pathline visualization, by connecting the positions of these molecules over time. The pathline information is generated at load-time, so no additional information in the input file format is used. However, the creation of this information increases the memory needed and the workload on the CPU. But using these pathlines allows to track the movement of the monomers over a period of time, which clarifies the joining, leaving, and rejoining phenomenon described above. Because the pathlines represent information of several configurations, the images tend to be more cluttered compared to rendering the filtered monomers.

Therefore, we decided to create a sparse visualization of the monomer movement between different molecular clusters, the *flow groups* $\mathbf{M}_f(e,v,t_e,t_v)$ described in Section 3.1. These are visualized as arrows, moving from the averaged position of all molecules in the starting configuration to the averaged position in the ending configuration. Hence, all information about the trajectory of the individual molecules is disregarded in this visualization.

The size of each arrow is calculated from the number of contained molecules. For the mapping from the number of molecules to the size of the arrow we used the cubic root with the factor provided by the Kepler conjecture ($\frac{\pi}{\sqrt{18}} \simeq 0.74048$), which corresponds to the closest packing of similar-sized spheres:

$$r(\mathbf{M}_f) = r_m \sqrt[3]{\frac{\sqrt{18}}{\pi}|\mathbf{M}_f|} \tag{5}$$

with r_m being the radius of the molecules and $r(\mathbf{M}_f)$ being the size of a flow group-enclosing sphere $\mathbf{M}_f$. The arrowhead radius, head and tail lengths are then set to the radius of this sphere to obtain a visually similar impression of volume, while additionally emphasizing its flow direction. This approach creates glyphs with a good representation of the amount of molecules in each flow group and allows for a qualitative comparison of flow groups and molecular clusters. To create a further coupling between the size of the flow group and the size of the molecular clusters involved, we use a special color coding of the flow group's arrow. The color is interpolated between two base colors depending on the ratio of the flow group molecule count and the molecular cluster molecule count. A completely blue flow group holds all molecules which formerly formed the molecular cluster, while an almost red flow group only holds very few molecules compared to the number of molecules in the cluster. So the effect that molecular clusters are present but missed by the cluster detection algorithm appears as a cluster vanishing, a rather blue flow group emerging, and a new cluster emerging where the still blue flow group vanishes (See Figure 4). Due to the clear coloring and the sparse visualization, these effects can easily be observed.

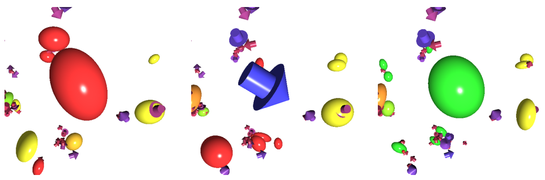

Fig. 4. Evaporating molecular cluster (red cluster ellipsoid in the middle of the left image) leaving a flow group (blue sphere, centered image) containing almost all molecules of the evaporated cluster, moving slowly and forming a new cluster (green ellipsoid in the right image). The smaller clusters in the left part of the images is surrounded by flow groups, but these are red and the cluster is rather stable (see Section 5).

4 Schematic View of Cluster Evolution

The molecule visualization allows to qualitatively judge the dataset as a whole as well as the cluster detection. This direct visual representation of the molecular datasets also allows to identify positions in space and time in the dataset where the clustering detection potentially fails to create proper results. However, it is hard to get quantitative proof in such situations, without switching to another tool to analyze the dataset. But using different tools complicates the workflow since neither synchronized selection nor coordinated views are available. So the potential error situation found in one tool must be found again in the other one, using low-level information like the IDs of the molecules or clusters. Our system allows to quantitatively analyze the molecular clusters and their evolution without changing the tool at all. For this, we created a schematic view of the evolution of the individual clusters in addition to the molecule visualization presented in Section 3.

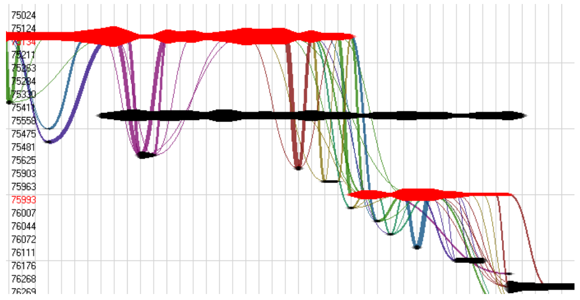

Fig. 5. The schematic view of the molecular cluster evolution with two selected clusters (red) and all corresponding flow groups connecting to other clusters (black) shows the 10,000 methane nucleation dataset with geometrical cluster detection. The flow groups' colors are based on the IDs e and v.

The two object classes represented in this view are the molecule clusters and the flow groups. The horizontal axis of the view represents the time line $\mathbf{T}$ of the dataset. Molecule clusters $\mathbf{S}_j(t)$ are represented as horizontal lines. These cluster lines are vertically ordered by their IDs j, but can also be rearranged by the user to generate clearer images. The size $|\mathbf{S}_j(t)|$ of a molecular cluster can be encoded as the thickness of the line. Because this is a time-varying value, the thickness changes when moving along the line horizontally, so that the line appears as a symmetric stripe.

The flow groups are also represented as lines, encoding their size $|\mathbf{M}_f(e,v,t_e,t_v)|$ as thickness, making them directly comparable to the lines representing molecular clusters. They are rendered as Bézier curves connecting the molecular cluster lines e and v at the times t_e and t_v. So the amount of molecules leaving the cluster at a time, represented by the decrease in the line width, can be compared with the amount of molecules leaving as a flow group. To ease the tracking of the flow groups, we support different color mappings from solid color with simple alpha blending to coloring based on the cluster IDs e and v to visually group the different flow groups between pairs of clusters.

Of course, the schematic view can be interactively zoomed and panned to maximize the effectiveness of the exploration of the dataset. Molecular cluster lines can be interactively selected by either clicking on the line in the schematic view or by picking the molecular cluster ellipsoid in the molecule rendering. Selected clusters are rendered with a different color in the schematic view and rendered with a halo for better perceptibility in the molecule view. The schematic view can then be filtered to only show the selected cluster lines, the connected flow groups, and the cluster lines directly connected to the shown flow groups.

5 Results

We present results for the following four datasets: two datasets of methane nucleation simulations, one with 10,000 spherical molecules and a trajectory of 5000 configurations, and one with 50,000 spherical molecules and 10,000 configurations. The last dataset represents a nucleation simulation of Difluoroethane (R-152a) with 100,000 two-center molecules and 1000 configurations. These three datasets are used to demonstrate our different visualization modes, including the molecular cluster ellipsoids and the flow groups. We also worked with a biochemical simulation of TEM-1 β-lactamase in water solvent with 28,000 atoms (spheres) and 2000 configurations, to compare our system with other applications.

Table 1. Performance and memory footprints for different datasets. *Fps* is an average value calculated over the complete trajectory. *Reloading* specifies the number of configurations consistently loadable per second.

dataset	memory footprint	fps	reloading
10,000 methane	1.561 MB	350	43
50,000 methane	7.801 MB	127	29
R-152a	15.600 MB	53	7
Tem1	1.114 MB	382	64

In comparison to widely used tools like VMD or Chimera, we found that their implementations are quite inefficient especially for time-dependent datasets. The TEM-1 dataset in the AMBER format can be loaded directly into Chimera at a rate of about 2 configurations per second and into VMD at about 5 configurations per second. Both tools offer the benefit that no preprocessing needs to be performed, but this also means that the user must wait 15 (or seven in the case of VMD) minutes before starting to work with the dataset as a whole. Our approach requires a preprocessing step, but accelerates the loading to more than 45 configurations per second. The performance of Chimera furthermore degrades so much while loading that it becomes unusable. This does not happen with our tool as it is designed to continually stream data thus not forcing the user to wait until the loading process is complete.

Table 1 shows performance values for the direct rendering of all molecules as individual glyphs. For other visualization modes, the values for fps and reloading vary. However, since these modes generate less graphical primitives (molecules of clusters are omitted and only one ellipsoid is drawn per cluster) the frame rates increase. The reloading values vary for a single dataset on different visualization modes because the amount of calculations performed at loading time differs. However, these variations are negligible, except for the pathlines, where the reloading rates can decrease down to 90% in most cases and 25% in the worst case (R-152a). Since our system uses a data streaming mechanism, the sizes of the data files on disk or in memory are not significant, as described in Section 3.1. Therefore table 1 only presents the average size in memory of one configuration for each dataset. The complete file size of the datasets is shown in table 2.

We ran our performance tests on a Intel Core2 Duo 6600 processor with 2.40 GHz, 2 GB memory, and an NVidia GeForce 8800 GTX graphics card with 768 MB graphics memory. The viewport size was 512^2 for all tests. All files were stored on the local SATA hard drive (no RAID hardware was used).

Table 2. Preprocessing times for all nucleation datasets. The total time is given in hours and minutes, while the time per configuration is given in minutes and seconds.

dataset	file size	total time	time per configuration
10,000 methane	2.6 GB	0:37:00	0:00.4
50,000 methane	24.9 GB	21:23:00	0:07.7
R-152a	8.7 GB	29:42:00	1:46.9

Table 2 shows the preprocessing times needed to prepare the datasets. The table does not show the times for the TEM-1 dataset. Since this is no nucleation simulation no derived data must be calculated and the preprocessor is only needed to convert the file format, which is almost as fast as simply copying the file over network. When preprocessing the three nucleation data files, however, the derived data has to be calculated which is a time consuming process. Since the datasets have trajectories of different length, Table 2 also shows the average calculation time per configuration, for better comparability. While the times of the both methane nucleation datasets are quite acceptable, the R-152a dataset needs a conspicuously long time to be preprocessed. This is due to the special structure of the data, where after a short period of time almost all molecules are clustered in many rather small clusters, making the cluster tracking, for example, rather expensive. These preprocessor runs were performed on a single machine with an AMD Opteron 248 processor running at 2.2Ghz with 4GB RAM. Compared to the simulations calculated on cluster computers with multiple processors and still needing several days, up to multiple weeks, the preprocessing time is acceptable. Planned improvements will be outlined in Section 6.

Playing back the trajectory as animation reveals flow groups temporarily taking the place of small clusters, as shown in Figure 4, which indicates a problem with the cluster detection. On the other hand, bigger clusters are always surrounded by some flow groups because many molecules first hit a cluster, but cannot join it immediately because of too different speeds and energy levels. They rebound, get slowed down and then join the cluster some configurations later. However, it is currently not clear if this is an error in the classification or if this is a valid effect needed to obtain meaningful results. To clarify the observed situation, the molecule rendering alone is insufficient. The schematic view of the clusters' evolutions provides additional insight.

Figure 6 shows two schematic views of the 50,000 methane nucleation dataset comparing a cluster detected with two different algorithms. The algorithm employed in the left view uses only the simple geometrical distances between molecules to define adjacencies. A molecule with a pre-defined number (normally four) of such neighbors seeds a cluster. The algorithm applied in the right view defines two molecules as clustered if the sum of their potential energy and their relative kinetic energy is negative[12]. This energy-based approach yields far better results when extrapolated to experimental values. The geometrical clustering not only detects too many and too large clusters, but also often creates multiple close-by clusters instead of a single one. The small red cluster S_{3007} at the top of the left schematic view is such an example. The ends of this cluster are connected to the lower and

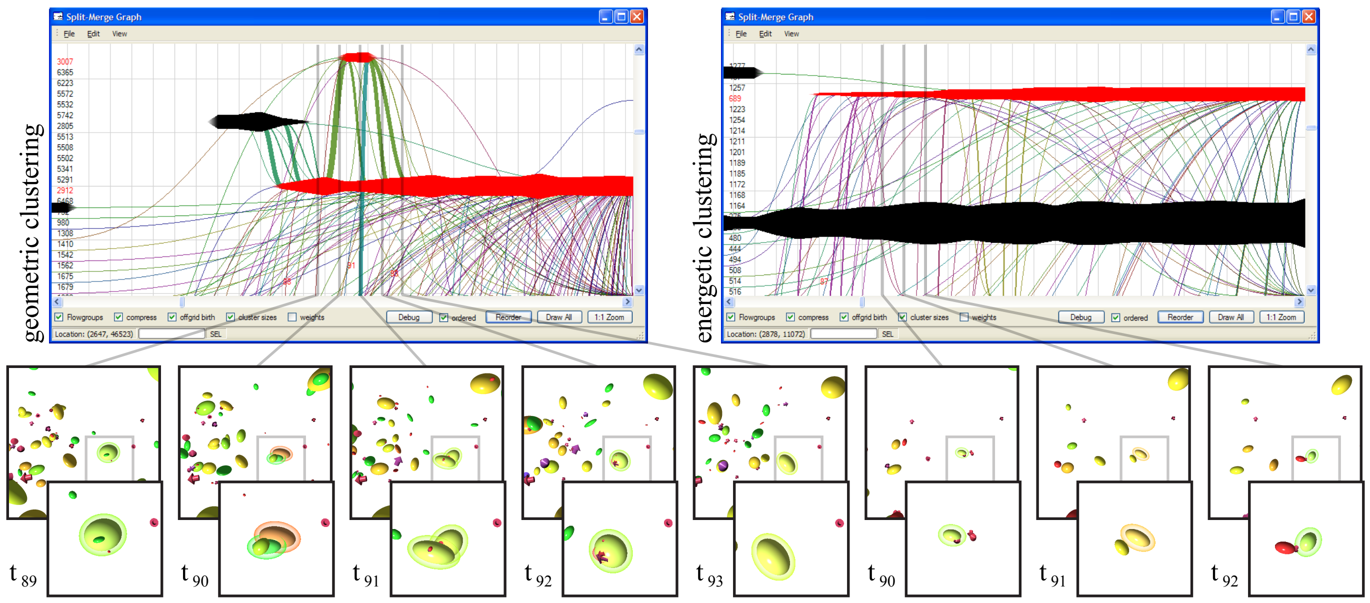

Fig. 6. Schematic views of two clustering algorithms applied to the 50,000 methane nucleation dataset. The top left image shows the results of a pure geometric clustering, and the top right image shows the results of a clustering based on energy levels. The lower images show zoomed in views of the selected cluster at different configurations. The geometrical clustering not only detects too large and too many clusters, it also splits one cluster into two for three configurations. The flow groups' colors are based on the cluster IDs e and v, such that flow groups between pairs of clusters are colored identically for easier visual tracking.

bigger red cluster S_{2912} with thick flow groups indicating that almost all molecules of S_{3007} came from and rejoin S_{2912}. The clustering with the energy-based algorithm shown in the right image does not exhibit such splintering clusters. Here, the cluster corresponding to S_{2912} is cluster S_{689}, which is rather constant in its size compared to the clear dent in S_{2912} in configuration t_{91}. There are only very small flow groups of only one or two molecules, which is characteristic for normal fluctuation between adjacent clusters.

Another interesting situation is given on the left side of the left schematic view in Figure 6, where the black cluster S_{2805} feeds cluster S_{2912} before it vanishes. Without the molecule view, the reason for this issue can hardly be identified. With the coupled view, we were able to see that cluster S_{2805} got rather slim and long around configuration t_{88} so that the geometrical approach failed to get the required four neighbors for one molecule to detect the cluster. Similar situations can be observed in the smaller 10,000 methane nucleation dataset. A small part of the graph of the geometric cluster detection for this dataset is shown in Figure 5.

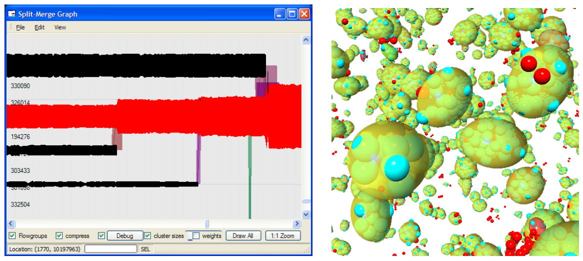
Split-Merge Graph
File Edit View
330090
326014
194276
303433
332504
Flowgroups compress Debug cluster sizes weights Draw All 1:1 Zoom
Location: (1770, 10197963)

Fig. 7. Schematic view (left) and molecule view (right) with transparent cluster ellipsoids and monomers (red; clustered molecules cyan) of the R-152a nucleation simulation. Almost all molecules are very quickly clustered in many rather small and stable clusters (appearing yellow in this color-scheme). Greater changes only happen when two clusters merge, which is clearly indicated by the corresponding thick flow groups in the schematic view.

All these findings are dataset-dependent, as Figure 7 shows. The nucleation of the refrigerant Difluoroethane (R-152a) results in most molecules clustered in many rather small droplets. Almost no fluctuation between the clusters takes place, except for rebounding molecules forming small flow groups as described above and for the merging of clusters, clearly indicated by large flow groups in the schematic view. Any errors in the cluster detection could therefore be spotted rather quickly. However, for this dataset the energetic clustering produces very good results.

The findings derived with our tool have already been applied to current research. The inadequacies in the geometric clustering algorithm used by our thermodynamics cooperation partners have been discovered with our tool. An improved energy-based algorithm based on the knowledge of the failure situations of the existing clustering algorithm has then been developed. However, in some situations also the pure energetical clustering delivers incorrect results. Thus, a combination of these approaches was created. The results of this hybrid clustering algorithm still vary depending on the simulation data. We are still investigating which of the energetical or hybrid approaches leads to the best results.

6 Conclusion

The system we presented in this paper allows thermodynamics researchers to analyze molecular cluster detection algorithms and to interactively explore time-dependent molecule nucleation datasets in an efficient and effective manner. The molecular visualization allows to identify points in space and time where molecular cluster detection may have failed. A directly coupled schematic view of the molecular cluster evolutions and the interaction between molecular clusters, represented by the introduced flow groups, allows to quickly verify the findings and to enhance the analysis of the observed situations, extracting valuable information on how to improve the cluster detection algorithms. The results section showed that our system is able to help users identify failures and weak spots in the cluster detection and that the workflow we presented is applicable to the given problem.

We believe that our approach can also be applied to other research fields using any kind of feature detection and tracking in scattered datasets like physics, biochemistry, and materials. Suitable cooperations with researchers from the areas of technical biochemistry and physics are in progress, and we are extending our software with additional visualization modes common in these fields as well as supporting other file formats and features.

The main drawback of our system is the need for the preprocessing step since we cannot perform the necessary calculations in real-time within the visualization software. We are going to optimize the preprocessor for parallel computation on PC clusters to minimize the time needed for preprocessing. Integration into the visualization software would allow to interactively view the dataset with an on-demand preprocessing of the requested configurations. In the case of on-line visualization we aim for a closer coupling between our preprocessing and simulation, since for calculation of our derived data we need information already available in the simulation, but which is too large to store in the resulting dataset.

Acknowledgements

This work is partially funded by the Landesstiftung Baden-Württemberg and by the DFG SFB 716. We want to thank Thomas Klein for his ellipsoid raycasting shader and the assistance provided in using it and Matthias Hopf for the pointcloud source code our work is based on. We also want to thank Martin Bernreuther for managing the simulation runs of most of the datasets used in this paper.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987.

[2] amira. http://www.amiravis.com/.

[3] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 243–250. IEEE Computer Society, 2004.

[4] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 53–64, 2002.

[5] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of SIGGRAPH*, pages 263–270. ACM Press, 1993.

[6] E. B. Dam, M. Koch, and M. Lillholm. Quaternions, interpolation and animation. Technical report, Department of Computer Science, University of Copenhagen, 1998.

[7] W. DeLano. Pymol: An open-source molecular graphics tool. *CCP4 Newsletter On Protein Crystallography*, 40, 2002.

[8] J. A. Fisk, M. M. Rudek, J. L. Katz, D. Beiersdorf, and H. Uchtmann. The homogeneous nucleation of cesium vapor. *Atmospheric Research*, 46:211–222, 1998.

[9] I. Ford. Statistical mechanics of nucleation: a review. In *Proceedings of the Institution of Mechanical Engineers, Part C, Journal of Mechanical Engineering Science*, volume 218, pages 883–899, August 2004.

[10] S. Gumhold. Splatting illuminated ellipsoids with depth correction. In *Proceedings of 8th International Fall Workshop on Vision, Modelling and Visualization*, pages 245–252, 2003.

[11] A. Halm, L. Offen, and D. Fellner. BioBrowser: A Framework for Fast Protein Visualization. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization Eurovis '05*, pages 287–294, 2005.

[12] T. L. Hill. Molecular clusters in imperfect gases. *The Journal of Chemical Physics*, 23:617–622, April 1955.

[13] D. Honeycutt and H. C. Andersen. Molecular dynamics study of melting and freezing of small Lennard-Jones clisters. *Journal of Physics and Chemistry*, 91:4950–4963, 1987.

[14] M. Hopf and T. Ertl. Hierarchical Splatting of Scattered Data. In *Proceedings of IEEE Visualization '03*. IEEE, 2003.

[15] M. Hopf, M. Luttenberger, and T. Ertl. Hierarchical Splatting of Scattered 4D Data. *IEEE Computer Graphics and Applications*, 24(4):64–72, 2004.

[16] W. Humphrey, A. Dalke, and K. Schulten. Vmd – visual molecular dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.

[17] V. Interrante and C. Grosch. Visualizing 3D flow. *IEEE Computer Graphics and Applications*, 18(4):49–53, 1998.

[18] B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):211–222, 2002.

[19] T. Klein and T. Ertl. Illustrating Magnetic Field Lines using a Discrete Particle Model. In *Workshop on Vision, Modelling, and Visualization VMV '04*, 2004.

[20] P. J. Kraulis. Molscript - a program to produce both detailed and schematic picts of protein structures. *Journal of Applied Crystallography*, 24:946–950, 1991.

[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 Trans. Vis. Comput. Graph.*, 12(5):1053–1060, 2006.

[22] E. Martz. Protein explorer: Easy yet powerful macromolecular visualization. *Trends in Biochemical Sciences*, 27:107–109, 2002. http://proteinexplorer.org.

[23] N. Max. Hierarchical molecular modelling with ellipsoids. *Journal of Molecular Graphics and Modelling*, 23(3):233–238, 2004.

[24] R. Pajarola, M. Sainz, and P. Guidotti. Confetti: Object-space point blending and splatting. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):598–608, 2004.

[25] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin. Ucsf chimera - a visualization system for exploratory research and analysis. *Journal of Computational Chemistry*, 25(13):1605–1612, 2004.

[26] G. Reina and T. Ertl. Hardware-Accelerated Glyphs for Mono- and Dipoles in Molecular Dynamics Visualization. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization Eurovis '05*, 2005.

[27] L. Ren, H. Pfister, and M. Zwicker. Object Space EWA Surface Splatting: A Hardware Accelerated Approach to High Quality Point Rendering. *Computer Graphics Forum*, 21(3), 2002.

[28] J. W. P. Schmelzer, G. S. Boltachev, and V. G. Baidakov. Is Gibbs' thermodynamic theory of heterogeneous systems really perfect? In J. W. P. Schmelzer, editor, *Nucleation Theory and Applications*, pages 418–446. Wiley-VCH, 2005.

[29] T. Schnabel, J. Vrabec, and H. Hasse. Molecular modeling of hydrogen bonding fluids: Monomethyamine, dimethylamine, and water revised. In W. E. Nagel, W. Jäger, and M. Resch, editors, *High Performance Computing in Science and Engineering '06*, pages 515–525. Springer, 2006.

[30] A. Sharma, A. Nakano, R. K. Kalia, P. Vashishta, S. Kodiyalam, P. Miller, W. Zhao, X. Liu, T. J. Campbell, and A. Haas. Immersive and interactive exploration of billion-atom systems. *Presence*, 12(1):85–95, 2003.

[31] T. C. Sprenger, M. H. Gross, A. Eggenberger, and M. Kaufmann. A Framework for Physically-Based Information Visualization. In *Proceedings of Eurographics Workshop on Visualization '97*, pages 77–86, 1997.

[32] Y. Suzuki, I. Fujishiro, L. Chen, and H. Nakamura. Case Study: Hardware-Accelerated Selective LIC Volume Rendering. In *Proceedings of IEEE Visualization*, pages 485–488, 2002.

[33] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Trans. Vis. Comput. Graph.*, 12(5):1237–1244, 2006.

[34] E. Tejada, J. Gois, L. G. Nonato, A. Castelo, and T. Ertl. Hardware-accelerated Extraction and Rendering of Point Set Surfaces. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization Eurovis '06*, pages 21–28, 2006.

[35] M. Volmer and A. Weber. Keimbildung in gesättigten Gebilden. *Zeitschrift für Physikalische Chemie A*, 119:277–301, 1926. (in German).

[36] J. Vrabec, J. Stoll, and H. Hasse. A set of molecular models for symmetric quadrupolar fluids. *Journal of Physical Chemistry B*, 105:12126–12133, 2001.

[37] D. Weiskopf and T. Ertl. A Hybrid Physical/Device-Space Approach for Spatio-Temporally Coherent Interactive Texture Advection on Curved Surfaces. In *Proceedings of Graphics Interface 2004*, pages 263–270, 2004.

[38] D. Weiskopf, T. Schafhitzel, and T. Ertl. Real-Time Advection and Volumetric Illumination for the Visualization of 3D Unsteady Flow. In *Proceedings of EG/IEEE TCVG Symposium on Visualization Eurovis '05*, pages 13–20, 2005.

[39] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378. ACM Press, 2001.