Faculty of Computer Science
University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Student Research Project No. 2372

Visualization of Polarization Domains in Ferroelectric Materials

Katrin Scharnowski

Course of Study:	Computer Science
Examiner:	Prof. Dr. Thomas Ertl
Supervisor:	Dipl.-Inf. Michael Krone
Commenced:	April 9, 2012
Completed:	October 9, 2012
CR-Classification:	I.3.7, I.3.8, J.2

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

Computational material science is an emerging research field which combines the methodology and theory of material science with computer aided simulation techniques and algorithms. High-performing computational resources can be obtained more and more easily. *Molecular Dynamics* (MD) particularly benefit from this, as well as from the development of new simulation techniques. The output of structured data also allows for the application of suitable visualization techniques. This is crucial for exploratory analysis, especially when dealing with large sets of particles or volumetric data.

Since the discovery of the first material showing ferroelectric behavior in 1921 by Valasek [Val21], ferroeletrics have been the matter of extensive studies in material science. Ferroelectric materials have two defining properties. A dielectric material is called *ferroelectric*, if it exhibits *spontaneous polarization*, and, additionally, this polarization can be reversed by applying an external electric field [RDL+07]. Figure 1.1 illustrates that the development of induced polarization in a ferroelectric material follows a hystereses loop, rather than being linear. This behavior is crucial for a lot of applications based on ferroelectrics, since it enables switching between two different states by changing the direction of the electric field [RDL+07]. The term "ferroelectric" was established by analogy with ferromagnetic materials, which where already widely known and showed the same ability to switch between two states of differently oriented polarization.

In most ferroelectrics a phase transition from a paraelectric to a ferroelectric state takes place when the material is exposed to decreasing temperature. This transition often corresponds to a distortion of the atomic displacement, relative to the high symmetry reference state in the paraelectric phase. The temperature at which the transition happens is called *Curie temperature*.

According to [Uch05], a variety of applications for ferroelectric materials can be thought of. Among others, the possibility to produce capacitors and memory devices is mentioned. High-permittivity capacitors are the major application for ferroelectric materials. The application of ferroelectric materials as memory devices has been investigated only recently. Conventional memory devices use SiO_2 films, which fail to maintain sufficient capacity with decreasing area. Due to their high dielectric constant at room temperature, several ferroelectric materials are promising alternatives for the fabrication of both volatile and non-volatile memory devices. Further applications mentioned in [Uch05] and elsewhere (see e.g. [Wad00]) include pyroelectric sensors, electro-optic devices, electrostrictive transducers or PTC thermistors.

The simulation of the movements of atoms and molecules through MD has been widely established and is applied in various areas, such as nanotechnology, biochemistry or biophysics. Here, atoms are represented by particles and their position in every time

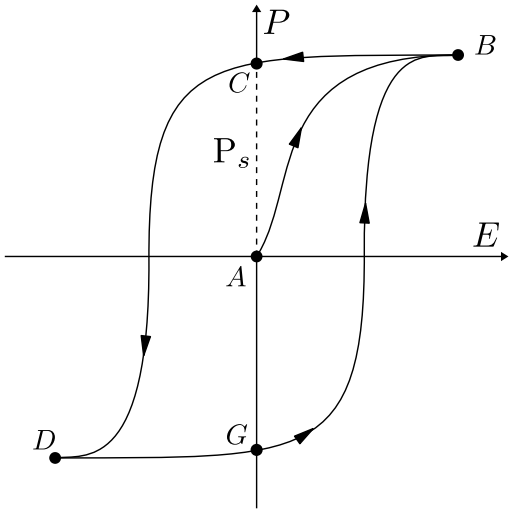


Figure 1.1: A schematic hysteresis loop of a ferroeletric (cf. [RDL+07, Wad00, WDH74]). Point A represents a generic starting point with randomly oriented domains and overall zero polarization. When applying an electric field, the dipoles start orienting according to the field until polarization of the material is saturated (point B). If the field is then reversed until it reaches zero, the material still exhibits the *spontaneous polarization* P_s (point C). The field is then completely reversed until point D is reached and the polarization of the material is saturated again. If the field is then raised again until the zero field is reached (point G) the material again possesses spontaneous polarization which, however, defers from the polarization exhibited in point C. The material, therefore, possesses two different states of spontaneous polarization. The state of the material can be switched by reversing the field back and forth.

step is computed by numerically solving Newton's equation of motion. This allows for insight into the structure and the properties of the examined material on a molecular level, which would be difficult or even impossible using experimental methods. Recent research allows for the simulation of phase transitions of ferroelectric materials when exposed to a graduate temperature decrease. This enables gaining detailed knowledge about the development of the size and structure of polarization domains.

The visualization of phase transitions, or polarization domains in general, is a rather novel research field. Recent work of Grottel et al. [GBM+12] shows how a combination of glyph based visualizations and isosurfaces derived from the original data can help understanding electrostatic properties of metal oxides.

The topic of the IEEE SciVis Contest 2012 is the detection of phase transitions in ferroelectric materials. The SciVis contest mainly targets computer scientists, who are not expected to have expert knowledge about the material scientific background. Hence, a list of several succeeding tasks was provided on the contest home page, in order to give

clues and some reference points to start with. The first task suggests to extract a vector field by calculating the temporal displacement of all titanium atoms. Regions which contain vectors of similar orientation should than be defined as domains. Furthermore the extraction of domain boundaries by the means of standard vector field operations is suggested. Here, zero lines are important, since these are the regions were the polarization changes and, therefore, vanishes. These domains, as well as other vector field singularities (critical points) should than be visualized. Additionally, a material scientific task was provided. Here, the participants were asked to use their visualization to investigate whether the phase transition occurs globally or starts locally and is than spreading through the rest of the material. The tasks can be found in Appendix A.1.

The goal of this work is to extract a vector field off the data provided for the SciVis-Contest 2012 which indicates polarization. Furthermore, to develop a visualization which helps understanding how the shape and size of polarization domains evolve over time. It should also allow for the identification of the phase transitions. Therefore, the tasks of the contest are to be addressed and used as a basis. However, in this student research project, the task description was only used as a lose orientation rather than following it exactly.

Based on the provided data a combination of a glyph based visualization and semi-transparent isosurfaces was implemented. It allows identifying some aspects of the phase transitions, as well as the movement and reshaping of domains. The results have been submitted to the IEEE visualization contest [SKBS12]. It has been awarded as winning submission to the contest and received outstanding marks by the reviewers.

The rest of this document is structured as follows:

Chapter 2 – Data Processing and Feature Extraction: This chapter starts by providing a minimal theoretical background. Then the data provided by the SciVis-Contest 2012 is described. Furthermore, the data processing and the derivation of the domains are adressed.

Chapter 3 – Visualization: In this chapter basic visualization techniques, which are combined later, are described. This includes GPU glyph raycasting, direct volume rendering and Line Integral Convolution (LIC). Additionally, possible issues when combining these methods are addressed.

Chapter 4 – Results and Discussion: In this chapter results are presented and discussed, which are obtained by applying basic visualization methods to the data extracted before. Furthermore, potential future work is pointed out.

Chapter 5 – Summary: This chapter summarizes the document.

2 Data Processing and Feature Extraction

The data provided for the IEEE visualization contest does not contain electrostatic dipoles. Therefore, a way has to be found to approximate the polarization of the material by only using the provided atom positions. In order to extract a vector field from this data, a basic understanding of the theoretical background is necessary. Another challenge is the strong thermal vibration of the individual atoms. Finally, domains have to be defined and a way has to be found to extract them from the vector field.

2.1 Theoretical Background

In [CR08] a *dielectric* material is defined as a material which exhibits a dipole structure. This means that there are oppositely charged entities present on a molecular or atomic level. This separation of positive and negative electric charge can be represented by *field vectors* or *dipole moments* as illustrated in Figure 2.1a. The magnitude of a dipole's field vector can be calculated by the the product of distance d between the oppositely charged particles and the amount of charge q.

$$p = qd \tag{2.1}$$

By applying an electric field to a dielectric material, the dipoles can be redirected in the direction of the field. This process is called *polarization* and is illustrated in Figure 2.1.

The mechanics which enable the polarization of a material, can be further split up into three aspects, all of which are described in [WDH74]. *Electronic polarization* (P_e) only exists while an electric field is externally applied to the material. It can theoretically occur in every atom. It is the result of the relative displacement of the electrons' gravity center to the positive nucleus (cf. Figure 2.2a). *Ionic polarization* (P_i) only occurs in materials which contain ionic bindings. Here, the electric field causes a displacement of the positive and the negative ions opposite directions. This leads to a dipole moment and, therefore, polarization (cf. Figure 2.2b). *Orientation polarization* (P_o) can only take place in materials that possess *permanent dipoles*, which are present even without an external field being applied. This phenomenon is known as *spontaneous polarization*. By applying an external electric field, the permanent dipoles can be reoriented in the direction of the field, causing the orientation polarization (cf. Figure 2.2c). This behavior is temperature dependent though. Since thermal vibrations counteract the alignment of the dipoles, the polarization decreases with increasing temperature. Every dielectric material exhibits

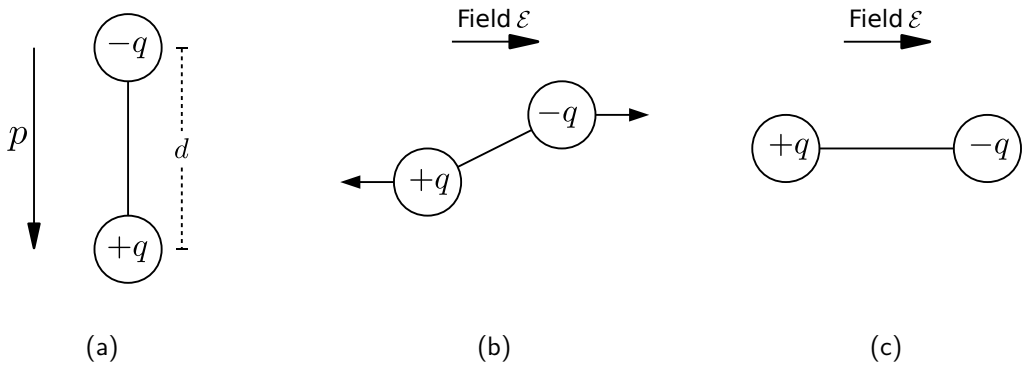


Figure 2.1: Reorientation of a dipole moment by applying an external field [CR08]. (a) schematically illustrates the field vector p representing a dipole. It starts at the negatively charged pole and points towards the positively charged pole. Its magnitude is defined by the charge q and the distance d between the two poles. In (b) an external electrical field is applied to the dipole. Consequently the dipole starts reorienting according to the field direction $\mathcal{E}$. (c) shows the resulting orientation after the field has been applied.

at least one of those components. The overall polarization of the material is, therefore, determined by the sum of all three components:

$$P = P_e + P_i + P_o \tag{2.2}$$

Ferroelectrics are a group of materials which exhibit spontaneous polarization. Additionally, this polarization can be switched between two states by applying an external field. One extensively studied group of ferroelectrics are *perovskite* oxides. Ghosez et al. [Gho02] describe their structure as a characteristic composition ABO_3. In this compound A and B represent generic particles of positive charge. These particles are opposing the negatively charged O atoms. The ideal perovskite structure is a centrosymmetric cubic structure with the A atoms being the corners and one O atom at the center of each face.

Barium titanate ($BaTiO_3$) is a ferroelectric oxide which has a perovskite structure at high temperature. This state is known as the cubic phase. If exposed to a temperature decay, it undergoes three phase transitions, each of which leads to a new ferroelectric state. A detailed description of the phase transitions, as well as the ferroelectric phases can be found in [KLBC93]. In all known ferroelectric crystals, the permanent dipole is the result of the positioning of the ions in the unit cells. Here, the displacement of the atoms causes a distortion of the symmetry present in the paraelectric phase [RDL+07].

The phase transitions in $BaTiO_3$ are temperature dependent. Above the Curie temperature, 393 K, $BaTiO_3$ possesses a typical cubic perovskite structure. Since the cells are centrosymmetric, there is no spontaneous polarization and the material is paraelectric. If the temperature is decreased, a change from the paraelectric cubic phase to the ferroelectric tetragonal phase occurs. The Ti atom then moves away from the cell center parallel

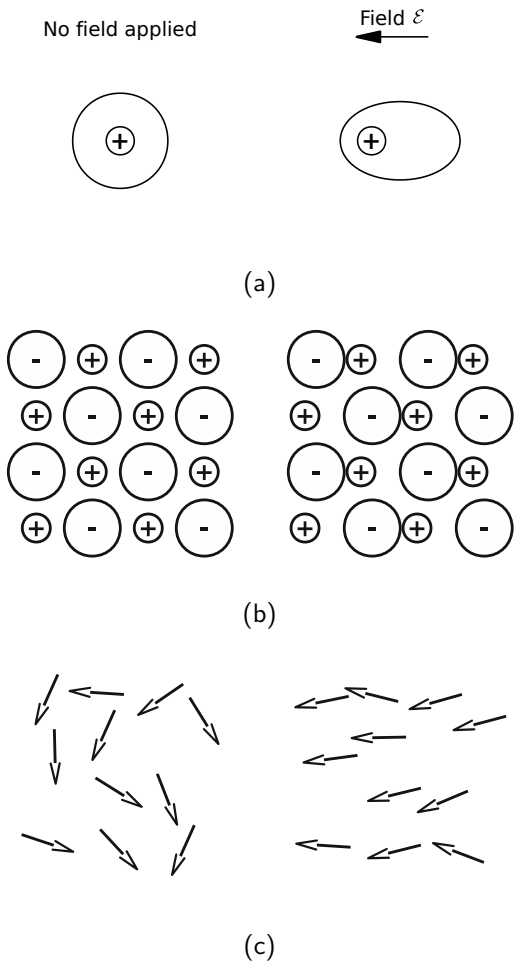


Figure 2.2: The three phenomenons resulting in polarization (taken from [WDH74]). (a) illustrates electronic polarization by the displacement of the electron cloud caused by the electric field. Ionic polarization can be seen in (b). It is caused by a rearrangement of ions. The third phenomenon is illustrated in (c). It shows orientation polarization, which only occurs in materials with permanent dipoles. By applying the electric field these dipoles get reoriented according to the field direction.

to the [100] axis. At the same time, the cell is elongated in this direction. At 278 K, the Ti atom movement, as well as the cell elongation, follow the [011] axis and the material transforms into another ferroelectric state with orthorombic structure. Finally, at 183 K, the last transition takes place and the resulting symmetry of the Ti movement and the cell elongation is rhombohedral.

There are different approaches to identify the different phases in $BaTiO_3$. According to Rabe et al. [RDL+07], the distortion is dominated by the movement of the Ti atoms relative to the oxygen atoms. In [KLBC93], on the other hand, the overall cell elongation along different axes is taken into account. In Figure 2.3, the development of the cell shape is illustrated for all phases.

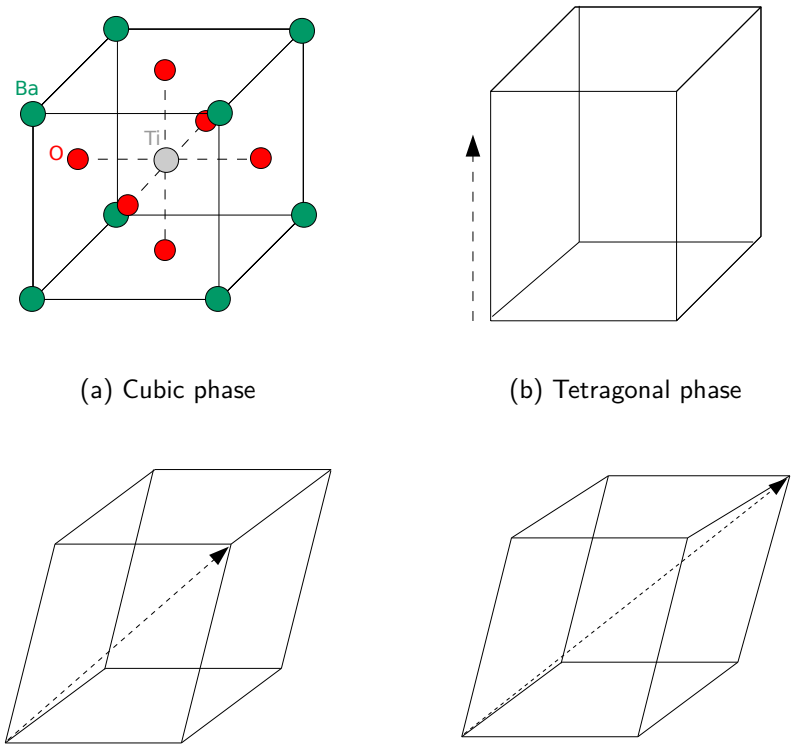
Even when there is no electric field present, the dipoles of a ferroelectric interact with another. According to [WDH74], adjacent dipoles show a tendency to align in the same direction. This gives rise to the formation of ferroelectric polarization domains, i.e. clusters of dipoles exhibiting similar orientation. The application of an electric field particularly leads to a growth of domains which are aligned in the direction of the field. Since the permanent dipoles originate from the distortion of the cell symmetry in the tetragonal phase, the rise of large domains could be used as indication for the transition from the cubic to the tetragonal phase.

2.2 Data Basis

In order to fulfill the visualization tasks of the SciVis Contest 2012 (see Appendix A.1), several data sets were provided on the contest homepage. The data set contains the results of MD simulations, which use potential models based on ab-initio calculations. It consists of a block of 50×50×50 unit cells of $BaTiO_3$ and a total of 625000 atoms. For every atom, the atom type and the Cartesian coordinates are provided. Additionally, the instantaneous velocity of all atoms was given. The atom positions are in Ångström (Å), velocities are in Å/fs. In general, the information contained in the data sets is time dependent. The simulation was executed over a time span of 10.0 ns in total and a history snapshot was taken every 20 ps resulting in 500 consecutive frames. The simulation contains the transition from the paraelectric cubic phase to the ferroelectric tetragonal phase, which is achieved by decreasing the temperature gradually.

In addition to the files containing the whole the time series, two separate time steps, one at the beginning of the series, and one relatively at the end of the series, were provided. Apparently, these two files should be used to achieve a before/after comparison for the material with regard to the phase transition. The data was provided using two different ASCII file formats. The VTK file format used by the visualization software *Paraview*[1] and the DCD file format, which can be read e.g. by *VMD*[2]. The complete time series was given in both formats, however the two individual time steps were only provided in the

[1] http://www.paraview.org/
[2] http://www.ks.uiuc.edu/Research/vmd/



(a) Cubic phase

(b) Tetragonal phase

(c) Orthorombic phase

(d) Rhombohedral phase

Figure 2.3: The cell deformation in the four phases of barium titanate (c.f. [KLBC93]). The unit cell has a perovskite structure with eight Ba atoms being the corners. In the (paraelectric) cubic phase the unit cell is centrosymmetric. In the phases illustrated in (b), (c) and (d) the material is ferroelectric. In the tetragonal phase the cell is elongated alongside the [001] axis. In the orthorombic phase the cell is elongated alongside the [011] axis laying inside one of the cube sides. In the rhombohedral phase the cell is deformed according to the [111] axis.

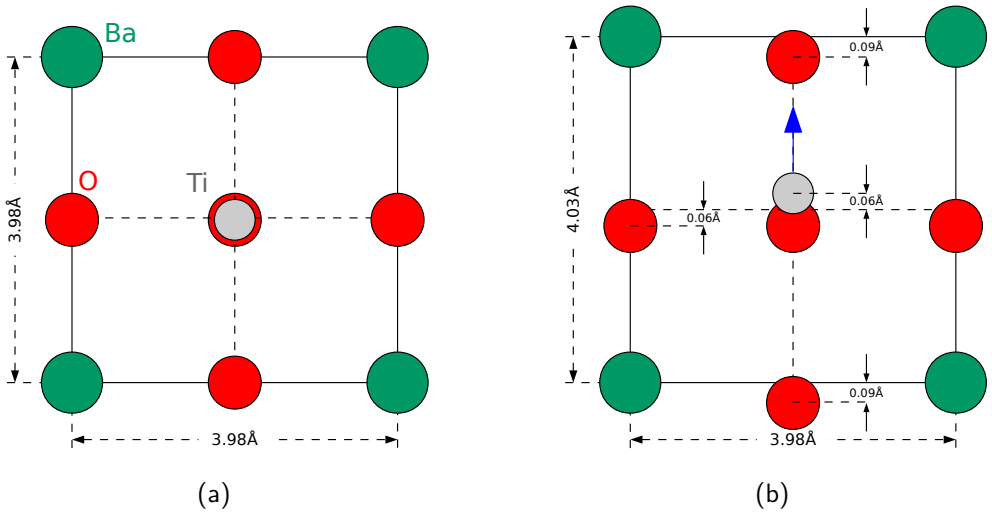


Figure 2.4: Cubic and tetragonal phase of barium titanate. In the tetragonal phase, the titanium atom starts shifting away from its original position, which creates a permanent dipole moment. The orientation of the resulting dipole moment is shown in blue.

DCD format. Thus, for simplicity, in this student research project, the data sets given in the DCD format were used.

In this file format, every time step contains of two commentary lines followed by four columns of data for every atom, respectively. The first commentary line contains the number of atoms, while the second commentary line contains additional information regarding the simulation, like e.g. the total energy. The first column contains the atom type and the other three columns contain x-, y- and z-values of the atom position. An example of the file format is given in Listing 2.1.

Since the framework used to implement the visualization, MegaMol [GRE12], uses streaming for large data sets, the atom positions were written to a simple binary format in order to reduce I/O operations to a minimum. All time independent information was computed beforehand and written to separate binary files. This includes the atom types as well as the unit cells and the connectivity information obtained from their definition. The atom types were decoded in char values (one char per atom), before being written into a separate file.

The connectivity of the crystal structure defined by the unit cells can be obtained in advance as well, since it does not change over time. Here, a unit cell was defined containing four Ba atoms in the corners, one Ti atom in the center and one O atom in the middle of every face, as illustrated in Figure 2.3a. In order to find all atoms belonging to one cell, a simple neighborhood search was executed for all Ti atoms. Since the atom types of all atoms are known, the 8 nearest Ba atoms and the 6 nearest O atoms could

```
 625000
i =     4500, time =      900.000, E =   -497481.4576749648
 O          2.0412313295         1.8817370173        -0.0745821260
 O          2.0357246595         1.8648101554         3.9490123179
 O          1.9401344984         2.0561252788         7.8955649008
 O          2.0938016553         2.0005280326        11.8685820573
 O          1.8947569302         2.0271176774        15.9288616358
 O          2.0511723827         2.0951983376        19.9188875456
 O          1.9182841173         1.9796873200        23.8144226579
 O          1.9659986184         2.0706196003        27.8294866854
 O          2.0429181287         1.9647054977        31.9209885780
 O          1.9892337415         1.9560424318        35.8655190735
 O          2.0700183761         2.1064522698        39.8478397430
 ...        ...                  ...                 ...
 ...        ...                  ...                 ...
```

Listing 2.1: Example of the ASCII DCD file format used in the SciVisContest2012. The first line contains the number of atoms. It is followed by a commentary line containing information concerning the simulation. Then the atom type and the coordinates are given in four columns for every atom, respectively.

easily be found for every Ti atom. Since the atoms roughly remain in their initial spatial order, these unit cells are feasible over the whole time series.

Additionally the cells were sorted according to the position of the Ti atom associated with every cell. Sorting the cells is necessary for operations which require the neighborhood of every cell to be known. The sorting is considerably simplified by the fact, that the whole data set is roughly axis aligned. First, all Ti atoms are reordered by the x-coordinate of their position. Subsequently, the data set can be divided into slabs aligned on the x-axis by taking 2,500 consecutive cells into account. Now, all Ti atoms contained in one slab are sorted by their position's y-value. This step results in scan lines. Finally, all Ti atoms in one scan line can be sorted by their position's z-value. The unit cells were then enumerated and written to a binary file as well. Here, one line per cell was written, containing the index of the cell as well as the indices of all atoms contributing to that cell. First, the indices of the eight barium atoms are given, followed by the six oxygen atoms. The last index is the titanium atom. Consequently, both the atom types and the unit cells can then be loaded independently of the usual streaming process.

As shown in Figure 2.5, the overall movement of the atoms is superposed by a strong thermal vibration. This is expected behavior in an environment like this, however, it is much more difficult to extract the local displacement under this circumstances. In order to filter the thermal vibration, the atom positions were averaged using a sliding time window, thereby producing a sequence of time-averaged positions for each atom. This should reduce the noisy oscillation of the atoms which might superpose the more subtle atom displacements causing the polarization. In order to find an appropriate time window several ranges were tested. An averaging of less than ten frames (200 ps) provided too little smoothing while an averaging over more than 35 frames (700 ps) caused too strong smoothing. The average atom displacement in each frame when using different

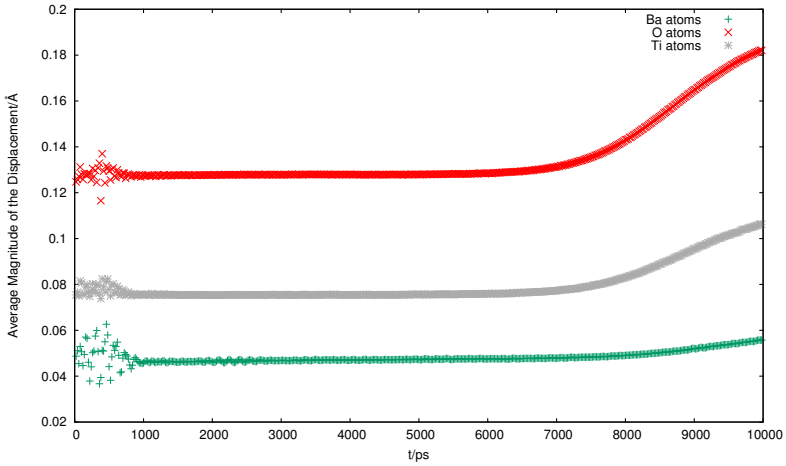


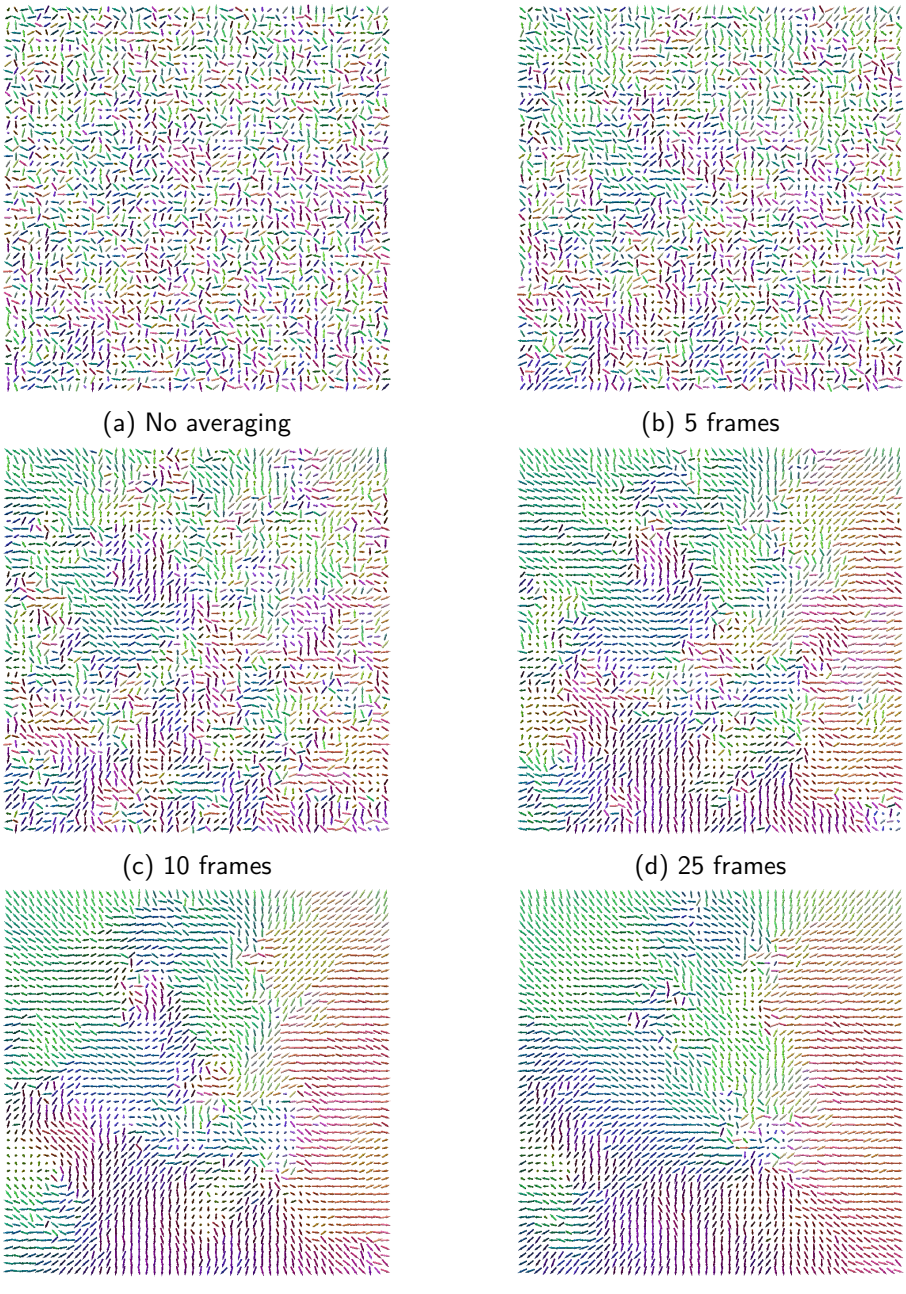
Figure 2.5: Magnitude of the atom displacement. The displacement has been computed over 1 frame (20 ps). The plot illustrates the average value of the displacement magnitude of all atoms per frame. The average has been computed separately for the three different atom types. The oxygen atoms are obviously the most mobile atom type. Their movement per frame even exceeds the expected length of the dipole moment in the fully developed tetragonal phase. To avoid that this vibration superposes the dipole moment, the atom positions have to be averaged over a certain time window.

time windows is illustrated in Figures 2.7 and 2.6. Consequently, a time window of 25 frames is used, which results in a time series of 475 frames.

2.3 Extracting the Vector Field

Ferroelectric polarization domains are clusters of dipoles which exhibit a similar orientation. The overall goal of the visualization is to illustrate the development (e.g. the size, the orientation of the contained dipoles) and movement of these polarization domains over time. This should allow for the identification of the phase transition from the cubic to the tetragonal phase, since the domains origin from the local displacement of the titanium atom in the tetragonal phase.

Exact electrostatic dipoles are not obtainable in this case, therefore, an approximation for the polarization has to be found. The approximation basically has to meet two requirements. There have to be identifiable domains towards the end of the time series which are not present at the beginning. Additionally, the vector field has to imply polarization. Since the displacement of the titanium atom is a strong indicator for the polarization, calculating the relative displacement of the titanium atom to the cell center is used as a starting point.



(a) No averaging (b) 5 frames

(c) 10 frames (d) 25 frames

(e) 35 frames (f) 50 frames

Figure 2.6: Averaging atom positions using different time windows. Here, the effect on the temporal displacement field of titanium atoms over 25 frames is illustrated. The vectors have been normalized for clarity. Using a time window larger than 25 frames only leads minor changes in the distribution of the vectors.

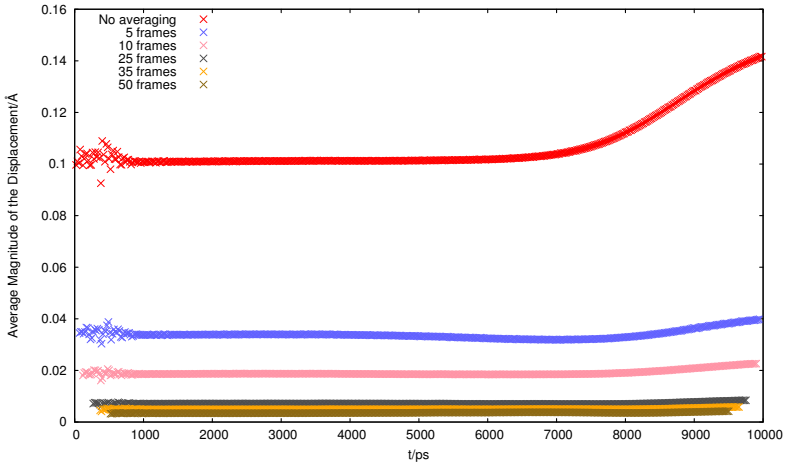


Figure 2.7: Magnitude of the atom displacement (over 1 frame) using different time ranges for averaging the position. The plot illustrates, that averaging over more than 25 frames does not lead to significantly weaker oscillation.

If idealized cells are assumed, the orientation of the polarization can be calculated by computing the displacement of the positive and the negative centroid of each cell. The position of the positive centroid is the average of the positions of the barium atoms and the titanium atom. The position of the negative centroid can be obtained by averaging over the positions of all oxygen atoms. Unfortunately, the vector field obtained by this method does not exhibit domains consisting of more than a few vectors. Particularly, there is no growth of these domains towards the end of the series.

It is possible that some of the atoms distort the results, since idealized cells as described above are usually not to be expected and, therefore, exact dipole orientations cannot be obtained. Concentrating on selected atom types might give an indication of the orientation of the polarization. In order to test several approaches, three vector fields were extracted. One vector field contains the displacement of the titanium atom and the centroid of all oxygen atoms. The second vector field contains the displacement of the titanium atom from the cell center which, in this case, is defined by the eight barium atoms in the cell corners. The third vector field solely uses the temporal displacement of the titanium atoms over a fixed time range. Please note, that, in an idealized cell structure, the orientation of all of these vector fields would be a feasible approximation.

The first two vector fields mentioned exhibit behavior similar to the vector field based on all atoms in the unitcell. The domains are relatively small (five to six vectors) and do not change significantly in size or shape. The third vector field, based solely on the temporal displacement of the titanium atoms, shows a noticeable development of domains through the whole time series. Since this vector field contains absolute displacements, it is naturally superposed by any deformation the material exhibits. This includes a noticeable shrinking of the whole structure as the temperature decreases. However, the displacements

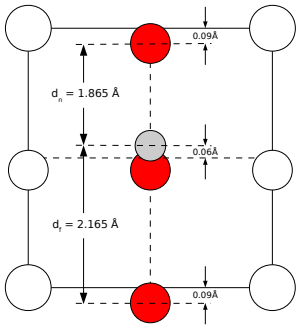


Figure 2.8: Relation of atom displacements as measurement for cell symmetry. The relation of the two distances d_n and d_f can be seen as a measurement of the cells symmetry. If their difference is significantly high in one direction in comparison with the other directions, the cell has probably developed a permanent dipole in that direction.

of the titanium atom are the strongest indicator for the transition from the paraelectric to ferroelectric phase. This vector field definition is also mentioned explicitly in the task description of the IEEE Visualization Contest. Thus, the temporal displacement field is used subsequently.

Another approach to identify the phase transitions is to concentrate on the cell elongation in the direction of one axis. This elongation goes along with the transition from the cubic to the tetragonal phase. One possibility to identify elongated cells is to compare the distances of the neighboring barium atoms of one unit cell. However, doing so does not deliver useful results, in which the distances along one axis are noticeably longer than the others. This might be a result of the averaging, since the elongation of the cuboid defined by the eight barium atoms is rather small (in theory, 0.05 Åin the fully developed tetragonal phase).

An alternative method uses the relation of the distances from the titanium atom to two opposing oxygen atoms, respectively (see Figure 2.8 for an illustration). This relation can be seen as a measurement for the symmetry of the cell. It is computed for all three axes of the unit cell and the maximum is taken into account. This maximum should allow for the identification of the elongation of the cell, since it is much more prominent that the very subtle elongation of cuboid defined by the barium atoms. However, using this approach it was not possible to clearly identify the elongation of cells.

2.4 Extracting Domains

In order to extract domains which deliver indications about the material's polarization, clusters of vectors with similar orientation have to be found.

To allow for better handling of the data, a uniform grid containing the vector field was extracted. There are several methods to create the uniform grid which mainly differ in the used interpolation method. In this case, a Gaussian splatting algorithm according to [KSES12] was used. Here, a Gaussian kernel is used for each particle to represent the density contribution at its position in space. The density of all neighboring particles is then accumulated for each voxel together with a vector quantity. In this case, the vector quantity was the displacement of the titanium atoms. In order to achieve appropriate smoothing, the radius of the Gaussian kernel was globally set to 2.5 Å. That way, the Gaussian kernel located in a unit cell slightly overlaps with the Gaussian kernels in adjacent cells.

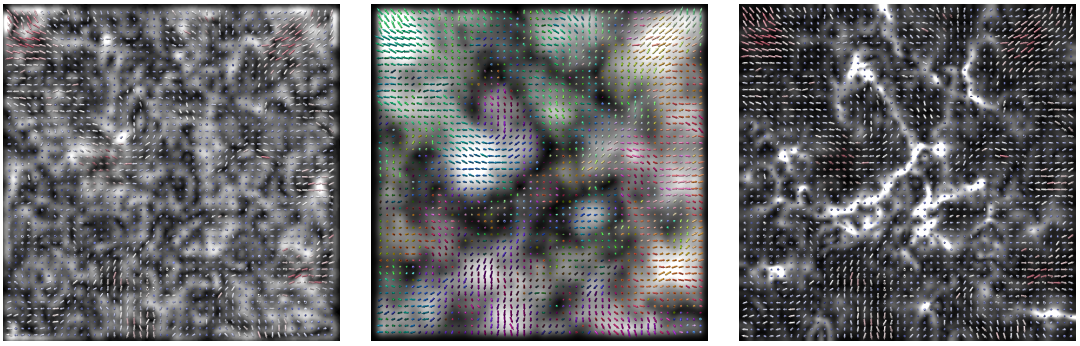
In the extracted vector field, areas containing similarly oriented vectors also exhibit a high vector magnitude. Since domains contain vectors of similar or equal orientation, the vector magnitude could, therefore, be utilized to define the domains. However, in a more general case, this coherence cannot be expected. Therefore, a better approach would be to take the local rotation of the vector field into account. This was done by calculating the curl magnitude of the vector field. For a vector field $F(x, y, z)$ the curl $\nabla \times F$ can be calculated as

$$\nabla \times F = \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right) e_x + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right) e_y + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right) e_z, \tag{2.3}$$

where e_x, e_y and e_z are the unit vectors for the x-, y- and the z-axis, respectively. The curl value is influenced by both the local vector orientation and the vector magnitude. Consequently, a small vector magnitude leads to a smaller curl magnitude. However, the vector orientation is the defining property for the sought domains. The vector field was, therefore, normalized before calculating the curl, to avoid that the vector magnitude affects the resulting scalar field. This leads to a high curl magnitude only in areas, where the vectorfield exhibits high local rotation, i.e. where domain boundaries are located.

The curl calculation was implemented using *C for CUDA*. CUDA (Compute Unified Device Architecture) is the computing engine in Nvidia GPUs, which can be accessed by using standard programming languages. This allows executing parts of the computation on the GPU, while taking advantage of the parallel architecture. The calculation takes place in several steps using five CUDA kernels subsequently. First, the vector field is normalized in order to suppress the influence of the vector magnitude on the resulting curl. Subsequently, three kernels are executed, which each are used to calculate one component of the curl vector using central differences to approximate the partial derivatives. Finally, the curl magnitude is computed and sent back to the CPU.

Figure 2.9 shows a summary of the scalar fields mentioned above. Strikingly, the curl magnitude based on the normalized vector field and the vector magnitude clearly correspond to each other. In areas of high vector magnitude, the curl magnitude is rather small. Subsequently, when using the term *curl magnitude*, the curl magnitude based on the normalized vector field is meant.



(a) (b) (c)

Figure 2.9: Texture slices of different scalar fields extracted from the vector field. In (a), the curl magnitude without prior normalization of the vector field is shown. (b) shows a slice textured by the vector magnitude in the data set. The magnitude roughly corresponds to areas containing vectors of similar orientation. (c) contains the magnitude of the curl after normalizing the vector field. When calculating the curl based on the normalized vector field, the curl magnitude and the vector magnitude correspond to each other. Here, the curl magnitude is especially high in areas of little vector magnitude.

2.5 Locating Critical Points by Calculating the Topological Degree

Locating critical points in the extracted displacement field can give hints about the location and size of domains. In this case, critical points are vector field singularities.

Greene[Gre92] described a way of calculating roots in three dimensional functions by recursively partitioning regions into smaller cells and discarding cells which do not contain the root. This approach can be utilized to find first order critical points in 3D vector fields. Here, the aim is finding cells which contain critical points by calculating the topological degree. The topological degree indicates whether the field vanishes inside the cell. Since critical points are points in which the vector field vanishes, the topological degree can be used to test whether a volume contains a first order critical point.

The topological degree of a cuboid cell can be calculated as follows [Gre92]. First the field is sampled at the eight corners of the cell. Each of the six sides of the cell is then divided into two triangles. The values sampled before can now be used to estimate the development of the field on the triangles. This can be done by assuming linear interpolation. In the case of a 3D vector field, this yields the solid angle A_T the vectors v_1, v_2 and v_3, sampled at the three corners of the triangle, are enclosing:

$$\begin{aligned}\tan^2(0.25A_T) = &\frac{tan(\theta_1+\theta 2+\theta 3)}{4}\times\frac{tan(\theta_1+\theta 2-\theta 3)}{4}\times\\ &\frac{tan(-\theta_1+\theta 2+\theta 3)}{4}\times\frac{tan(\theta_1-\theta 2+\theta 3)}{4},\end{aligned} \tag{2.4}$$

with θ_1 being the angle between the vectors v_2 and v_3. If the triple cross product $v_1 \cdot v_2 \times v_3$ is negative, the area calculated for that triangle is taken to be negative. In order to decide whether the volume encloses a vanishing point, the solid angles of all twelve triangles are accumulated. If their sum is greater than 4π, and, therefore, greater than the solid angle of the unit sphere, the volume contains a null point.

In this project, no bisection algorithm was implemented. Instead, the topological degree was calculated for the smallest possible cell size (depending on the resolution of the extracted uniform grid) using a straightforward CPU implementation. Additionally, the calculation was parallelized using *OpenMP (Open Multi-Processing)*, which allows executing code in several threads on the CPU.

3 Visualization

In this chapter, basic techniques are explained, which are later combined to achieve the actual visualization. This includes point based glyph ray casting, Line Integral Convolution and direct volume rendering. For each of these techniques the theoretical background is given before describing details of the actual implementation and modifications made by the author. The implementation was done using OpenGL, GLSL and CUDA and was embedded into the visualization framework *MegaMol* [GRE12].

3.1 GPU Glyph Ray Casting

Using glyph representations has been established as a standard approach to visualize large sets of particles. A way of rendering glyphs by using programmable graphics hardware is described in [KE04]. Here, one `GL_POINT` per glyph is sent to the render pipeline and the correctly projected glyph is then created in the fragment shader. The basic idea is to define an implicit surface, representing the glyph, by only a few parameters. Both the position and the parameters are sent to the shader program. Based on this information, the surface of every glyph is then shaded in the GPU. This technique was further developed by Reina et al. [RE05] by combining several basic glyph types to a dipole glyph. In this project *GPU Glyph Ray Casting* similar to the one described in [RE05] was used.

3.1.1 Algorithm

The exact type of the surface is generic, the method can, therefore, be applied to different glyph types without changing much of the procedure. All glyph types used in this student research project as well as the according parameters are summarized in Figures 3.2 and 3.3. Assuming that the center point of the glyph lays at the origin, the implicit surface of a sphere is defined by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}^2 - r^2 = 0. \tag{3.1}$$

A cylinder, oriented parallel to the x-axis, is defined by

$$\begin{pmatrix} 0 \\ y \\ z \end{pmatrix}^2 - r^2 = 0. \tag{3.2}$$

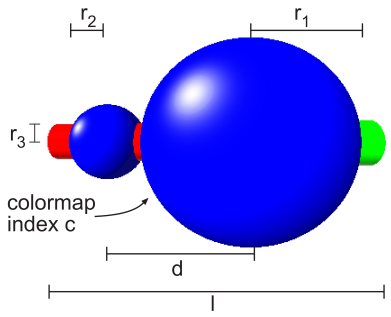


Figure 3.1: Dipole glyph and the adjustable visualization parameters (taken from [RE05]). r_1 and r_2 are the radii of the two spheres representing the particles. r_3 is the radius of the cylinder, l is the total length of the dipole and d is the distance between the two particles' midpoints. The colormap index c is used to determine the color of the glyph.

However, additional boundary conditions are needed to clamp the sides of the cylinder and define the boundary surfaces:

$$\begin{pmatrix}1\\0\\0\end{pmatrix} \cdot \vec{x} \leq \frac{l}{2}, \begin{pmatrix}-1\\0\\0\end{pmatrix} \cdot \vec{x} \geq -\frac{l}{2} \tag{3.3}$$

The third basic glyph type, a cone, oriented parallel to the x-axis, with its tip at the origin, can be specified by

$$\left(\frac{r}{h}\right) x^2 - y^2 - z^2 = 0 \tag{3.4}$$

With the boundary conditions being

$$x \leq h, x \geq 0. \tag{3.5}$$

More complex glyph types can be specified by combining the implicitly defined surfaces and setting the parameters according to the context. For instance in [RE05], a dipole glyph consisting of one cylinder and two spheres is defined as illustrated in Figure 3.1. In this project, basic sphere glyphs, basic cylinder glyphs and arrow glyphs, defined by a combination of the cone and the cylinder glyph, were used.

Once the surface is defined, it is intersected with a ray cast from the eye position.

$$\vec{r} = \lambda \cdot \vec{s} + \vec{p_{eye}} \tag{3.6}$$

In order to rotate the glyph, the eye position is orbited around the glyph before doing the intersection. That way local orientation is emulated when rendering the surface.

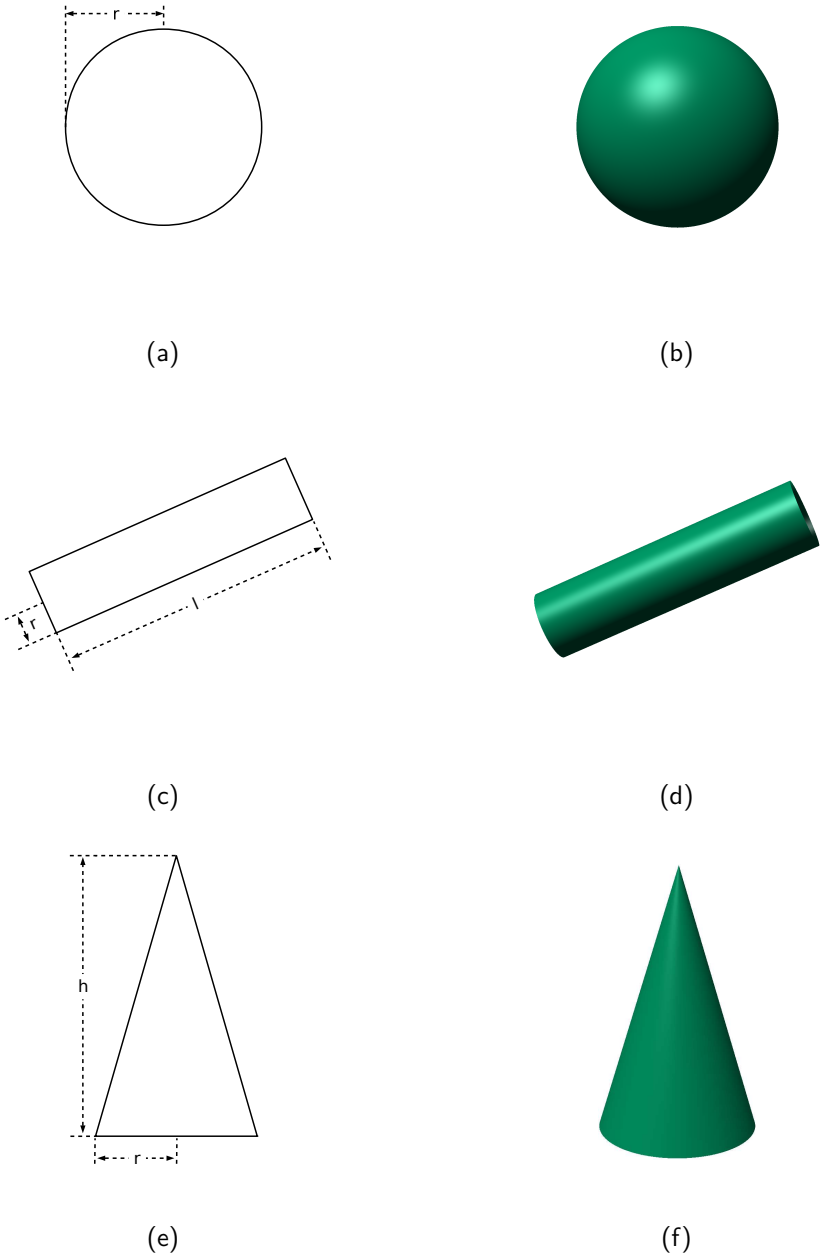
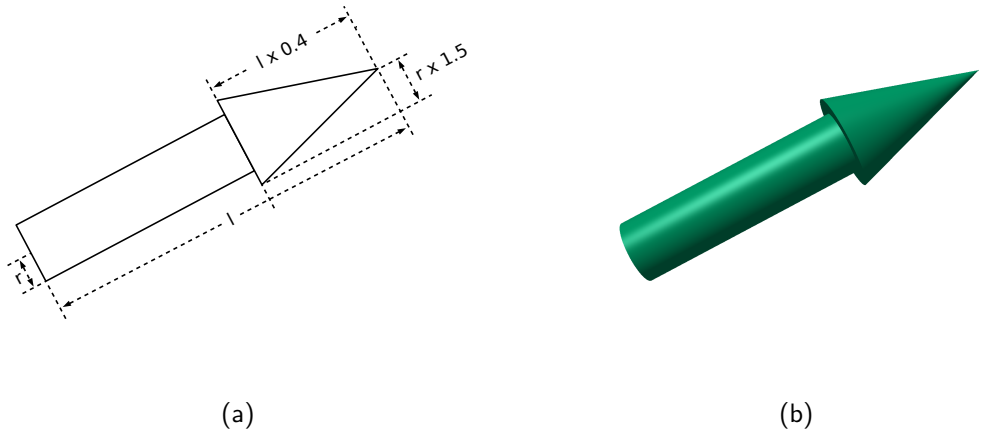


Figure 3.2: A summary of basic glyph types. The row on the left hand side schematically illustrates the parameters needed to define the implicit surfaces for the different glyph types. In the images shown on the right hand side *Blinn-Phong Shading* has been applied to the surfaces.



(a) (b)

Figure 3.3: The arrow glyph type and the associated parameters. The arrow glyph combines the cylinder and the cone glyph. The overall length of the arrow and the cylinder radius are sent to the shader program. The dimensions of the tip are then extracted from these two parameters as illustrated in (a). In (b), BLinn-Phong Shading has been applied to the surface.

3.1.2 Implementation

All calculations regarding the geometry of the glyph are done on the GPU. The position of the glyph objects is transferred to the render pipeline as `GL_POINTS`. Further information sent to the vertex program consists of all parameters describing the dimensions of the glyph (see Figures 3.2 and 3.3 for a summary) as well as a quaternion specifying the local orientation. If possible, texture coordinates are used to store the information. Otherwise, additional vertex attribute arrays can be used. The vertex program then does all calculations which are constant for all fragments of the glyph. This includes the transformation of the camera position and the light position to the local coordinate system of the glyph. Additionally, a rotation matrix is obtained from the quaternion. However, this step is not necessary for sphere glyphs, since their reorientation does not visually change the glyph object.

The vertex program also determines the point size necessary to enclose the whole glyph. The point size can dynamically be set in the vertex program through the built-in vertex shader variable `gl_PointSize`. In order to use this variable, it has to be enabled beforehand in the current OpenGL context. The actual size of the point sprite is calculated as follows: Depending on the selected glyph type, the normalized device coordinates of the corners of the surrounding object are computed. In the case of sphere glyphs, a bounding rectangle is sufficient. Therefore, only four corners have to be transformed in this case. In the case of non-centrosymmetric glyph types, the local rotation has to be taken into account as well. Thus, all other aforementioned glyph types require a bounding cuboid, consisting of eight corners. The result are maximum and minimum values for

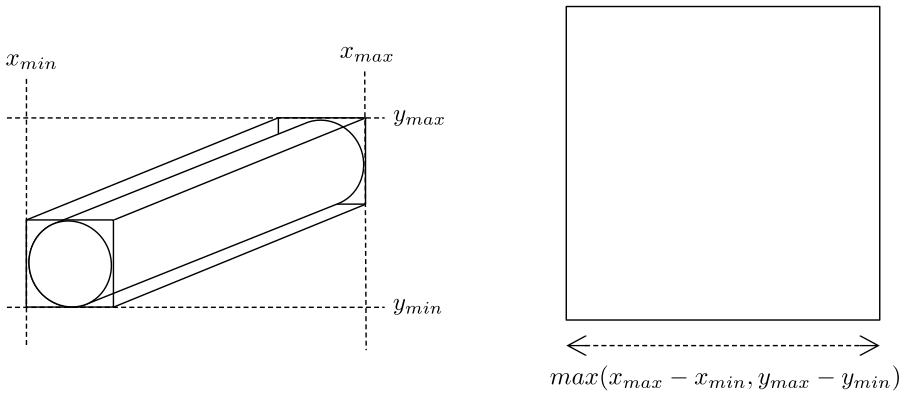


Figure 3.4: Determining the pixel size for a point sprite. In order to yield maximum and minimum values for the dimensions of the point sprite, screen space coordinates of the eight corners of the surrounding cuboid are computed. The point size is then set according to the maximum side length defined by that coordinates.

the fragment positions the point sprite has to cover to surround the whole glyph. The point size is then set according to those values. Figure 3.4 illustrates the calculation of the point size for a cylinder glyph.

In the fragment program, the coordinates of the current fragment in the local coordinate system of the glyph are calculated. This is done by transforming the screen space coordinates to view space using the parameters of the viewport (top t, bottom b, left l, right r), the viewport size (width w, height h) and the position of the near clipping plane (z_N). The view space coordinates can then be calculated as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{VS} = \begin{pmatrix} \frac{x_{SS}}{w} \cdot (r-l) \cdot z_N + l \cdot z_N \\ \frac{y_{SS}}{h} \cdot (t-b) \cdot z_N + b \cdot z_N \\ -z_N. \end{pmatrix} \tag{3.7}$$

These coordinates are then transformed to object space by multiplying them with the inverse `GL_MODELVIEW` matrix. Subsequently the object space position of the fragment is transformed to the local coordinate system of the glyph, which is done by subtracting the object space position of the glyphs pivot point.

If not using sphere glyphs, the rotation matrix calculated in the vertex shader is then multiplied with these glyph space coordinates. Using the fragment's and the camera's glyph space position the correctly rotated viewing ray can be calculated, it is then given by

$$ray = normalize(fragPos - camPos). \tag{3.8}$$

In order to obtain the intersection point, different quadratic equations have to be solved, depending on what glyph type is used. If none of these equations can be solved, the ray did not hit any of the implicit surfaces and the fragment is discarded. Otherwise, this yields a number of possible intersection points. To test whether an intersection point is inside the boundaries of the glyph, different criteria can be defined according to the glyph type. When using arrow glyphs, two quadratic equations are to be solved, which yields two possible intersection points for the cylinder and the cone, respectively. In order to test which of those points are feasible (i.e. inside the boundaries of the glyph), the intersection points are first projected onto the x-axis of the glyph's local coordinate system. The boundary conditions of the cone and the cylinder can than be applied to the four resulting scalar values. The intersection of the cylinder glyph can be addressed using the same principle. When using a combined glyph type, more than one intersection point can be feasible at the same time. For instance, feasible intersection points for both the cone and the cylinder can be obtained, when using arrow glyphs. Therefore, a priority order is applied to the intersection points as an additional criterion. If an intersection point is feasible, all intersection points with lower priority are set infeasible.

Depending on what surface is hit by the ray, the normal is computed in glyph space accordingly. Subsequently, local lighting is computed per pixel by using the *Blinn-Phong shading model* [Bli77]. The Blinn-Phong shading model is based on the shading algorithm developed by Phong [Pho75]. Here, the local lighting is separated into three terms: *ambient*, *diffuse* and *specular*. While the ambient term is a fixed value, the diffuse and the specular term have to be calculated. In order to compute the diffuse term, which distributes reflected light evenly in all directions, the dot product of the normal N and the light direction vector L has to be obtained. For the specular term, the dot product of the reflection vector R and the viewing ray V is needed. Blinn modified the calculation of the specular term, by using the halfway vector H, which is the sum of L and V.

Once the intersection point of the current fragment has been determined, the screen space depth is computed and written to the depth buffer. This can be done by transforming the glyph space coordinates to object space and then multiplying the result with the `GL_MODELVIEWPROJECTION` matrix.

3.2 Direct Volume Rendering

In order to render isosurfaces based on a three dimensional scalar field, direct volume rendering was used by implementing a front-to-back ray marching approach on the GPU.

3.2.1 Algorithm

Ray Marching is an image order algorithm for direct volume rendering, where a ray is cast from the position of the camera through every pixel. The resulting color of the pixel is computed by evaluating the volume rendering integral along the ray. This integral is

representing a simplified optical model which only contains *absorption* and *emittance* and ignores scattering properties of light. The intensity reaching the observer is given by

$$C = \int_0^\infty c(t) \cdot e^{-\tau(0,t)} \tag{3.9}$$

with $c(t)$ being the intensity at location t and $\tau(0,t)$ the loss of intensity caused by absorption. In praxis, C is computed by sampling scalar values alongside the ray and accumulating the intensity assigned to these values, e.g. by a predefined transfer function. The computation is done iteratively using the over operator, which was first described in [PD84] (see also [Lev90]). C_{in} and α_{in} are the values computed in the preceding step, C and α are the values obtained by applying the transfer function to the sample and C_{out} and α_{out} are the newly computed values. Here, the color value is given by

$$C_{out} = C_{in} + C(1 - \alpha_{in}) \tag{3.10}$$

and the alpha value is separately computed with

$$\alpha_{out} = \alpha_{in} + \alpha(1 - \alpha_{in}). \tag{3.11}$$

The final color is then defined as

$$C_{final} = C_{out}/\alpha_{out}. \tag{3.12}$$

The approach described above generally applies to generating volumetric images. It can also be used to extract *isosurfaces* from scalar fields. Isosurfaces are sets of points which all exhibit the same constant value in the scalar field. This value is called *isovalue*. As mentioned before, the resulting image of the ray marching algorithm depends on the transfer function used. Isosurfaces can, therefore, be extracted by using a transfer function which assigns an α value of zero to all samples except the ones containing the user defined isovalue.

The light intensity is accumulated using interpolated samples at discrete locations. Thus, even when using a rather small step size, the algorithm is likely to miss the exact location of the isosurface. Therefore, for every intersection point, the two enclosing samples have to be found. In every iteration, the difference of the sample's scalar value and the isovalue is computed and stored. The result is then compared to the value computed in the last iteration. If both values have opposite signs, the isovalue has been passed and, therefore, the two enclosing samples have been found (see Figure 3.5 for an illustration).

The difference calculated before can then be used to find the exact location of the isosurface. If two samples s_a and s_b are taken at the enclosing points of the intersection point, the value s_c at the intersection point can be obtained using linear interpolation (see Figure 3.5). If the difference of s_a from the isovalue is Δ_a and the difference of b from then isovalue is Δ_b, s_c can be calculated as

$$s_c = s_b \frac{\Delta_a}{\Delta_a + \Delta_b} + s_a \frac{\Delta_b}{\Delta_a + \Delta_b} \tag{3.13}$$

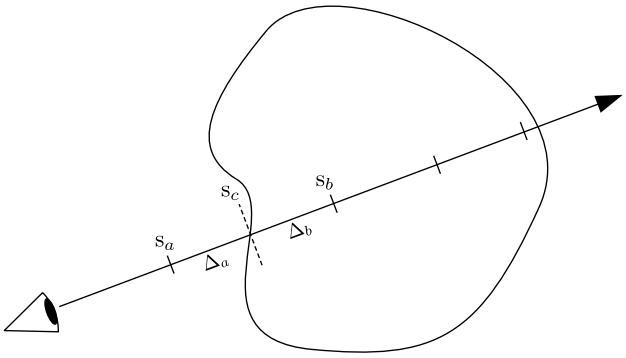


Figure 3.5: Extracting an isosurface using ray marching. After every step, Δ_a and Δ_b are compared. If both values have opposite signs, the isosurface has been passed by the ray. In this case, the exact location of the intersection point can be obtained using interpolation.

3.2.2 Implementation

When implementing the algorithm described above using OpenGL and GLSL, a color texture was precomputed before the rendering. The color of a sample is then determined by sampling the color texture rather than evaluating a transfer function.

In order to reduce the per-pixel workload, the direction vector of the ray has been calculated in a two-pass approach as described in [KW03]. In the first render pass, the backside of the cuboid representing the volume texture is rendered. In OpenGL, this can be achieved by enabling front face culling and then rendering the cube into a texture with the color value of every vertex set to its texture coordinates. For the second render pass, the same cuboid is drawn again, only this time the back is culled and the front is rendered. The assignment of texture coordinates for the back and the front of the cube is illustrated in Figure 3.6.

The actual ray for every fragment in texture space can then be computed by the fragment program. The interpolated texture coordinates of the front side deliver the starting point for the ray tracing algorithm in texture space. The following GLSL code snippets shows how the direction vector of the ray can be computed. Here, the texture coordinates at the backside of the cube are subtracted from the texture coordinates of the current fragment:

```
vec3 ray_end   = texture2D(back_buffer, texc).rgb;
vec3 ray_start = gl_TexCoord[0].stp;
vec3 ray_dir   = normalize(ray_end - ray_start);
```

The actual ray is then given by

```
vec3 ray = ray_start + delta*ray_dir;
```

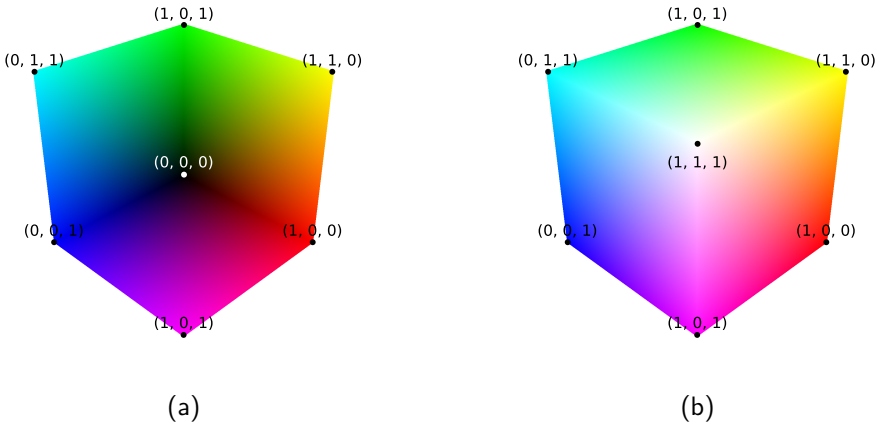


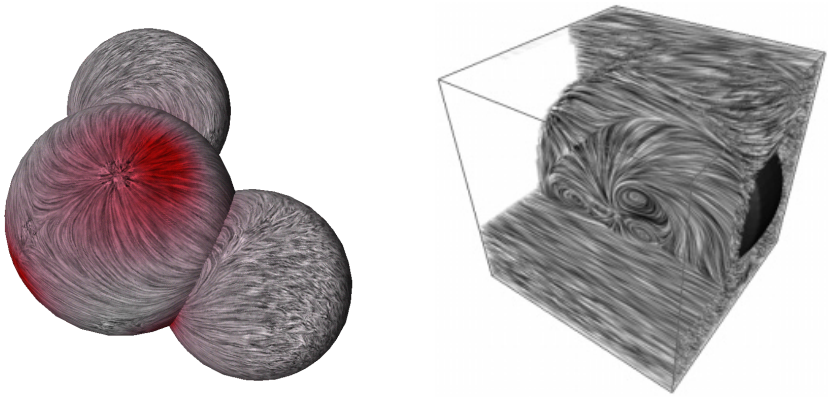
Figure 3.6: The texture coordinates stored in the color values (c.f. [KW03]). (a) shows the backside of the cube, (b) shows the front.

where `delta` is the step size used for the ray marching. This also delivers the maximum ray length, which can be used as a criterion to stop the ray marching. The ray traversal now can take place in texture space and no coordinate transformations are necessary.

The ray marching is practically done by precomputing a vector which is pointing in the direction of the ray and subsequently adding that vector to the starting. In every iteration, the scalar field is sampled and the difference with the isovalue is calculated as described above. Once the samples which are enclosing the isosurface have been found, the fragment is shaded.

For this purpose, the corresponding color is obtained, e.g. from a color texture, and the gradient is computed based on the neighboring scalar values. The normals can be obtained by computing the gradient of the scalar field at the respective position of the intersection point using central differences. However, the exact texture coordinates of the intersection point have to be computed first. Since the ray traversal takes place in texture space, it is sufficient to take the texture coordinates of the last sample before the intersection point into account and add an accordingly scaled vector in the direction of the ray. The color and the gradient of the intersection point are then set using texture samples at this location. Once the normal and the color are available, the computation of local lighting can take place and the result is accumulated as described above. Additionally, the backside of the surface is drawn darker to give additional visual clues.

One of the acceleration techniques used in this implementation is *early ray termination* [KW03]. Since the colors are accumulated front-to-back, the ray marching can be stopped if the alpha value is greater than one. In this case, the fragment is completely opaque and objects behind it are not visible. Another criterion for ray termination is the total length of the ray, which can be compared to the maximum ray length obtained before.



(a) (b)

Figure 3.7: LIC applied to 3D vector fields. (a) is an example for the application of LIC on a tessellated surface (taken from [BSH97]). Here, the vectors are projected onto triangles. (b) shows the use of clipping surfaces in combination with LIC to illustrate vector field features in a 3D vector field (taken from [RSHTE99]).

3.3 Line Integral Convolution (LIC)

Line Integral Convolution (LIC) is a texture synthesis technique, which can be used to visualize vector field features. It was first described by Cabral et al. [CL93], who extended the *spot noise* algorithm introduced by van Wijk [Wij91]. Cabral et al. [CL93] combined this approach with known line drawing techniques and developed a DDA (Digital Differential Analyzer) based convolution algorithm as well as the line integral convolution.

The original LIC implementation has a lot of redundancies. This was taken care of by Stalling et al. [SH95], who describe an improved LIC algorithm, which reuses information already computed before. They also introduce the use of fourth order numerical integration in order to determine the ideal step size adaptively for every step.

The original LIC only refers to 2D vector fields defined in a 2D Cartesian grid. However, LIC actually can be applied to arbitrary surfaces in 3D space by using both 2D or 3D vector fields. In [For94, LVW95], the application of the algorithm to curvilinear grids which can be parameterized by 2D coordinates, is described. A method to use LIC on general 3D surfaces was given by Battke et al. [BSH97]. Here the surface is first tessellated and a local coordinate system is computed for all triangles. LIC is then applied to every triangle respectively. When using 3D vector fields, a straightforward application of the LIC method often leads to problems with occlusion of interesting details. There are several possibilities to circumvent these issue. In [BSH97], the vector field is projected on the 2D surfaces, while maintaining their magnitude. This leads to areas with little vector magnitudes to contain the original noise texture (see Figure 3.7a). Another way to use LIC with 3D vector fields are clipping surfaces. This method has been described in [RSHTE99], an example is shown in Figure 3.7b.

3.3.1 Algorithm

In all approaches, the intensity of a pixel is computed by convolving a filter kernel with a texture containing random noise. In general, the pixels intensity is defined by

$$I(x_0) = \int_{s_0-L}^{s_0+L} k(s - s_0)N(\sigma(s))ds, \tag{3.14}$$

where $\sigma(s)$ is the path being used to do the convolution, parameterized by the arc length, N is the noise texture and k is the 1D filter kernel. In praxis, a discrete version of that method is used. Here, a box filter can be used to compute the intensity. The intensity is then defined by

$$I(x_0) = \frac{1}{2L+1} \sum_{i=-L}^{L} N(x_i), \tag{3.15}$$

with x_i being discrete points on the sampling path.

Using the DDA approach the vector field is sampled once for every pixel and then the noise texture is convolved with a straight line which is parallel to the sample. Consequently, the intensity of the noise texture is integrated along the chosen direction in the vector field. With $\vec{v_0}$ being the sample of the vector field at position x_0 and λ being a scale factor, the path can iteratively be defined as

$$x_i = x_{i-1} + \vec{v_0}\lambda \tag{3.16}$$

For the actual LIC algorithm, Cabral et al. [CL93] suggest starting a stream line at the center of all pixels which is defined by

$$x_i = x_{i-1} + \vec{v_{i-1}}\lambda_{i-1}. \tag{3.17}$$

Here the λ_i are chosen accordingly to make sure, that the stream line exactly hits the nearest pixel. The actual LIC algorithm is, therefore, more complex and less efficient, but represents the vector field features in a more accurate way. The difference between the two approaches is illustrated in Figure 3.8. In both cases the application of the kernel results in a visual coherence between pixels which also show similarities in the vector field, when sampled at the respective position. Both cases, therefore, allow illustrating vector field features, such as domain walls and the location of critical points. A comparison of the results of both approaches is given in Figure 3.9.

3.3.2 Implementation

In this project, a variation of the classic LIC algorithm was implemented on the GPU to be used in combination with axis aligned clipping planes and isosurfaces. Both the LIC approach for 2D slices and the algorithm for the isosurfaces have been implemented using GLSL.

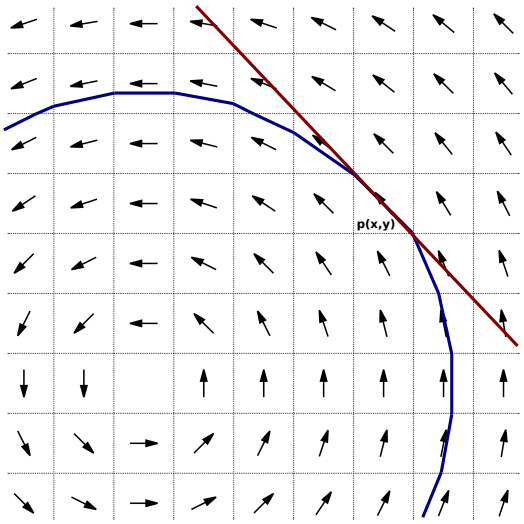
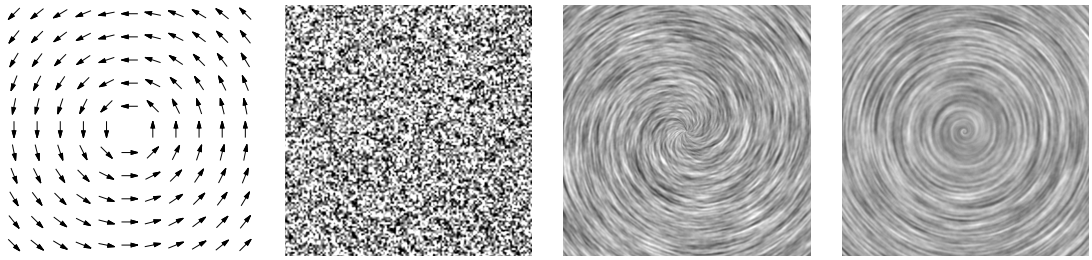


Figure 3.8: The different paths used in DDA convolution and LIC. The path used in the LIC algorithm is shown in blue, whereas the path in red illustrates the straight line used in the DDA approach.



(a) (b) (c) (d)

Figure 3.9: The DDA approach and the LIC algorithm applied to a procedural circular vector field. The vector field in (a) is defined by $(x, y) = (-y, x) * 3.54$, it is, therefore, vanishing at the origin $(0, 0)$. The vector field is combined with a noise texture (see (b)) using the both DDA (c) approach and the LIC (see (d)). Both approaches illustrate the overall shape of the vector field. However, the vector field is represented more accurately by the LIC generated pattern. This is clearly visible in the areas around the critical point. Here, the DDA generated image implies a sink/source which is not present in the original vector field.

```
float LIC(int l) {
    int i;
    vec3 v;
    vec3 stp = gl_TexCoord[0].stp;

    float colLic = texture3D(randNoiseTex, stp).a;
    v = texture3D(uniGridTex, stp).xyz;

    for(i = 0; i < l; i++) {
        stp -= v;
        stp = clamp(stp, 0.0f, 1.0f);
        colLic += texture3D(randNoiseTex, stp).a;
    }

    stp = gl_TexCoord[0].stp;

    for(i = 0; i < l; i++) {
        stp += v;
        stp = clamp(stp, 0.0f, 1.0f);
        colLic += texture3D(randNoiseTex, stp).a;
    }

    colLic /= float(l+l+1);
    return colLic;
}
```

Listing 3.1: Basic convolution algorithm in GLSL (c.f. [BC11]).

A straightforward implementation of the DDA convolution on the GPU, is given in [BC11]. The program code can be found in Listing 3.1. It basically implements the DDA convolution described above in the fragment program. The vector grid as well as the noise texture are sampled using bilinear interpolation. The vector field is sampled once per fragment, whereas the noise texture is sampled on equidistant steps on a line oriented according to the local vector field. The path is then followed backwards and forwards, starting at the fragment position. Subsequently, the color values obtained from the noise texture are accumulated. Finally, the accumulated color intensity is normalized. Applying the LIC only to visible fragments, rather than precomputing it for the whole vector field, saves computation time and bandwidth, especially when using 3D vector fields.

The implementation described above was used as a basis to apply LIC to arbitrary surfaces. Both slices and isosurfaces, generated by the ray marching algorithm described in Section 3.2, have been used. However, several modifications have been applied to the original implementation. The code given by [BC11] actually implements DDA convolution, since the vector field is sampled only once per fragment. In this project streamlines, have been used for more accuracy.

Applying LIC to a surface using a 3D vector field often leads to noisy spots in areas in which the vector field is nearly orthogonal to the surface. This can be desirable in some cases, e.g. when identifying exactly those areas. Nonetheless, it is practical to project the sampled vectors on the surfaces, in this case. For a general surface with the normal $\vec{n}$,

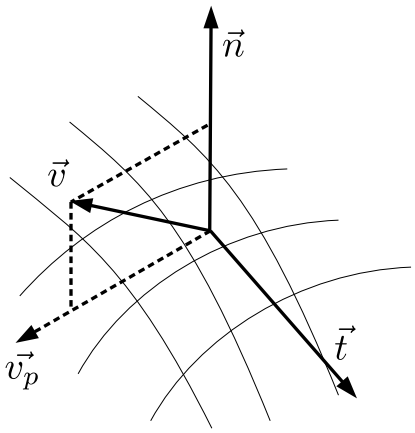


Figure 3.10: Projection of vectors onto the surface. $\vec{n}$ is the local normal of the surface. The tangent $\vec{t}$ can be computed by calculating the cross product of the sampled vector $\vec{v}$ and the normal. The projected vector $\vec{v_p}$ is the bitangent of the local tangent space coordinate system and is therefore obtained by the cross product of $\vec{n}$ and $\vec{t}$.

this can be done by first computing the tangent, which is orthogonal to both the sampled vector $\vec{v}$ and the normal. This can be achieved using the cross product:

$$\vec{t} = \vec{n} \times \vec{v} \tag{3.18}$$

In the same way, the bitangent can be calculated:

$$\vec{v_p} = \vec{n} \times \vec{t} \tag{3.19}$$

The bitangent is in this case, the projected and normalized vector $\vec{v_p}$. The whole process is illustrated in Figure 3.10.

In the case of axis aligned slices, the calculation gets much easier. It reduces to

$$\vec{v_p} = \begin{pmatrix} x_v x_n \\ y_v y_n \\ z_v z_n \end{pmatrix}. \tag{3.20}$$

In fact, it is sufficient to set the according coordinate to zero.

In order to visualize the orientation of polarization domains, LIC has been implemented for the extracted isosurfaces. To create a noise free and clear result, the vectors are projected onto the isosurface. Following the surface exactly is not possible in this case, since it is created via ray marching and, therefore, no parameterization of the surface is available. However, it is sufficient to project only the first vector onto the isosurface to create a visual coherence.

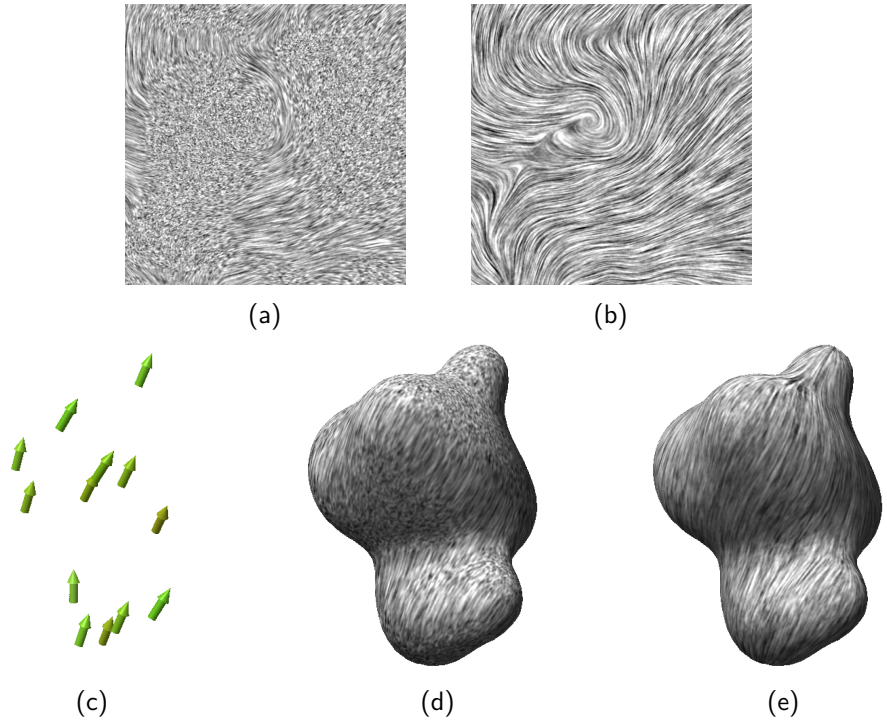


Figure 3.11: Comparison of LIC with and without vector projection. (a) and (b) show LIC applied to axis aligned slices. (d) and (e) show the LIC applied to the vector field shown in (c). In both cases, projecting the vector field onto the surface eliminates the noisy areas, where vectors tend to be orthogonal to the surface.

In an interactive environment, the user is expected to visually explore the LIC by changing the camera position and orientation. Therefore, zooming by the user should be taken into account. Here, the chosen resolution of the random noise texture has a great impact on the resulting texture. If the resolution is too high, the stream lines are not distinguishable any more when zooming out too much. On the other hand, when it is too small, the pixels of the noise texture become visible when zooming in. The reason for that is that the LIC calculation is done on a per-fragment level, whereas the size of the texture is fixed. Letting the user scale the texture coordinates used to sample the random noise texture is one possible approach to circumvent this issue. In effect, this leads to tiling the random noise texture. It should, therefore, be taken care, that the original texture is not too small, since otherwise, repetitive patterns could be visible in the LIC.

Choosing the right step size is crucial for the quality of the final image. Ideally, the path would follow the current direction until it hits either the next pixel in the random noise texture or the next vector from the vector grid, whichever is closer. In order to make the resulting image controllable, all sampled vectors are normalized and rescaled by a user defined scalar factor. Adaptive scaling, like e.g. described in [CL93], seems to be a more accurate approach but it requires more computation time.

Changing the stream length, as well as the vector scaling, has a great impact on the resulting image. If the stream length is too low, the picture gets very noisy, however, if it is too long, the final image is smeared out too much and the overall contrast gets very low. As stated in [CR08], this choice influences the overall performance of the algorithm. Since it is important to maintain interactivity in this implementation, the smallest value delivering still good results is used. In this case a stream length of ten used for all paths has provided the best results.

3.4 Combining Arrow Glyphs and Isosurfaces

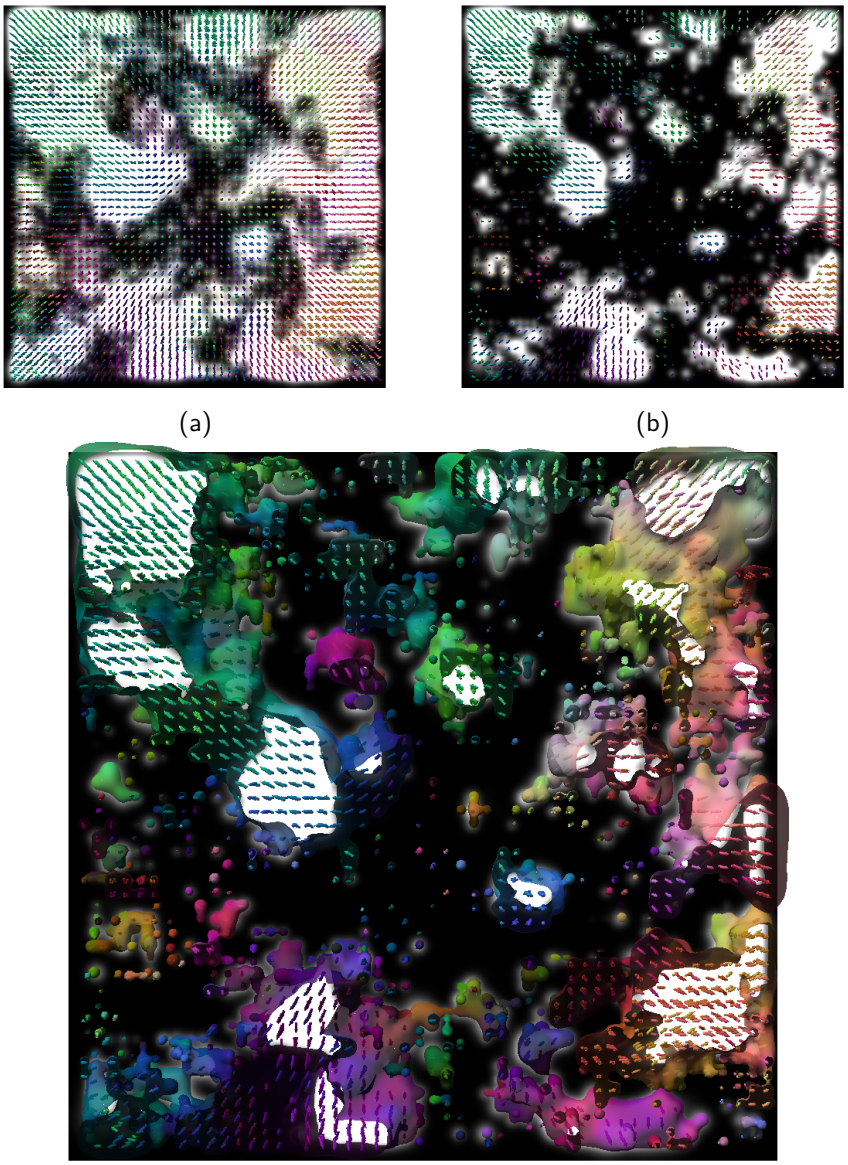
In order to emphasize vector clusters forming the domains, a combination of the arrow glyph based representation described above and semitransparent isosurfaces is used. This can be achieved by first filtering the vectors according to the curl magnitude. In order to do that, the curl magnitude is obtained at the positions of the vectors using nearest neighbor sampling. When the value of the curl magnitude is above a user-defined threshold, the vector is filtered out. Subsequently, Gaussian kernels are placed in a density map for all vectors which were not filtered out. The resulting density map then resembles the actual distribution of the vector clusters. Isosurfaces can now be extracted from that density map and combined with the arrow glyph representation described in Section 3.1. If LIC is applied to that isosurface, the resulting texture represents the movement inside the isosurface, rather than the original vector field at the exact location of the surface. In this case, this is a desired effect, since the isosurface is used to visualize additional properties of the cluster it is encircling. When combining the arrow glyphs with the isosurfaces, two additional issues have to be addressed.

The first issue is the combination of semitransparent isosurfaces, obtained by ray-marching, and the opaque arrow glyphs. Combining those two visualization methods requires a modification of the raymarching algorithm. Thus, a modified version of the two-pass method described above has been used to implement a z-buffer test inside the ray marching algorithm. First, all opaque objects are rendered into a frame buffer object. This delivers a buffer containing depth values and colors of the scene. When rendering the backside of the cube, in addition to the texture coordinates, screen space coordinates are written to the buffer. In the final render pass, where the ray marching takes place, the ray is traversed both in texture space and in screen space. Every iteration the screen space depth of the current position is compared with the one in the source buffer. When the screen space z-value of the current position on the ray is larger than the one stored in the source buffer, the traversal is terminated and the current accumulated color is combined with the one stored in the source buffer. This actually enables an additional criterion for the early ray termination.

Another issue that has to be addressed is the fact that parts of the arrow glyphs tend to stick out of the encircling isosurface, which disrupts the visual appearance of the smooth isosurfaces. If the isosurfaces are based on a density grid, obtained by Gaussian kernel splatting as described above, the position of the isosurface can be aligned with the arrow glyphs by using a kernel size according to the arrow length. In this project, both the

arrow glyphs and the Gaussian kernel size was set according to the actual vector length, linearly scaled for better visibility.

Figure 3.12 shows texture slices based on the density grid obtained with the method described in this section.



(a) (b)

(c)

Figure 3.12: Texture slices illustrating the density grid. In (a), no filtering has been applied to the vector field. However, the approximate shape of the domains is already noticeable, since vector magnitude and orientation correspond in this case. (b) shows the final density grid, with the domains being extracted by filtering according to the curl magnitude. In (c), a cutout of the isosurfaces extracted from the density grid can be seen.

4 Results and Discussion

One of the main goals of the SciVisContest was to show the connection between atom displacements and polarization. Thus, the methods described in the previous chapter were applied to the provided data in order to gain insight into the development of both the polarization and the changes in the perovskite structure of the material. This allows for the determination of the time span, in which the phase transition is taking place. In order to reduce visual clutter, both the glyph ray casting and the direct volume rendering were combined with axis aligned clipping planes, which produce a cube shaped cutout of the data.

4.1 Performance of the Implementation

The data processing and visualization methods described in the previous chapter were applied to the time dependent vector field defined by the displacement of the titanium atoms (125,000 vectors in total). The time needed for data related computations was measured separately, since those computations are only necessary if the time step in the data set changes. The test system was an Intel Core 2 Quad Q6600 with 4 GB RAM and a NVIDIA GeForce GTX 560 with 1 GB VRAM.

The major workload for the data related computations in each time step includes the extraction of the uniform grid, the calculation of the curl magnitude and the calculation of the density map. Additionally, the time needed to locate critical points was measured. The results for the time dependent calculations are given in Table 4.1.

In order to test the performance of the actual rendering several representations were compared with each other. Here, some reasonable scenarios were tested using arrow glyphs and both opaque and semitransparent isosurfaces with different textures. The resolution was 1024x768 and the camera was fully zoomed in. All vector were filtered

Calculation	[s]
Extraction of a uniform grid	0.18
Calculation of curl magnitude	0.08
Generation of the density grid	4.94
Location of critical points	100.5

Table 4.1: The performance of the calculations necessary for the time dependent data in seconds.

Render mode	**[Frames/s]**
Arrow glyphs	85
Semitransparent isosurfaces	14
Arrow glyphs and semitransparent isosurfaces using a color texture	16
Arrow glyphs and opaque isosurfaces using LIC (streamlength 10)	40

Table 4.2: The performance of rendering in *frames per seconds*.

using a reasonable threshold for the curl magnitude, which resulted in 17968 visible vectors (about 14% of the original count). Finally, the combination of the arrow glyphs and the isosurfaces was tested using both a color texture and the LIC. The results for the rendering related calculations are given in Table 4.2.

In general, interactivity cannot be maintained when using dynamic data, however for static data an interactive frame rate can be achieved. In this case, the bottleneck seams to be the semitransparent isosurfaces obtained by raymarching.

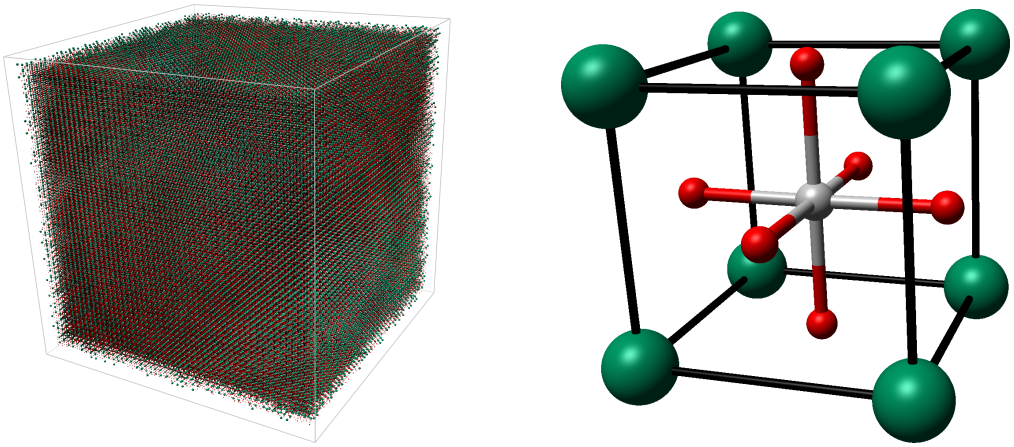
4.2 Atom Movement over Time

In most ferroelectric materials, the polarization heavily depends on local atom displacements. The overall atom movement over different time spans can, therefore, be useful to determine the changes in the perovskite structure, as well as the deformation of the material. The changes in the perovskite structures of the unit cells can deliver hints about their state regarding the phase change. The deformation of the material is important as well, since the material is expected to change its shape while undergoing temperature changes.

In order to get a first impression of the material and the distribution of the unit cells, the perovskite structure was rendered using sphere and stick ray casting. Figure 4.1 shows the whole structure as well as a single unit cell. For time independent data the atom positions were linearly interpolated for every rendered frame. Applying this method to the time dependent data without any averaging highlights the strong oscillation of the individual particles representing the atoms. When averaging the positions over 25 frames, as suggested in Chapter 2, no vibration can be perceived. However, the overall movement of the atoms over the whole time span gets visible when using an adequate frame rate.

The arrow glyph ray casting described in 3.1 was applied to the displacement field covering several time windows. The length of the arrow glyphs corresponds to the actual vector length, but has been linearly scaled for better visibility. The arrow glyphs midpoint is positioned at the position of the vector and the glyph is oriented according to the vector orientation. The deformation of the material over time can be visualized by color coding the arrow glyphs based on their orientation and their magnitude.

Figure 4.2 shows the displacement of all atoms over a time span of 10 ns, thus, the whole time series. The visualized data has been reduced to a slice of 50×50×10 unit



(a) (b)

Figure 4.1: Visualization of a unit cell using GPU ray casting. (a) shows the entire cube-shaped data set. (b) contains a single unit cell. Here, Ba atoms are shown in green, the red spheres represent O atoms and the Ti atom at the cell center is rendered in grey. The sphere glyphs are scaled according to the atoms' van der Waal radii.

cells. The color coded vector orientation illustrated in the texture slices shows that the movement of the atoms tends towards the center, which implies an overall shrinking of the structure. When coloring the arrow glyphs according to vector magnitude, the stronger movement of atoms in off-center areas is highlighted, while atoms near the center seem to move much less.

When using the smallest possible time window of one frame and coloring the glyphs according to the element type, it is possible to gain insight in the strength of the movement of the different element types. As already mentioned in Section 2.2, the different atom types are not equally mobile. In general, the oxygen atoms are the most mobile.

Figure 4.4 shows some of the key time steps, with a length filter applied. At the beginning, a series of short peeks in the overall displacement length can be perceived. This is also visible when plotting the average displacement length (see Figure 2.5) and is probably a byproduct of the simulation. At about 800 ps the atoms start getting more mobile, which lasts until the end of the simulation. This is staggering since the mobility is expected to get lower when decreasing the temperature. Therefore, the increasing mobility might have something to do with the ongoing phase transition, which until then has been counteracted by the temperature decrease.

Using a smaller displacement time frame shows other characteristics of the material. In Figure 4.3 arrow glyphs have been rendered for some key moments of the time series using a time window of 25 frames for the displacement field. The material is indeed shrinking as stated before, however, at about frame 410 the shrinking stops and the structure is

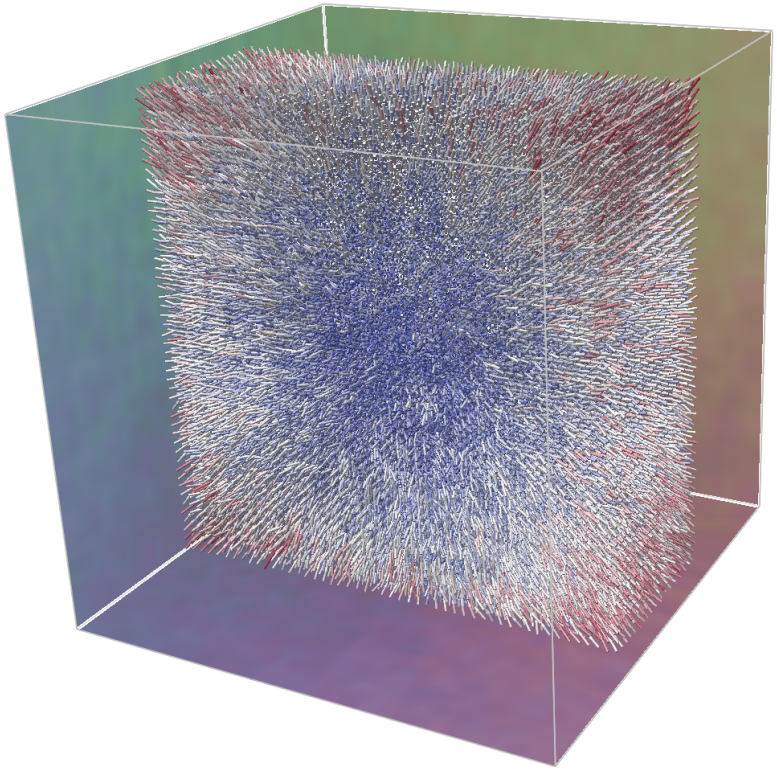


Figure 4.2: The displacement field of the Ti atoms over a timespan of 10 ns. The arrow glyphs are colored by the displacement field magnitude using hot-cold coloring scheme (blue: low values; red: high values). The low mobility of the atoms near the center and the fact that the rest of the atoms clearly move toward the center illustrate the contraction of the material. Additionally, slices color-coding the vector field orientation at the boundaries of the data set are shown. They also convey the overall movement of the atoms toward the center.

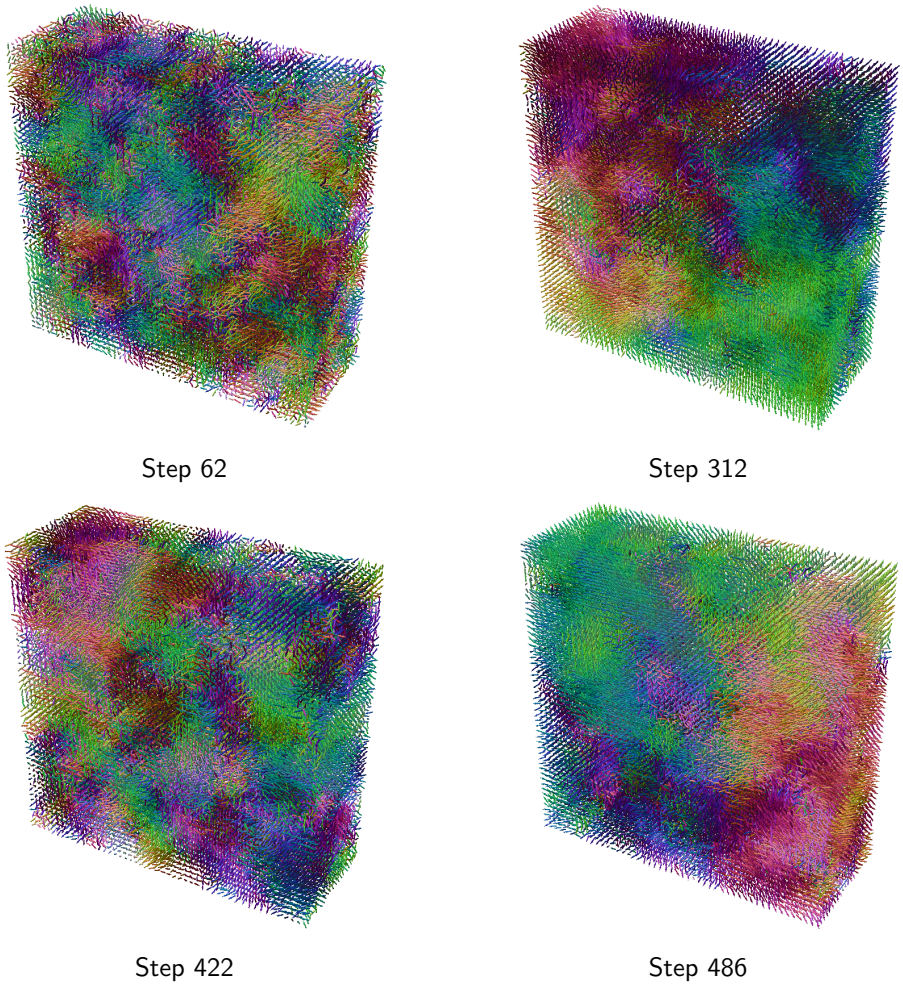
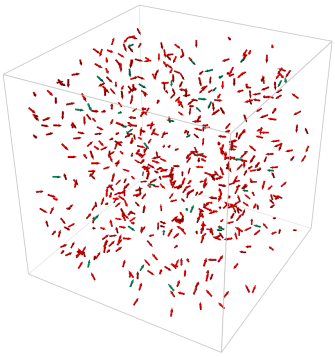
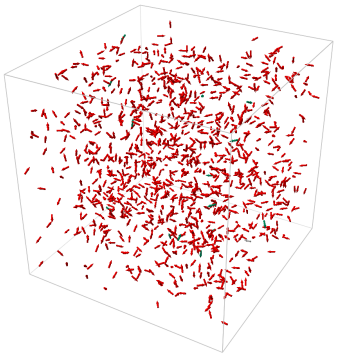


Figure 4.3: Rendering of the displacement field using arrow glyphs, while using the coloring according to the vector orientation. Here, the time window for the displacement field is 25 frames. (a) illustrates the small evenly distributed domains at the beginning of the time series. The temperature related deformation in (b) is illustrated by large clusters of vectors pointing towards the center of the material and, therefore, exhibiting similar orientation. The smaller clusters in (c) illustrate, how the expansion of the material is beginning to take effect. In (d) most of the vectors are pointing away from the center.

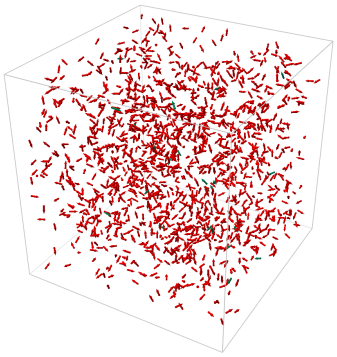
Figure 4.4: Rendering of atoms' displacement field. Here, a time window of only one frame was used to compute the displacement field for all atom types using averaged atom positions (25 frames). Several pulsating peaks can be observed right at the beginning of the time series. Starting at about 6000 ps the atoms are becoming more mobile. Their mobility further increases until the end of the time series is reached.



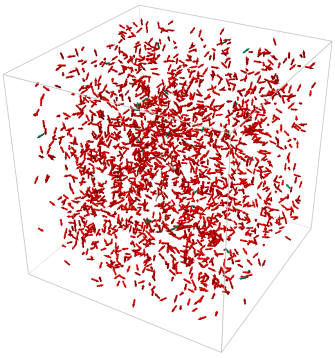
6250 ps



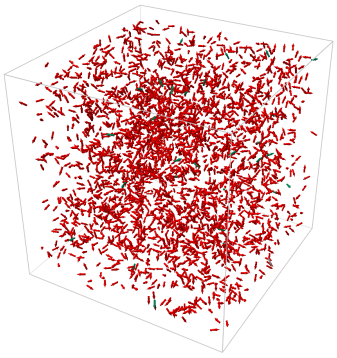
8250 ps



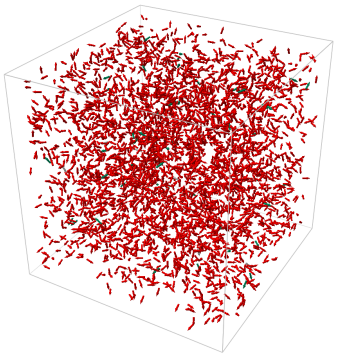
8450 ps



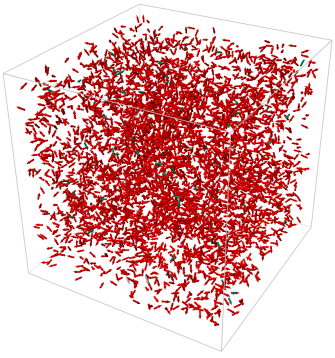
8650 ps



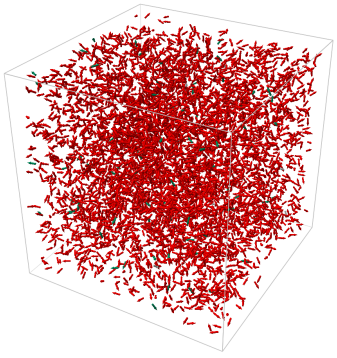
8850 ps



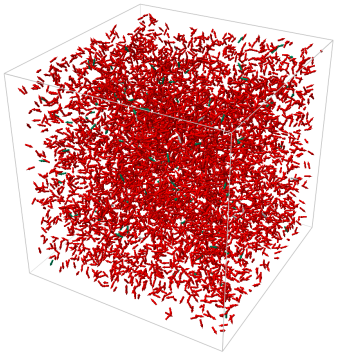
9050 ps



9250 ps



9450 ps



9650 ps

expanding again. This indicates the phase transition from the cubic to the tetragonal phase, since it is accompanied by cell elongation.

Texture slices can give clues about the movement of the atoms over time, although they only represent a small part of the data. In order to get a good overview, three axis aligned texture slices were rendered. Their exact position can interactively be changed by the user. Figure 4.8 shows key moments of the time series using 2D LIC and critical points, which are represented by sphere glyphs. The movement towards the center is highlighted by the LIC as well as the distribution of the critical points in the vector field.

4.3 Visualization of Domains

Since the dipoles in the material tend to align to each other when undergoing a phase transition, domain sizes and shapes are an indication for the phase transition. These domain features can be visualized by using arrow glyphs in combination with adequate filtering, as well as volume rendering. These two methods can also be combined.

When visualizing a time step of the displacement field using arrow glyphs using either orientation or magnitude coloring vector clusters can easily be identified. The rather high magnitude inside clusters is also highlighted. Figure 4.5 shows a comparison of both coloring modes when applied to the last frame of the time series of the displacement field.

When representing the data set using arrow glyphs most parts of the data set are concealed due to the dense distribution of data. This issue can be addressed by only showing parts of the data set as shown in Figure 4.5. Another option to reduce the visual clutter is adequate filtering. There are two possible filters which can be applied to the displacement field. In regions of similar orientation, the magnitude of the curl is quite small, while, at the same time, the vector magnitude tends to be high. In this project the curl magnitude was used since orientation is the defining property of the domains. As illustrated in Figures 4.5l and 4.5m, filtering by curl magnitude gives clues about the location and size of domains.

In order to visualize the movement and the resizing of domains in the extracted vector field, isosurfaces were rendered using the ray marching approach described in Section 3.2. This allows for a comparison of the different ways of extracting the domains. This includes the domains extracted by the curl magnitude, by the vector magnitude and by the density map. The isosurfaces were colored using the magnitude, the orientation and a surface LIC based on the vector field.

As mentioned before, the vector magnitude inside the vector clusters tends to be higher than in the rest of the vector field. The magnitude can, therefore, be used to extract isosurfaces which correspond to the domains. In Figure 4.6 isosurfaces were extracted based on the vector magnitude. The shrinking and subsequent expansion is clearly visible. However, fine structures of the polarization domains are only approximated. Additionally, this approach assumes that the vector field has a high vector magnitude in areas of high curl magnitude, which is not always the the case in a more general application.

The curl magnitude is the defining property of the domains, therefore, the representation should be more accurate than the one achieved by using the vector magnitude. Isosurfaces

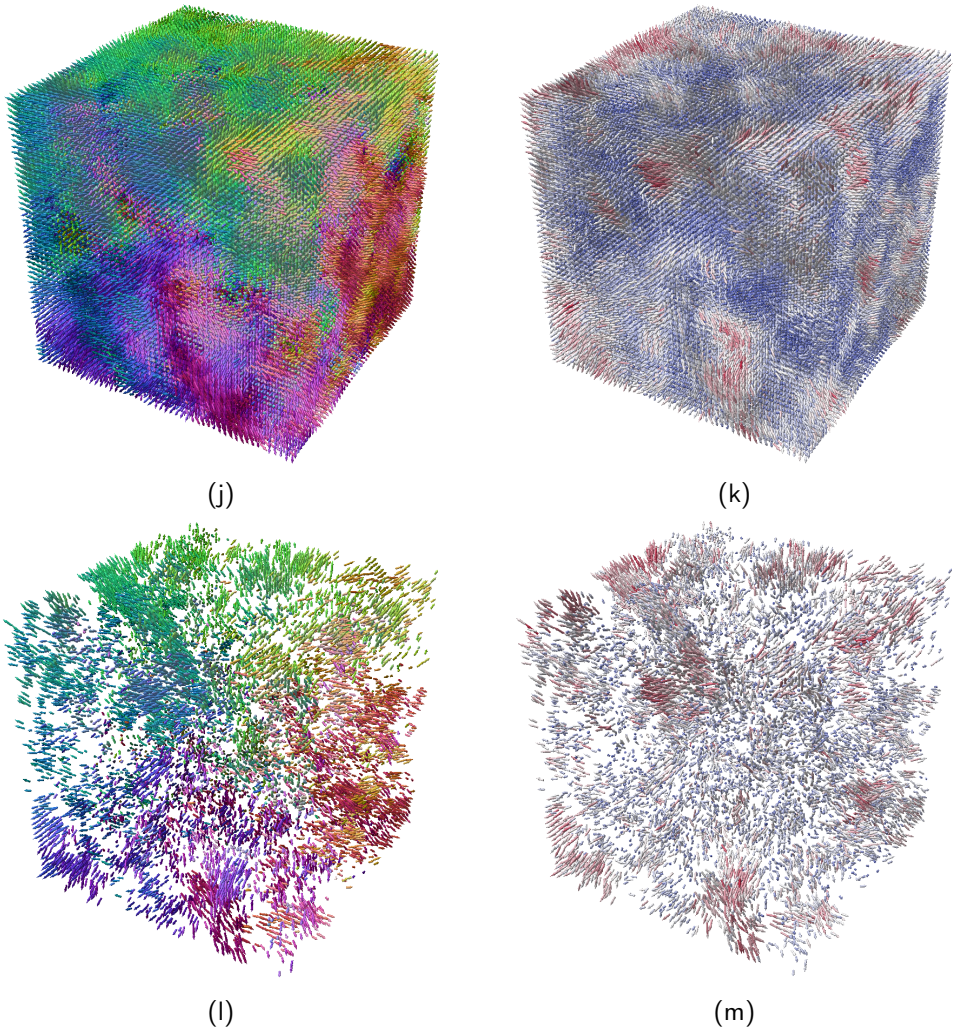


Figure 4.5: Rendering of the displacement field using arrow glyphs. In (j) and (l) the orientation is color-coded. In (k) and (m) the color is determined by magnitude. The vectors inside the clusters appear to have a significantly higher magnitude than the rest of the vector field.

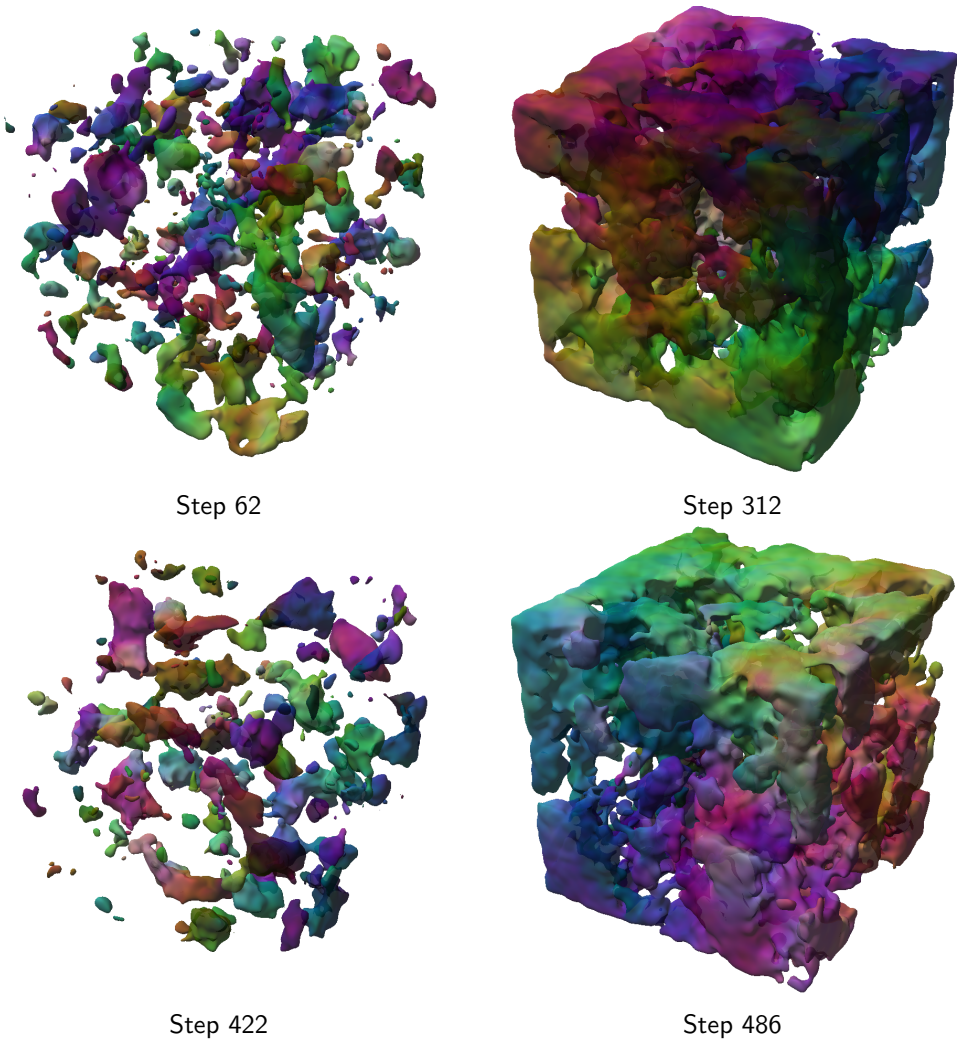


Figure 4.6: Isosurfaces based on vector magnitude. The isosurfaces extracted from the vector magnitude show a development similar to the one illustrated in Figure 4.5. The reason for that similarity is the coherence between the vector magnitude and the vector curl magnitude.

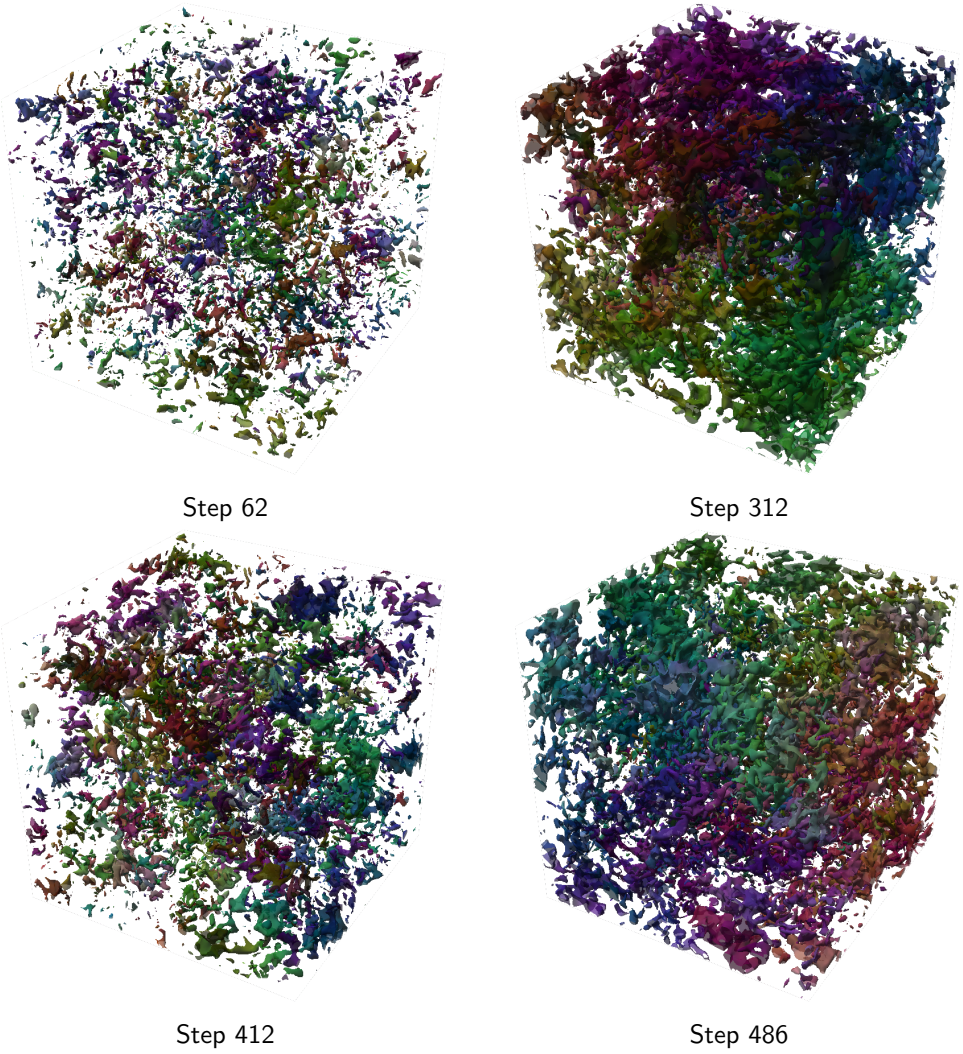
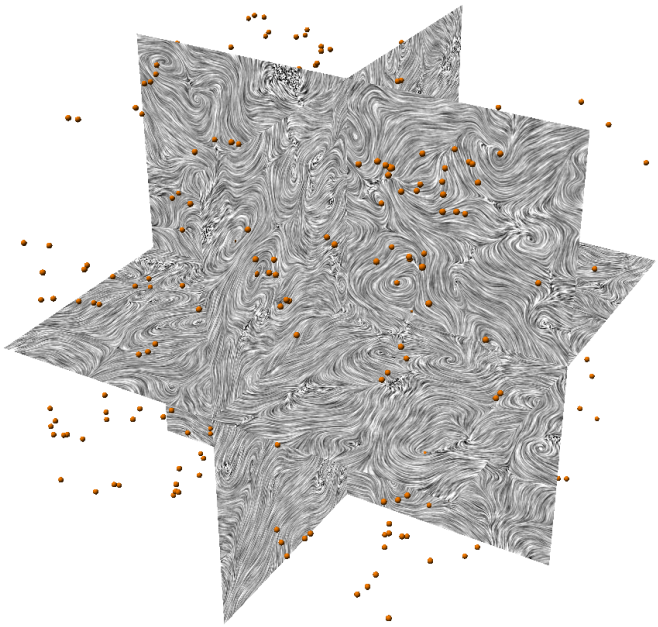
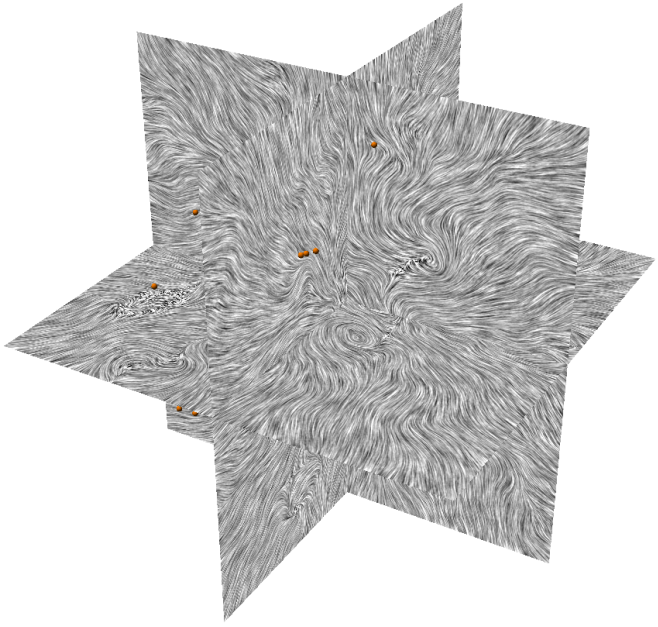


Figure 4.7: Isosurfaces based on curl magnitude. The growth of domains caused by the deformation of the material is clearly visible. However, the irregular isosurfaces make it difficult to distinguish between individual domains.

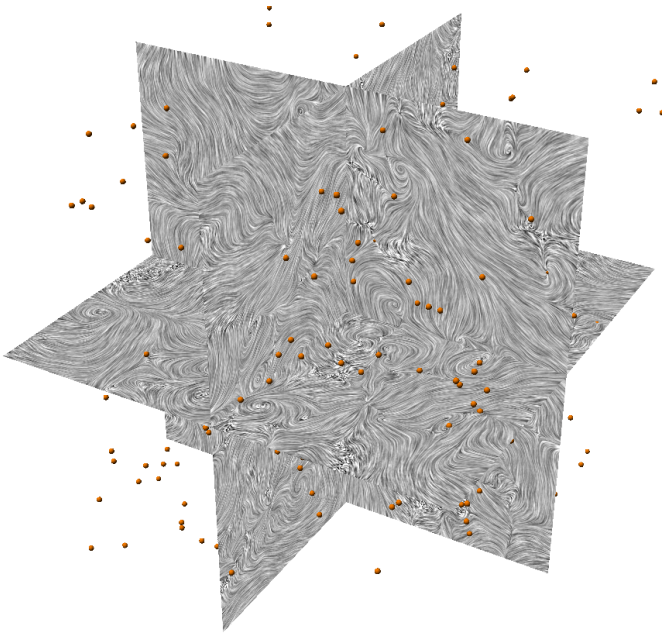


Step 62

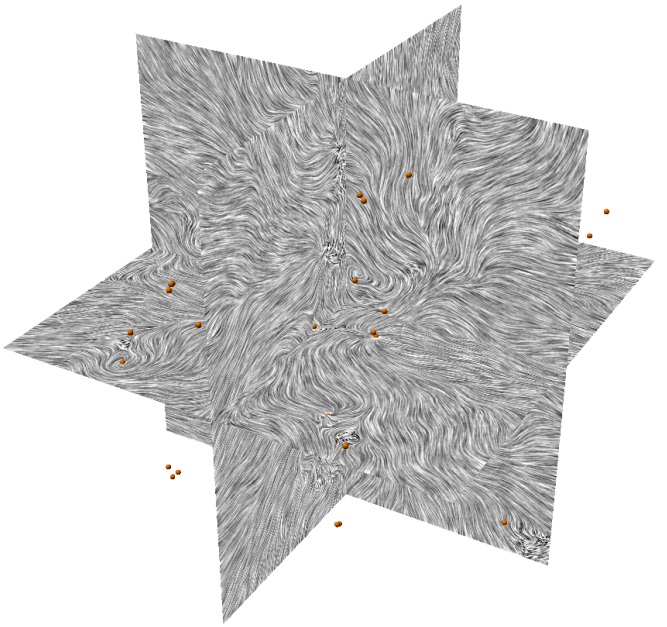


Step 312

Figure 4.8: Rendering of the displacement field using LIC. In the beginning of the time series, the vector orientation is quite irregular, which is illustrates in the high number of evenly distributed singularities (shown in orange). When the material starts deforming, adjacent vectors tend to be oriented similarly and the number of singularities is significantly reduced.



Step 412



Step 486

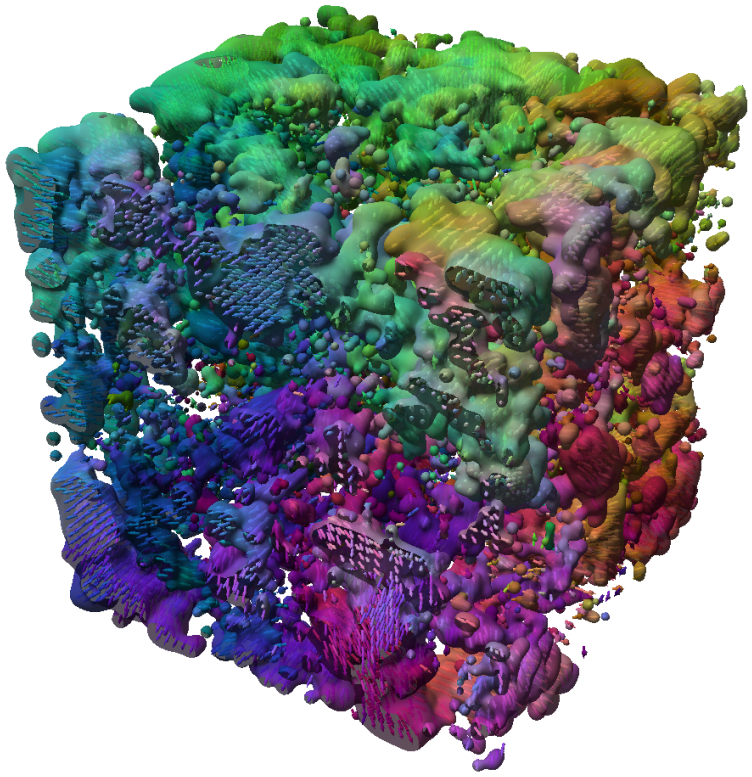


Figure 4.9: Isosurfaces based on a density grid using coloring by vector orientation. Using the density map as a basis for the isosurfaces emphasizes vector clusters without producing too much visual clutter.

were extracted using the curl magnitude, as shown in Figure 4.7. Although the curl magnitude delivers a more accurate representation of the vector field's domains, it produces a rather noisy visualization. When applying this visualization to the time dependent data, it is hard to perceive certain developments in the domains, e.g. a change in size and shape or location.

Aiming for isosurfaces which encircle the vector clusters should give a good impression about the movement and reshaping of domains with a certain level of accuracy while not producing as much visual clutter as when using the curl magnitude alone. Here, the density grid derived from the vector clusters was used as a basis. The resulting isosurfaces encircle the domains and can be combined with the glyph based representation, to give additional clues about the orientation of the vectors inside the clusters. Using the isosurfaces in combination with the arrow glyphs further emphasizes the vector clusters, since it is hard to perceive the actual spatial distribution of the clusters using arrow glyphs alone. Figures 4.9, 4.10 and 4.11 show the three different textures applied to the isosurfaces.

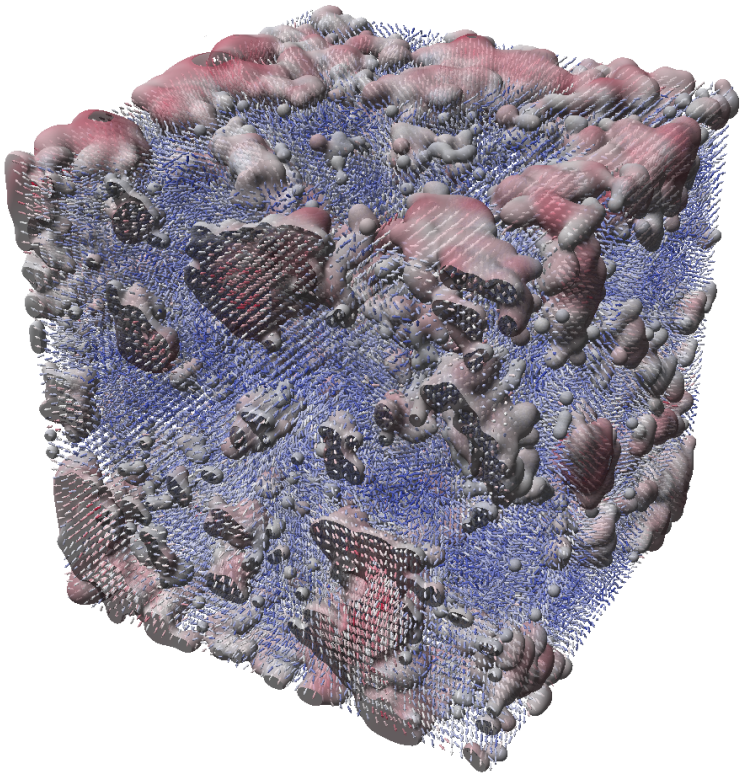


Figure 4.10: Isosurfaces based on a density grid using coloring by vector magnitude. Although the domains have been obtained by filtering according to the curl magnitude, they also roughly correspond to the vector magnitude, which is especially high inside the clusters.

Applying the visualization based on arrow glyphs and the density grid to the time series allows observing the evolution of the location and shape of the domain boundaries over time (see Figure 4.12). The material starts out with a number of evenly distributed domains of relatively small size which are rather unstable. During the first 200 time steps (4 ns), the atoms start getting more mobile. This mobility is mainly caused by the temperature related shrinking of the material. Since the absolute temporal displacement of the titanium atoms is used, this leads to vector clusters of increasing size. The movement in these areas is mainly oriented towards the center. Additionally, these clusters tend to be larger and more stable in off-center regions. After about 8 ns the clusters start to get unstable. This might indicate the beginning of the phase transition, which involves an elongation of the unit cells and, therefore, the material is expected to gain volume. This trend is counteracting the previous domain formation, which presumably was mainly caused by the contraction of the material. Towards the end of the time series at about 8.5 ns, the contraction of the material caused by the decreasing temperature seems to be compensated by the elongation of cells. At this point, only small, unstable domains

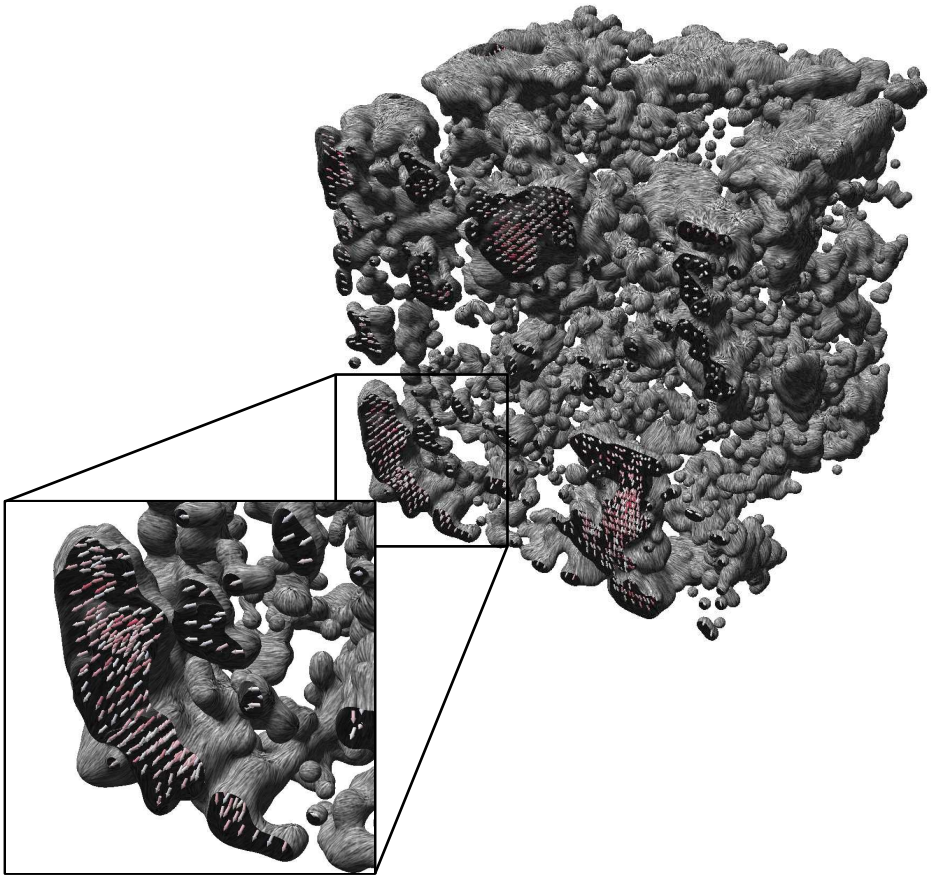


Figure 4.11: Isosurfaces based on a density grid using LIC. In the cutout the texture coordinates used to access the noise texture have been scaled up, in order to obtain a finer LIC texture. Using the LIC illustrates the actual orientation of the vectors inside the clusters, which cannot be achieved by the color coding of the vector orientation alone.

can be observed. Subsequently, the size of the domains increases once again until the end of the simulation is reached. Since these domains only partially comply with the volume dilatation of the material, it can be assumed, that they are influenced by the phase transition and, therefore, can be seen as an indication for the actual polarization domains.

The overall goal of the SciVisContest 2012 was to find a suitable visualization to be able to identify phase transitions and get indications of polarization domains in ferroelectric materials. Neither the data set nor the task description provided a lot of details about how to extract the actual vector field. Nonetheless, using the visualization developed in this student research project, proposals about the correlation between the atom displacement and the phase transition could be made.

This is also brought out in the reviewer comments received for the submission. Here, it is noted that, despite problems with the data a "meaningful visualization of several aspects of the transition" was provided which also suggests "interpretations and criteria for further thinking about the physical phenomenon". Additionally, using the curl magnitude to pre-filter the vector field is pointed out as a crucial step. Furthermore, the use of techniques used in more established research fields, like e.g. fluid dynamics, was recognized as a "very good innovation."

4.4 Future Work

All in all, the visualization implemented in this student research project can be used in various other cases, e.g. when clusters of vectors are to be emphasized or, in a more general case, when certain areas in volumetric data sets are to be emphasized.

The data used in this work represents a very special case with certain features, which cannot be expected in the general case. For instance, in some cases, clusters of dipole moments are actually just chains of dipoles (see e.g. [PBL09]). The method used for extracting domains in this project fails in that case, since it needs domains to be separated by a certain distance. A better way to extract regions with vectors of similar orientation might be to perform actual clustering algorithms instead of filtering out vectors based on vector field features. In this case, features of the clusters like their size or orientation can be used to filter out clusters of little interest. This way visual clutter can be reduced further. If the actual clusters are available, dipole chains could be visualized e.g. by using stream ribbons.

For some of the techniques used, alternative or extended algorithms could have been implemented. An alternative to using arrow glyphs in order to visualize features of the 3D vector field could be the 3D Line Integral Convolution presented in [FW08]. However, using this approach would produce a similar output where domains appear as clusters of objects and it would still be hard to perceive the actual spatial distribution of the domains. The extracted isosurface could have been further improved by adding more visual clues to the final image. There are several methods to add global illumination to the image while keeping interactive frame rates. One possibility is the Screen Space Ambient Occlusion developed by Crytek [Mit07]. A way of simulating ambient occlusion

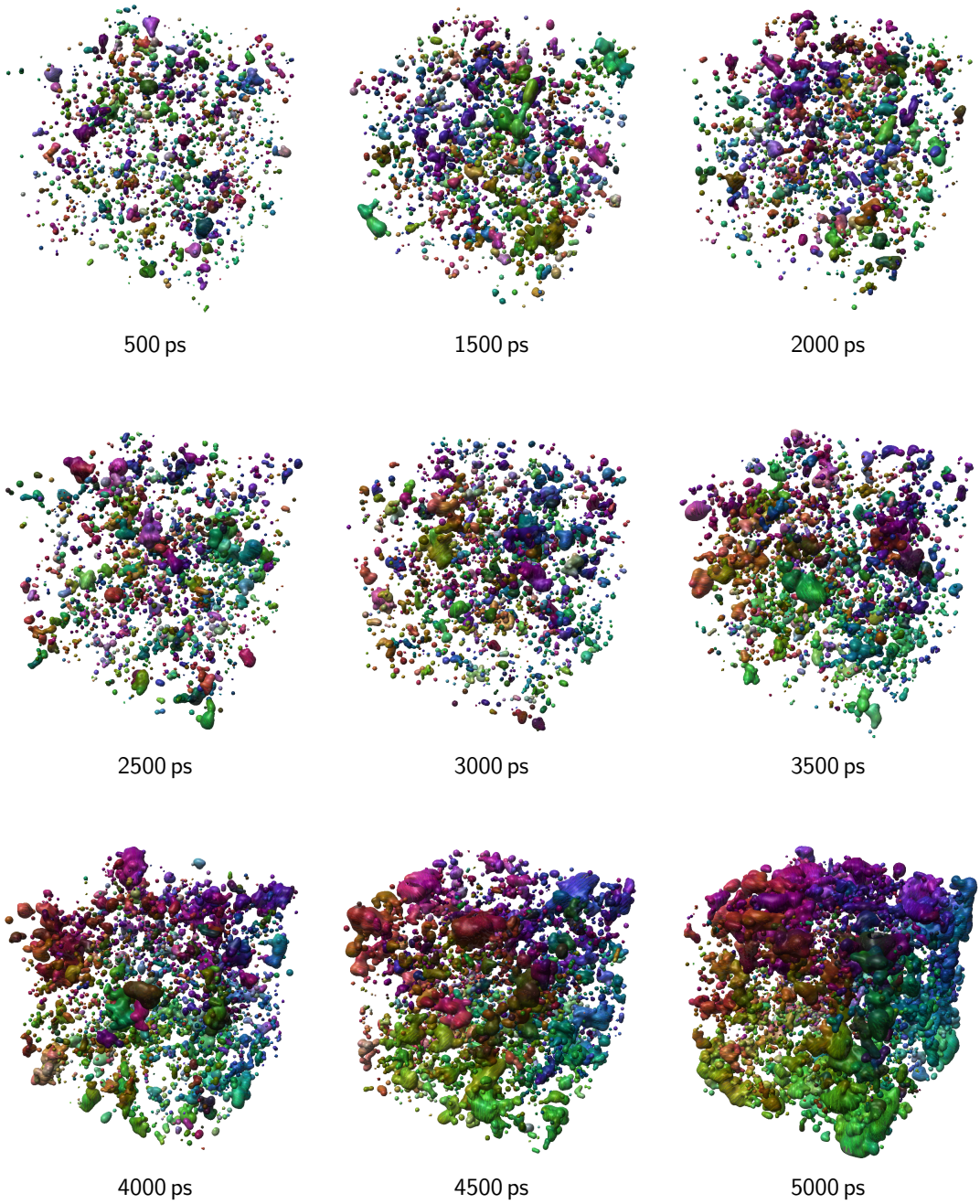
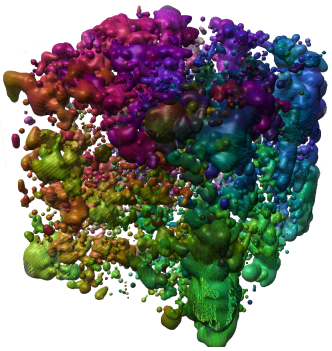
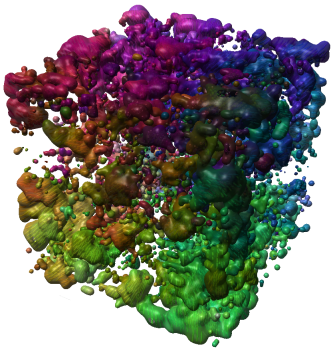


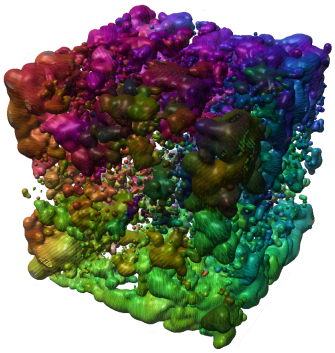
Figure 4.12: Rendering of the time series using isosurfaces based on the density map. Here, semitransparent isosurfaces were drawn in combination with a glyph based representation of the vectors. The shrinking and subsequent growth of the clusters caused by the deformation of the material are clearly visible.



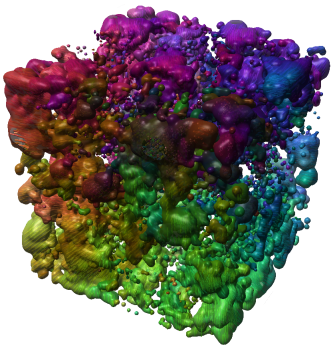
5500 ps



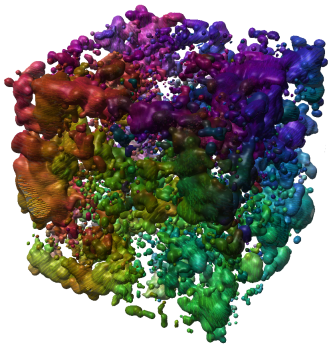
6000 ps



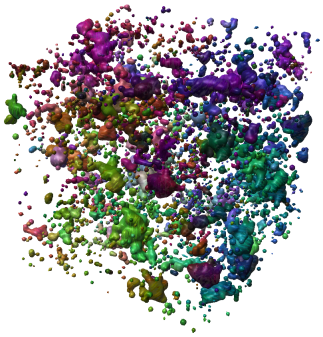
6500 ps



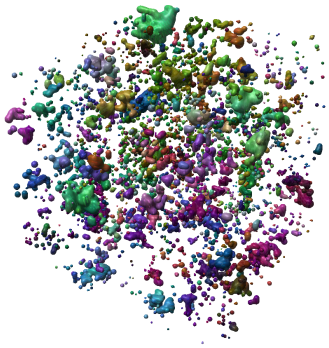
7000 ps



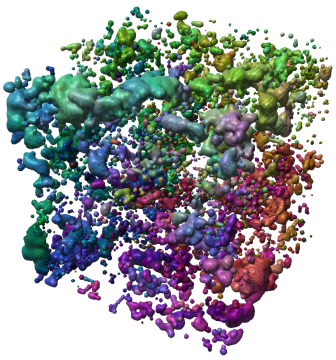
7500 ps



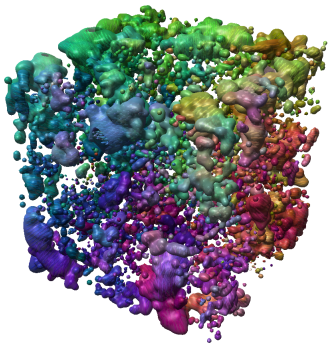
8000 ps



8500 ps



9000 ps



9500 ps

when using direct volume rendering was presented by [HLY10]. Both approaches could help further emphasizing the shape of the isosurfaces representing the domains. Using the current implementation of the calculation of the topological degree, in order to find critical points, it is not possible to maintain interactive frame rates. Implementing an actual bisection method would have made the implementation much faster. It could also have profited from using the GPU in order to parallelize the algorithm.

5 Summary

The goal of this project was to use the data provided by the SciVis Contest 2012 to gain insight into the development of domains and phase transitions in ferroelectric materials. This should help material scientists to detect phase transitions and investigate the details of domain structures and their development over time. However, the task description provided for the contest was followed only loosely. The data provided contained atom types and the respective atom positions, which, exhibited high oscillation. Since the oscillation was likely to superpose the local displacement defining the polarization, the atom position have been averaged over a sliding time window.

Since the data provided did not contain exact electrostatic dipoles, a suitable approximation had to be found. The strongest indicator for the polarization in the tetragonal phase of barium titanate is the displacement of the titanium atom in the cell center. Several methods of approximating the polarization based on the displacement of the titanium atom relative to the cell center have been tested, however, none of them showed a noticeable development of domains in shape or size. Therefore, the absolute temporal displacement of the titanium atoms was used, although this approximation is likely to be superposed by the contraction of the material due to the temperature decay.

In this project, domains were defined as clusters of vectors exhibiting a similar orientation. One way of measuring the local change of orientation of the vector field is to take the curl magnitude of the normalized vector field into account. Here, domains are defined, were this value is rather low. Domains can then be obtained by filtering out vectors, which are located in areas of high curl magnitude.

The pervoskite structure of the lattice as well as the movement of atoms can be visualized using glyph based representations. The unit cells were rendered using sticks and spheres, whereas the atom movement was illustrated using arrow glyphs, with different coloring modes. This highlighted the deformation of the material, which was caused by both the temperature and the phase transition. Domains were visualized using a combination of arrow glyphs and isosurfaces based on different scalar fields. The isosurfaces based on the vector magnitude represented the domains only approximately, whereas the more accurate representation based on the curl magnitude produced rather noisy outputs. Thus, in the end, a density grid based on the vectors, which are forming the clusters, was used in order to emphasize the domains.

The visualization of the atom movement as well as the domains allowed for the interactive exploration of some aspects of the phase transition in barium titanate. Additionally, the visualization methods described can also be applied to more general cases, e.g. where clusters of similarly oriented vectors have to be highlighted.

Bibliography

[BC11] M. Bailey, S. Cunningham. Graphics Shaders: Theory and Practice. CRC Press, 2 edition, 2011.

[Bli77] J. F. Blinn. Models of light reflection for computer synthesized pictures. SIGGRAPH Comput. Graph., 11(2):192–198, 1977. doi:10.1145/965141.563893.

[BSH97] H. Battke, D. Stalling, H.-C. Hege. Fast line integral convolution for arbitrary surfaces in 3D. In H.-C. Hege, K. Polthier, editors, Visualization and mathematics, pp. 181–ff. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[CL93] B. Cabral, L. C. Leedom. Imaging vector fields using line integral convolution. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, SIGGRAPH '93, pp. 263–270. ACM, New York, NY, USA, 1993. doi:10.1145/166117.166151.

[CR08] W. D. Callister, D. G. Rethwisch. Fundamentals of materials science and engineering: an integrated approach. Wiley, Hoboken, N.J., 3. ed., international student version edition, 2008.

[For94] L. K. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In Proceedings of the conference on Visualization '94, VIS '94, pp. 240–247. IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[FW08] M. Falk, D. Weiskopf. Output-Sensitive 3D Line Integral Convolution. Visualization and Computer Graphics, IEEE Transactions on, 14(4):820 – 834, 2008.

[GBM+12] S. Grottel, P. Beck, C. Müller, G. Reina, J. Roth, H.-R. Trebin, T. Ertl. Visualization of Electrostatic Dipoles in Molecular Dynamics of Metal Oxides. In IEEE Trans. Vis. Comp. Graph. 2012.

[Gho02] P. Ghosez. Microscopic Properties of Ferroelectric Oxides from First-principles: Selected Topics. Number Teil 1 in Troisième cycle de la physique en Suisse romande. Troisième cycle de la physique en Suisse romande, 2002. URL http://www.phythema.ulg.ac.be/Download/Files/Cours_Ferro.Ghosez.pdf.

[Gre92] J. M. Greene. Locating three-dimensional roots by a bisection method. Journal of Computational Physics, 98(1):178–178, 1992.

[GRE12] S. Grottel, G. Reina, T. Ertl. MegaMol™: A Visualization Middleware for Point-based Molecular Data Sets, 2012. http://www.vis.uni-stuttgart.de/megamol.

[HLY10] F. Hernell, P. Ljung, A. Ynnerman. Local Ambient Occlusion in Direct Volume Rendering. IEEE Trans. Vis. Comput. Graph., 16(4):548–559, 2010. URL http://dblp.uni-trier.de/db/journals/tvcg/tvcg16.html#HernellLY10.

[KE04] T. Klein, T. Ertl. Illustrating Magnetic Field Lines using a Discrete Particle Model. In VMV'04, pp. 387–394. 2004.

[KLBC93] G. H. Kwei, A. C. Lawson, S. J. L. Billinge, S. W. Cheong. Structures of the ferroelectric phases of barium titanate. The Journal of Physical Chemistry, 97(10):2368–2377, 1993.

[KSES12] M. Krone, J. E. Stone, T. Ertl, K. Schulten. Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories. In EuroVis 2012 Short Papers, volume 1. 2012.

[KW03] J. Krüger, R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In Proceedings IEEE Visualization 2003. 2003.

[Lev90] M. Levoy. Efficient ray tracing of volume data. ACM Trans. Graph., 9(3):245–261, 1990. doi:10.1145/78964.78965.

[LVW95] W. C. de Leeuw, J. J. Van Wijk. Enhanced Spot Noise for Vector Field Visualization. In Proceedings of the 6th conference on Visualization '95, VIS '95, pp. 233–. IEEE Computer Society, Washington, DC, USA, 1995.

[Mit07] M. Mittring. Finding next gen: CryEngine 2. In ACM SIGGRAPH 2007 courses, SIGGRAPH '07, pp. 97–121. ACM, New York, NY, USA, 2007. doi:10.1145/1281500.1281671.

[PBL09] M. Pasciak, S. E. Boulfelfel, S. Leoni. Size and Time Rescaling at the Paraelectric to Ferroelectric Phase Transition in BaTiO3, 2009. http://arxiv.org/abs/0901.4560.

[PD84] T. Porter, T. Duff. Compositing digital images. SIGGRAPH Comput. Graph., 18(3):253–259, 1984. doi:10.1145/964965.808606.

[Pho75] B. T. Phong. Illumination for computer generated pictures. Commun. ACM, 18(6):311–317, 1975. doi:10.1145/360825.360839.

[RDL+07] K. Rabe, M. Dawber, C. Lichtensteiger, C. Ahn, J.-M. Triscone. Modern Physics of Ferroelectrics:Essential Background. In Physics of Ferroelectrics, volume 105 of Topics in Applied Physics, pp. 1–30. Springer Berlin / Heidelberg, 2007.

[RE05] G. Reina, T. Ertl. Hardware Accelerated Glyphs for Mono- and Dipoles in Molecular Dynamics Visualization. In EuroVis05: IEEE Symposium on Visualization, pp. 177–182. 2005.

[RSHTE99] C. Rezk-Salama, P. Hastreiter, C. Teitzel, T. Ertl. Interactive exploration of volume line integral convolution based on 3D-texture mapping. In Proceedings of the conference on Visualization '99: celebrating ten years, VIS '99, pp. 233–240. IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[SH95] D. Stalling, H.-C. Hege. Fast and resolution independent line integral convolution. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, SIGGRAPH '95, pp. 249–256. ACM, New York, NY, USA, 1995. doi:10.1145/218380.218448.

[SKBS12] K. Scharnowski, M. Krone, P. Beck, F. Sadlo. Visualization of Polarization Domains in Barium Titanate, 2012. Submission to the IEEE SciVis Contest.

[Uch05] K. Uchino. Ferroelectric Devices. CRC Press, 2 edition, 2005.

[Val21] J. Valasek. Piezo-Electric and Allied Phenomena in Rochelle Salt. Physical Review, 17:475–481, 1921. doi:10.1103/PhysRev.17.475.

[Wad00] V. Wadhawan. Introduction to Ferroic Materials. Gordon & Breach, 2000.

[WDH74] O. H. Wyatt, D. Dew-Hughes. Metals , ceramics and polymers: an introduction to the structure and properties of engineering materials. Cambridge Univ. Pr., London, 1974.

[Wij91] J. J. van Wijk. Spot noise texture synthesis for data visualization. SIGGRAPH Comput. Graph., 25(4):309–318, 1991. doi:10.1145/127719.122751.

All links were last followed on 10/06/2012.

A Appendix

The following section contains the task description of the IEEE visualization contest 2012 as found on the official contest website [1].

A.1 Task description SciVis Contest 2012

The overall goal is, as usual, a visualization that helps computational material scientists to better understand exactly what is going on. In this case, this means to understand how, when, and where the phase transitions occur.

We suggest to tackle the following visualization tasks, but you are welcome, of course, to make additional visualizations as you deem fit.

Represent the dataset as a vector field. File A and B contain atomic positions at time t=0 ps (pico seconds) and time t=2 ps. With respect to the initial distribution, atoms tend to be in a less symmetrical position, which promotes the formation of local polarization domains. The magnitude of the displacement can be obtained by evaluating the displacement vectors for each atom. Some atoms would not displace much, some other are more sensitive. Filter out the less mobile, concentrate on the larger displacements. It may be appropriate to multiply the displacements by a scalar, for better visibility. Locally, the orientation of some vectors will tend to be similar. The changes are a function of the portion of volume considered. Try to make sense of the formation of domains with a simple representation technique. The different directions of atom displacemente (Ti) can be organized into "domains". That means, that some portion of the volume can be characterized by a similar direction of displacements. This could be visualized by arrows, particles, field lines, using something more complicated like LIC.

Polarization change may be better captured by the definition of domain boundaries. The vector field defined under 1. can be analyzed by means of standard vector field operations. A change of polarization in a particular volume corresponds to a sharp rotation of displacement vectors. What mathematical descriptor can be good for this matter of fact ? As a working hypothesis, the scientists suggest looking for zero lines in the dataset. Zero may represent vanishing polarization (zero or very short displacement), or vanishing of some derived scalar or vector quantity.

From the boundaries obtained under 2, try to construct a suitable visualization. Since many combination of displacement directions and magnitudes can in principle be expected, a number of features may be expected from the vector field. A number of singularities can

[1] http://sciviscontest.visweek.org/2012/VisContest/Tasks.html

be anticipated. A limiting case may be when a number of vectors will point away from the same point, or merge into a single point. Many intermediate cases can be imagined. Find a good method to make the "nodes" in the vector field visible. A Line Integral Convolution approach might be a good tool.

Phase transition: From the experience collected above, find a suitable visualization for the change of displacement vectors as a function of time. Is there an evolution in the trajectory, which could suggest a phase transition ? Is it possible to capture the process monitoring the evolution of a single function, connected with the change of the atomic displacement? Is there an obvious (auto)correlation function ?

Evolution of shape and boundaries: How can the evolution of the clustering be described along the transitions ? Maybe a volumetric approach can be useful to help understanding the complex shapes, which are forming along the process. Can a suitable scalar value be associated with the vector to help understanding what is going on inside the clusters, giving some idea about the rapidity of the displacements taking place inside ? How about cluster boundaries, are they at all stable?

Basic Research Question: Here it is where the visualization is asked to provide insights into the physical process. How is the change from the initial cluster distribution towards a different scenario taking place? Is there a means to tell whether it is a local change, which is more and more spreading over the whole dataset, or is it more collective ? "Local" corresponds to what in the jargon of phase transition is called nucleation. Is any of the previous functions useful in capturing the changes in details, is the change initially just a nucleus, a handful of atoms ? Which visualization is more suitable to make sense of this step ?

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Katrin Scharnowski)