2008

Space-time multiresolution approach to atomistic visualization

Dipesh Bhattarai
Louisiana State University and Agricultural and Mechanical College

SPACE-TIME MULTIRESOLUTION APPROACH TO ATOMISTIC VISUALIZATION

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Dipesh Bhattarai
B.E., Tribhuvan University, 1999
M.S., Alabama A&M University, 2003
December 2008

To my parents

and

my beloved wife, Shraddha.

Acknowledgments

Foremost of all, I would like to thank my major advisor Dr. Bijaya B. Karki for guiding me through various stages of uncertainties and doubts. When I did not know what I was doing, he was the one who supported me, believed in me and guided me. I would also like to thank Dr. S. Sitharama Iyengar, Dr. Jianhua Chen and Dr.Tevfik Kosar for agreeing to be in my dissertation committee and providing valuable feedback. I would like to thank Dr. Boryung Ju, the dean's representative, for her interest and constantly asking questions about the status of the work.

I would like to acknowledge Dr. Ashok Verma and Gaurav Khanduja for providing a stimulating intellectual environment in our laboratory and acting as a sounding board when I was not sure about worth of my own thoughts. I would also like to express my thanks to Ms. Vera Watkins and Ms. Amy Fowler for taking care of the administrative details so that I could focus on my work. I would also like to express my gratitude to the whole Department of Computer Science for providing such an excellent environment for intellectual pursuit.

I would also like to acknowledge the support from National Science Foundation and NASA for this research.

I would not be here with this work if it was not for my family. Their excitement was always source of my energy that kept on motivating me through these four and half years. Last but not the least, I would like to express my sincere gratitude to my beloved wife, Shraddha, for pushing me hard so that I could finish this dissertation. She would not settle for less than the finished work. If it was not for her, I would not have been able to finish. This work belongs to her as much as it does to me.

Table of Contents

List of Tables

List of Figures

Abstract

Time-varying three-dimensional positional atomistic data are rich in spatial and temporal information. The problem is to understand them. This work offers multiple approaches that enable such understanding. An interactive atomistic visualization system is developed integrating complex analyses with visualization to present the data on space-time multiresolution basis facilitating the information extraction and generate understanding. This work also shows the usefulness of such an integrated approach.

The information obtained from the analyses represents the system at multiple length and time scales. Radial distribution function (RDF) provides a complete average spatial map of the distribution of the atoms in the system which is probed to explore the system at different length scales. Coordination environments and cluster structures are visualized to look at the short range structures. Rings are visualized to understand the medium range structure. Displacement data and covariance matrices are visualized to understand the dynamical behaviors. Combinations of rendering techniques including animation, color map, sphere, polygonal and ellipsoid representations, pathlines and glyphs are used during the visualization process. The three-dimensional atomic configurations are reproduced accurately during rendering because of their physical significance while attributes such as coordination number, coordination stability and atomic species lack direct physical relevance and provide additional flexibilities in rendering.

The performance results show interactive frame rates are achievable for systems consisting upto a thousand atoms. Such systems are typical of the systems simulated using first principles molecular dynamics simulations. The effectiveness and the usefulness of this work are justified for complex material systems using silicate and oxide liquids for visual analyses. The exploratory approach taken here has not been reported anywhere else before.

The major contributions of this works are:

1. A new approach to the atomistic visualization advocating a formal integration of data analyses into the visualization system to improve the effectiveness and also present an implementation of the exploratory atomistic visualization system with integrated spatio-temporal analytical techniques.

2. The modeling of coordination environments, stability of the coordination environments, clusters, ring structures and diffusion for individual atoms.
3. The use of the visualization system for visual analysis of various liquid mineral systems of geophysical relevance.

Chapter 1 Introduction

Atomistic visualization is a technique to visually understand an atomistic system. The atomistic system consists of atoms of different species and the visualization depicts the relationships among the atoms in the system. The visualization usually assumes there are no relationships among the atoms because either the atoms actually do not interact with each other, such as in case of uncorrelated systems like gaseous systems, or there is not enough information available about the interatomic interactions. In the first case, the assumption does not detract the usability of the visualization because the visualizations are rarely used to understand the uncorrelated systems. On the other hand, if the atoms are correlated with each other, the assumption detract from the usefulness of the visualization. For example, visualization is routinely used for the systems such as crystals, biomolecules, and liquids where the atoms are correlated with each other. The assumption limits the level of understanding the visualization can generate. Also, such assumption is fundamentally problematic because the data points closer in space and time are highly correlated and are dependent with each other [1]. An atomistic dataset is a spatial dataset and contains highly correlated data points even though the correlation is not explicitly provided. Such correlations among the atoms in the system have to be first quantified for a meaningful visualization.

We model the time-varying atomistic dataset as a random point process to analyze it and develop an atomistic visualization system. Previous works in the atomistic visualization assumed the atomistic systems to be deterministic. The assumption prevented the use of results from the random point process theory for data modeling. In case of the time varying atomistic systems, the states of the system at different times were assumed independent and uncorrelated to each other. Such assumptions made the dependencies and correlations among multiple snapshots difficult to understand. This work approaches the problem from a completely new perspective and considers the complete dataset during the analyses. Such approach makes the visualization of the correlations easy. The theory of random processes provides many useful tools for the analyses of such time-varying datasets.

1.1 Background

The visualization is essentially a human activity [2] of converting data into graphical forms such that the hidden information is visible. In other words, visualization is a process of making invisible things visible. The conversion process may occur wherever something can be imprinted whether it be a piece of paper or a computer screen. In other words, "visualization is an activity which a human being engages in, and that it is a cognitive activity" [2]. These definitions do not specify how to visualize or where to visualize except that the humans use it as a tool in better understanding the data.

The numbers don't lie. But they also don't tell everything. A scientific dataset contains hidden patterns that require domain specific background knowledge and analytical tools to unlock them. Graphical representations make the hidden patterns visible, apparent and obvious with little mental effort. The generation of efficient graphics [3] to transfer such information with little "mental cost" [4] is the objective of the visualization research. The problem is non-trivial and becomes more complicated because multiple factors - visual perception, domain specific knowledge, data analysis and transformation - are involved during the generation and understanding an image. Many of these factors are not completely understood yet and require more extensive understanding of multiple overlapping scientific disciplines. For example, the problem of visual perception - the problem of how humans perceive the world around it and understand - is still unsolved. Also, a two-dimensional plane is generally used for the visualization whereas the data being visualized can be ordinal, time-series, multidimensional, or multivariate. The optimal mapping of such diverse sets of data into two-dimensional plane is also unknown. These factors have far reaching implications in multiple scientific disciplines from brain research and psychology to computer graphics and display technology.

The graphical representations are effective in abstracting information from the data [5] and in sparking the imagination [6]. Despite such effectiveness, it becomes impractical to create separate static graphical representations for all the relevant information especially for complex datasets. There might be too much information to see. Instead of such static representations, the visual exploration provides a better framework by delegating the responsibility of finding the appropriate information to the user. The exploratory visualization also compensates for the differential between the rate of growth of the available data and the rate of improvements in the

techniques in data analyses and information mining. The data have become problem rather than solution. The visualization helps to look at the data from multiple perspectives.

T	Cp	T	Cp	T	Cp	T	Cp
18.28		70.26	4.635872	145.03	22.45134	254.87	43.55126
21.12	0.04184	75.6	5.66932	154.92	24.73162	259.61	44.7437
23.05	0.092048	81.29	7.045856	160.29	26.21694	266.88	45.48845
25.06	0.142256	85.52	7.828264	173.53	28.71898	274.04	46.58466
27.19	0.192464	91.29	9.154592	183.37	31.02018	279.19	47.07418
29.66	0.221752	97.04	10.39724	194.73	33.35903	286.69	47.75618
32.41	0.309616	102.83	11.65662	208.21	35.95311	287.69	47.57208
34.92	0.430952	108.58	13.20052	219.27	38.05766	293.42	47.7771
37.75	0.560656	114.13	14.31346	236.77	41.24169	298.81	48.08253
41.37	0.681992	119.2	15.76113	237.71	41.97389	300.14	47.73944
45.43	1.037632	124.4	16.91591	238.63	40.70195	304.06	47.79802
54.12	1.8828	128.41	18.40123	238.69	41.34629	304.22	47.84404
57.42	2.317936	129.79	18.0707	246.14	42.30442		
60.99	2.765624	136.05	20.17943	248.39	42.66006		
65.42	3.552216	140.35	21.21288	253.63	44.2709		

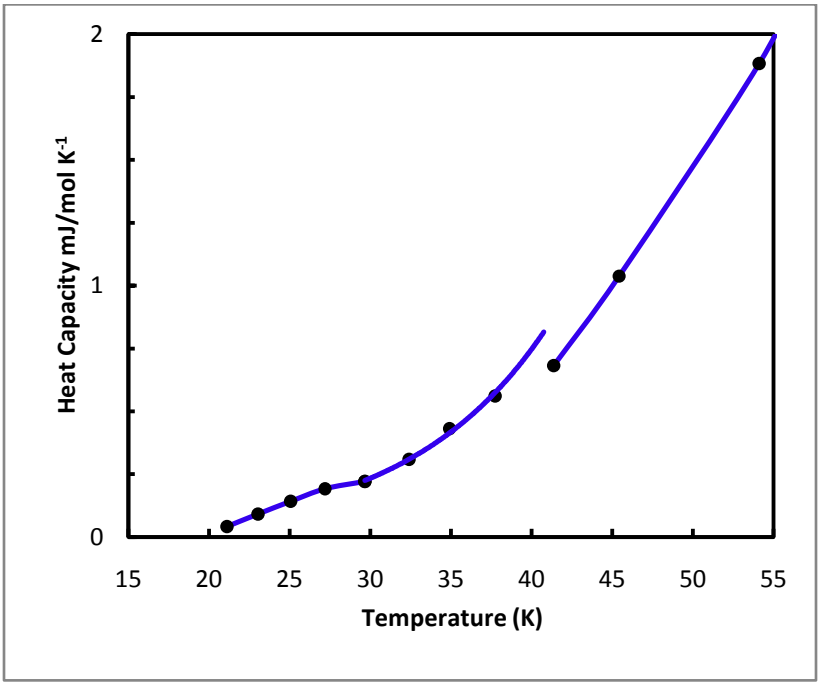

Figure 1-1: Relationship between heat capacities and temperature

A frequently asked question is: Why visualize? What is so special about vision? One major reason is we have become more facile in controlling visual responses by using visual metaphors and more information can be transferred visually to brain faster than by other sensory means, such as sound or touch. Sonification, tangible visualization and haptic rendering are also three emerging research areas that utilize sound and touch to transfer information to brain. Bertin

[3] shows that we can make our brain process data the way we want by using language of the symbols. For example, verbal language is a set of auditory symbols, written language is a set of visible codes, and these languages have been used to communicate. Similarly visual objects can also be used as codes to codify separate kind of information by attaching meanings to the objects. Bertin explains properties of visual symbols, such as size, length, texture and color, that can be used to represent properties of a data on a plane of a paper. He also recognizes the importance of computers in such visualization. Figure 1-1 shows one example depicting the importance of graphical representation of data. The table contains the specific heat capacities of MgB_2 (Magnesium Diboride) [7]. At the time of publication, the authors were unable to see the discontinuities in the specific heat characteristics because they did not represent the data graphically. They were unable to grasp the complete relationships between the two variables. This resulted in a delay in the discovery of superconductivity in Magnesium Diboride for 47 years. This is nicely presented in a talk by Warren Picket [8]. A simple chart at the bottom of the Figure 1-1 clearly shows the discontinuities in the relationship that indicates the sharp changes in conductivity at a specific temperature.

Tufte [9-12] presents beautiful graphical representations of different types of data exemplifying the efficient usages of graphics. Such ingenious usages of two dimensional plane of a paper in presenting data of increasing complexity can also be regarded as visualization in modern sense. There are two commonly recognized categories of computer assisted visualization. One is information visualization and the other is scientific visualization. Information visualization is applicable to abstract data where physicality of the underlying system is irrelevant. For example, in visualizing the fluctuations of stock prices, physical representations of stocks or prices are not important. There is no need to display the pictures of dollar bills to represent stock prices to get the information across. On the contrary, physicality of the system being visualized is relevant and important in scientific visualization. For example, when an MRI data of a human head is visualized, correct reproductions of the shape of the head and the skeletons are important. Similarly when an atomic structure is visualized, the physical representation and three dimensional arrangements of the constituent atoms have to be correct. From this perspective, scientific visualization is more restrictive in data representation than information visualization. Despite the differences, there are some overlaps between the information visualization and the scientific visualization. While the visual representations in the

scientific visualization must preserve the physical representation and the spatiality of a given system, there are other information that do not have direct physical significance. Such information can be visualized using the techniques from the information visualization. This implies the information visualization and the scientific visualization are applicable together and techniques from information visualization can be additionally used to enrich the information content of the scientific visualization. Such information enrichment is utilized throughout this work.

Scientific visualization is widely used in essentially all areas of mathematics, science, engineering and technology including materials modeling and simulation. It has become indispensible in many areas such as mathematics and nanomaterials. Here, we have to recognize that the visualization is not a hard science. The results from visualizations are domain specific and they must strive to fulfill the domain specific needs. The target application area of this research work is computational materials modeling and simulation.

1.2 Trends in Computational Materials Science and Challenges for Visualization

The current trends in computational materials science demand visualization along three major directions. First, the large-scale simulations produce large datasets and the challenge is to render those massive datasets. Second, a wide range of properties and processes including structure, elasticity, rheology and melting of materials are simulated, and the challenge is to fulfill domain-specific requirements within a single visualization framework. Third, computationally intensive simulations using quantum mechanical methods produce precise data, and the challenge is to extract information and visualize them. The first problem warrants that the visualization system be able to handle large datasets containing millions of atoms and efficiently render them into corresponding graphical representations. The second problem warrants that the visualization system be able to correlate measured quantities, provide features for interactive manipulation of various parameters, and allow the exploration of the data. The third problem warrants that analytical algorithms be integrated into the visualization framework to enable information extraction and exploratory visualization of the dataset.

An effective scheme is developed in this work to support interactive atomistic visualization of the dataset from first principle molecular dynamics atomistic simulations at space-time multiresolution. The structural properties of the minerals vary according to spatial

and temporal intervals used and the approach taken in this work enables the exploration of the properties at different spatial and temporal intervals. This work tries to fulfill the demands posed by the third class of problems. The effectiveness of the scheme is shown in a series of visual analyses performed to understand and compare the properties of the various minerals. During the course of this work, several problems were encountered and some of the problems were solved and implemented into the system and the remaining problems require further work.

1.3 Target Dataset

This work mainly grew out of the frustration due to the problem in analyzing and understanding the scattered time-varying point data generated by the molecular dynamics simulations. The simulations are very good at solving a given set of equations and generating massive amount of outputs but they are not very good in helping to understand the output. The datasets here are also produced by one of such simulations, first principle molecular dynamics (FPMD) simulations. The FPMD simulations are based on the interatomic interaction potentials derived within a quantum mechanical framework. Even though we use the dataset from such simulations as examples, the usefulness of the visualization approach is not restricted to only these kinds of datasets and is applicable in general to the other atomistic datasets also. With some modifications, this visualization approach can also be used to visualize and explore a general point dataset as well. The dataset considered in this work represents a computational solution to an N-body problem where the bodies interact with each other in a potential field generated by their atomic and electronic configuration. The general N-body problem is known to be analytically intractable when N is larger than two. The atomistic systems regularly contain hundreds of atoms.

The FPMD simulations accurately represent the micro-phenomena and the generate précis data. A sample dataset (Figure 1-2) contains complete information about the system and extraction of detailed quantitative and qualitative information is important in understanding the system. Such extra information is combined with the dataset and rendered on the fly instead of simply rendering the given data. Such visualization enables better understanding of the system at multi space and time details. In other words, this approach facilitates the extraction of global and local spatial-temporal details for information about bonding, pair correlation, coordination,

structural units, clustering, structural stabilities, defects, diffusion and other dynamical processes. Aggregation of these information helps in understanding higher level phenomena.

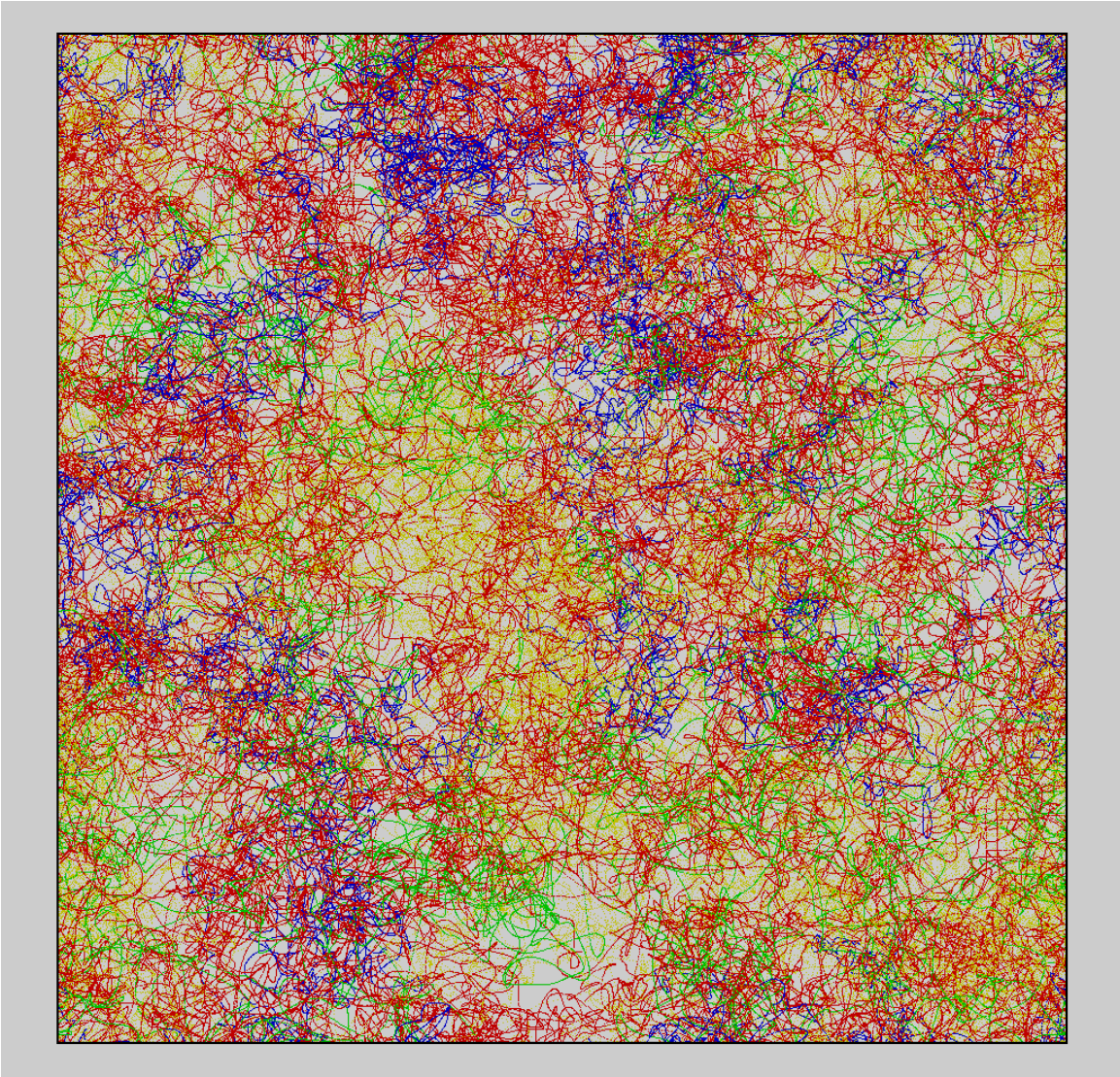

Figure 1-2: A complete atomistic dataset with colors indicating the types of the atoms.

The set of trajectories of all the atoms contains complete information about the system dynamics and structure, but the extraction of information that from the data is non-trivial. The data has to be looked upon from various perspectives employing multitudes of exploratory techniques to understand what the dataset is trying to tell [13]. The problem of atomistic visualization has so far been regarded as the problem of rendering larger number of molecules and bonds. Such assumption is valid only if the dataset itself contains information about the structures in the system. When the dataset does not contain structural information, such as the

datasets considered in this work, prevailing approach to the atomistic visualization becomes ineffective. The visualization of the dataset containing descriptions of the structures leads to molecular visualization that works at a higher level in analytical hierarchy than the atomistic visualization. An atomistic dataset does not contain information about how various atoms are connected with each other to form structures. The dataset is analyzed to find such connectivity relationships and the scheme followed in this work integrates spatio-temporal analysis and rendering tools together to enable such extraction. The information obtained after the analyses is further utilized to enable higher level structural analyses. The visualization system is a proof of concept for a formal integration of the analytical algorithms into an interactive atomistic visualization framework to improve the effectiveness of the visualization system as a whole. Such systematic approach has not been reported previously although a large number of atomistic visualization works are found in the literature (see Chapter 2).

1.4 Problem Statement

An atomistic dataset representing disordered system such as liquid is a spatio-temporal dataset generated as a result of the underlying random point process guided by quantum mechanical interactions. Such assumption is not very implausible when we look at the complexity of the quantum interactions that go into determining the positions of individual atoms in the system. Such level of complexity hinders deterministic treatment of the dataset and we have to take recourse to an indeterministic treatment to make any analyses feasible. The theory of random point process provides valuable fundamental theory and techniques for analysis of the atomistic datasets. Also in the disordered dynamic systems, the instantaneous spatial distribution of particles ceases to be important. We have to look at the average systemic behavior, rather than the instantaneous behavior of the individual particles, to gain insight about the system.

The objective of this work is to:

1. Model the spatio-temporal dataset as random point process so that the techniques developed in the field of random point processes can be utilized to analyze the dataset.
2. Integrate the spatio-temporal analyses and visualization techniques to understand the properties of the dataset.
3. Visualize the dataset and the analytical results using a space-time multiresolution scheme to gain insight and enable understanding of behavior of the atomic system under consideration.

This problem is approached along three different lines. First is to analyze the spatial proximity of individual atoms to each other, and their collective tendency to form a structure. Given single atomic configuration dataset produced at one FPMD step, information about bonding, correlation, coordination and clustering is explored with generation and subsequent rendering of various types of additional data. Second is to analyze temporal proximity of atoms, which determines the dynamical behavior of the system. The atomic positions are rendered as a function of time and a variety of atomic displacement data are generated and visualized. And the third is to integrate the two types of analyses to visualize how the proximity relationships among the atoms (spatial information) vary over time (temporal proximity). All the spatial properties can also be animated to have an instantaneous view of the dynamics. Moreover, the simulation data represent an ensemble and it is preferable to visualize the time-averaged and finite-time-span behavior of the atomic structure and movement also. The ergodic ensemble enables the use of time-averaged properties such as coordination stability and bond stability as representative of the system. These three objectives are merged into one visualization system that provides a higher level picture of the underlying dataset.

1.5 Organization of the Dissertation

The organization of the remaining of the dissertation is as follows. In Chapter 2, I review the previous studies related to the atomistic visualization and present the basics of the FPMD. In Chapter 3, I explain the concept of atomistic visualization by differentiating it with the molecular visualization, describe the nature of the data to be visualized and provide a high level overview of the spatio-temporal analyses in the current context. Then in Chapter 4, I describe the assumptions and concepts used for the structural modeling of the atomistic dataset. In Chapter 5, I present the concepts used to visualize the dynamical properties. In Chapter 6, I then explain the concepts presented in the previous chapters from the implementation perspective. The algorithms and the data structures are explained for the individual concepts and performance of the individual algorithms are also presented. In Chapter 7, I describe the complete visualization system and present the high level descriptions of the optimizations implemented to make the system interactive. And finally in Chapter 8, I present the visual analyses of different mineral systems performed using the visualization system.

Chapter 2 Related Works

The visualizations have been used in science for a long time mainly because of the unique capabilities of the human visual system to internalize the graphics. The visuals are known to be more effective communicators of information than numbers and descriptions. The numbers and descriptions require multiple levels of indirection to abstract their meanings whereas graphics can be processed in parallel by human brain. Such effectiveness of the graphics comes mainly due to three variables - the two dimensions of a plane and the variation of the marks - the visual perception has at its disposal. The visual perception is atemporal [3] and it comprehends the relationships among three different components instantaneously. For example, hearing is perception of signals in time, so it is temporal in nature, whereas seeing is perception of signals in space, hence it is atemporal. For these reasons we see various visual symbols all around us that are designed to transfer the time critical information. For example, when we drive, we see traffic signals. While traveling at a high speed, the symbols are designed in such a way that they could be perceived, internalized and understood quickly. Such a design has to take our inherent ability to associate certain colors or shapes with the preconceived notions into account. It also warrants some amount of training in understanding those symbols. For example, an octagon filled with red color has become synonymous with STOP sign for people driving in roads of the United States and reaction to the STOP sign is almost automatic. We do not need to read the word written inside the octagon. But such reaction has little to do with octagon or red color per se. It is in most part due to the training and in some part due to the way we perceive the red color.

On another note, the alphabets are also a form of visualization: the visualization of sound. The groups of alphabets help us externalize and communicate our thoughts so that a reader can figure them out. For this reason, a piece of writing written in one alphabet that means so much to a person becomes meaningless to another person with no knowledge of the alphabet. The alphabets simply become set of meaningless abstract shapes. So the communication requires some set of symbols and corresponding set of meanings attached with each symbol. When no meanings are attached to the symbols, they do not convey any meaning. Such symbols cannot communicate and can become “polysemic” and “pansemic” [3], that is the symbols can mean many things or anything. Hence, such symbols are not useful for visualization.

The visualization is a means of creating visual symbols and assigning meanings to them so that they can facilitate the precise communication of the information. Such assignments have to be unambiguous. There is not one standard set of visualization symbols that are useful in every kind of visualization, but depending upon the domain of visualization, there are some common practices and common symbols. For example, in molecular visualization, the atoms are visualized as spheres and bonds are visualized using cylinders, in vector visualization, direction of a vector is visualized as an arrow and in tensor visualization, a third-rank tensor is visualized using an ellipsoid. So, different sets of visual symbols can be used for different kinds of datasets. A visualization system can use the commonly used visual symbols and when the common set of symbols limits the functionalities of the visualization system, it has to create new visual symbols and assign precise meanings to them.

2.1 Computer Assisted Visualization

Computer assisted visualization took an increased significance after the advent of computer graphics. The computer graphics has made the transformation of data into visual symbols efficient and flexible. Even though the effectiveness of graphics were known for a long time, its use for general purpose data presentation was very limited due to the amount of time it used to take to convert the data into graphical form. The availability of computer graphics brought data graphics to the forefront of data presentation and made possible myriads of different techniques such as Chernoff faces [14], star coordinates [15], parallel coordinates [16] for information visualization. Other visualization techniques were also used for scientific data, such as medical data, astronomical data and biological data.

Recognizing the importance of visualization in general scientific endeavors, National Science Foundation (NSF) published a report, "Visualization in Scientific Computing" [17]. The publication of this report is taken as the starting point of the visualization as a formal scientific discipline. The field of biomolecular structures was one of the first to find the effective use of visualization prior to the publication of the report. The report [17] recognized the visualization of biomolecular structures as one of the several major fields that could potentially benefit from the formal visualization research. Under the heading of "Molecular Modeling", the report [17] predicts "synthesis", "analysis" and "communication" of the biomolecules as the three major areas where visualization could be potentially useful to the scientists.

2.2 Molecular Visualization

The research works on biomolecular visualization produced various visualization techniques and algorithms such as molecular surfaces [18-21] and ribbon structures [22, 23]. These techniques have been used in several publicly available molecular visualization software systems. Other works include early applications reported in [24-26]. At those times, displaying large number of molecules was considered itself a feat. The researches were mainly done from the computer graphics perspective rather than from the visualization perspective. Visualization application used to be designed on an ad hoc basis. It is important to distinguish between the objectives of the research because what we search for is what we get. The objective of molecular visualization was to act as a representation medium whereas the objective of the atomistic visualization in this work is to act as an exploratory medium. The works so far reported were mainly concerned with rendering large number of atoms and lacked in the underlying intelligence to enable exploration of the structures. The atomistic visualization builds upon these computer graphics techniques to provide higher level interfaces that are useful for exploring the underlying dataset.

There are large amount of both commercial and public domain molecular and atomistic visualization works [27-29]. MolScript [30] provided feature to take a molecular dataset as an input and produce graphical representation in multiple graphical formats. It was available as a command line tool and there was no facility to interactively manipulate the molecular structure. Later, the MolScript was extended to RasMol, then to Protein Explorer [31] and ultimately into MDL Chime [32]. These molecular visualization tools provide following common features:

- A wide variety of representation (rendering and coloring) modes such as balls, points-and-lines, balls-and-sticks, space-fill, polyhedra, measurement of distances, angles, and dihedrals
- Crystalline properties (e.g., switching between primitive and conventional cell settings, displaying the Wigner-Seitz cell and Brillouin-zone), superimposition of electronic charge density surfaces
- Animations (particularly, for MD simulation trajectories) imported either from files or from a direct connection to a running MD simulation, support for variety of database file coming from a variety of simulation and experimental sources

The publicly available applications, such as AtomEye [29], XCrySDen [28], VMD [27], Chimera [33], VESTA [34] and commercially available applications, such as [35, 36] also allow interactive manipulation of the underlying dataset. AtomEye [29] features an order-*N* execution time and memory, where *N* is the number of atoms in the system. XCrySDen [28] allows isosurface rendering of charge density and Fermi Surface. Atomsviewer [37, 38] exploits an immersive environment to enable exploration of billion-atoms system. The large scale visualization systems have also been developed on an *ad hoc* basis to handle the output from large molecular dynamics simulation [39]. Many molecular visualization systems also provide scripting language interpreter [30, 40, 41]. The scripting languages let user manipulate the molecular dataset [27] and extend the functionalities of the application. This way, the application handles general cases and delegates the specific cases to the user for programming. Such delegation becomes more feasible both from implementation and user perspectives compared to handling every possible operating modes inside the application.

2.3 Graphics Subsystem

The graphical subsystem of a computer used to be tightly coupled with the software application. Each graphical application had to manage the graphics resources itself. That was one of the reasons visualization systems had to spend a lot of their time managing the resources unrelated to their ultimate objective. Voorhies, et al. [42] first showed the feasibility of representing the graphics capabilities as a system resource such that an application programmer could use it without worrying about the low-level details of the graphics management. This led to the development of various graphics libraries and ultimately removed the barrier to graphics programming allowing more applications to use the underlying graphics capabilities easily and efficiently. The graphics applications are now easier to develop than they were before due to the availability of hardware independent graphics programming languages such as OpenGL [43] and DirectX [44].

2.4 Background on Computer Graphics

2.4.1 Coordinate System in 3D Computer Graphics

In mathematics, a point in three dimensional Euclidean space is represented by an ordered pair of three real numbers, *(x, y, z)*, each representing distance along three canonical orthonormal axes, *x*-axis, *y*-axis and *z*-axis. But when points needed to be transformed for

computer graphics purpose, such representation does not result uniform transformation representations [45]. A fourth component, *w*, is added to the ordered pair, *(x, y, z, w)*, to make the transformation homogeneous and such representation is commonly known as homogeneous coordinate. The points with $w = 0$ are called points at infinity. The transformation is called homogeneous because various affine transformations such as translation, rotation, scaling and shearing can be composited together to create single complex transformation. Such transformations can be represented by a 4x4 matrix.

Geometrical shapes in three-dimensional space are constructed by compositing varying number of primitives, such as points, lines and polygons. Individual points and shaper are transformed to create a three dimensional scene. The objects in the scene can have visual properties that help in rendering them. When all the visual properties of the scene have been determined, the entire three-dimensional scene goes through a 3D graphics rendering pipeline. At the end of the rendering pipeline a resultant 2D image is generated that can be directly mapped onto the computer screen.

2.4.2 3D Graphics Rendering Pipeline

A 3D scene goes through 3D graphics transformation pipeline. The pipeline consists of multiple stages that transform the scene into a two-dimensional image that can be displayed on the screen. This pipeline is described in many standard textbooks for computer graphics. Here, I describe transformational stages of the rendering pipeline in brief to provide context for the following discussions. Viewing in 3D is complicated by disparities in the dimension of the scene world and the dimension of the display world. A scene is described in 3D world and the display screens are 2D by nature.

The Figure 2-1 shows a view of 3D graphics rendering pipeline used in modern graphics processors. Each object comprising a 3D scene is specified in its local object coordinate, x_o. The object coordinates are transformed into eye coordinates, x_e, using the model-view matrix, M, to form a composite scene. At this stage, complete scene is described using a single coordinate system, commonly called world coordinate system. The eye coordinates are then transformed by the projection matrix, P, into the clip coordinates, x_c. There are two major projection mechanisms: orthogonal projection and perspective projection. The orthogonal projection simply maps the 3D scene into a 2D screen without taking the real world distortion that makes the

objects that are farther smaller into account. The perspective projection takes that distortion into account. An image generated using perspective projection looks more realistic than an image generated using orthographic projection. The clips coordinates are then clipped into a 3D view volume. The portions of the scene falling outside of the view volume are clipped away. As a final step, the clip coordinates are mapped into window coordinates using the viewport matrix, V.

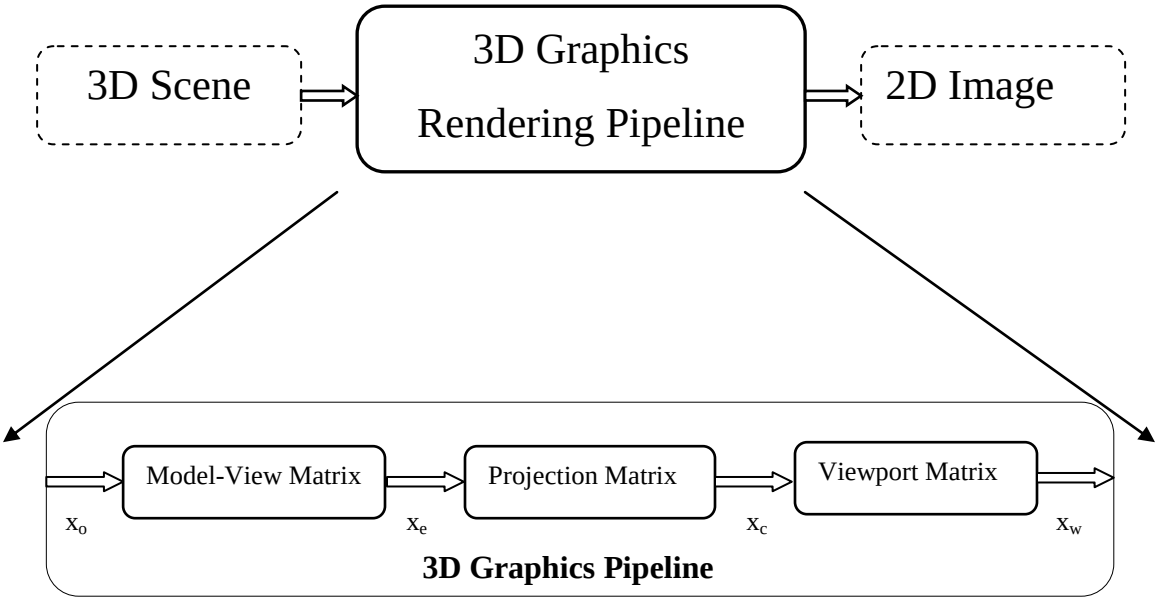

Figure 2-1: 3D graphics rendering pipeline

A modern 3D graphics rendering pipeline contains many other intermediate stages, such as texture mapping, lighting calculations, rasterization, shading, blending, etc that add flexibility in rendering varieties of shapes with different colors and qualities. The texture mapping in combination with the shading enables a wide range of effects, such as shadow mapping, bump mapping, environment mapping, etc. Shader programmability in the modern GPUs also enables evaluation of the various lighting models that are important in making a scene realistic.

2.4.3 Programmable Graphics Processors Units

The 3D graphics rendering pipeline is implemented separate from the central processing unit (CPU) as a graphics processing unit (GPU). GPU receives 3D data and commands from the CPU and acts upon them according to the commands issued by the CPU. The graphics dataset

can be operated upon on highly parallel basis. As a packet of the graphics data can be made independent of other packets of the graphics data, the stream programming model is found to be very suitable for computer graphics applications [46].

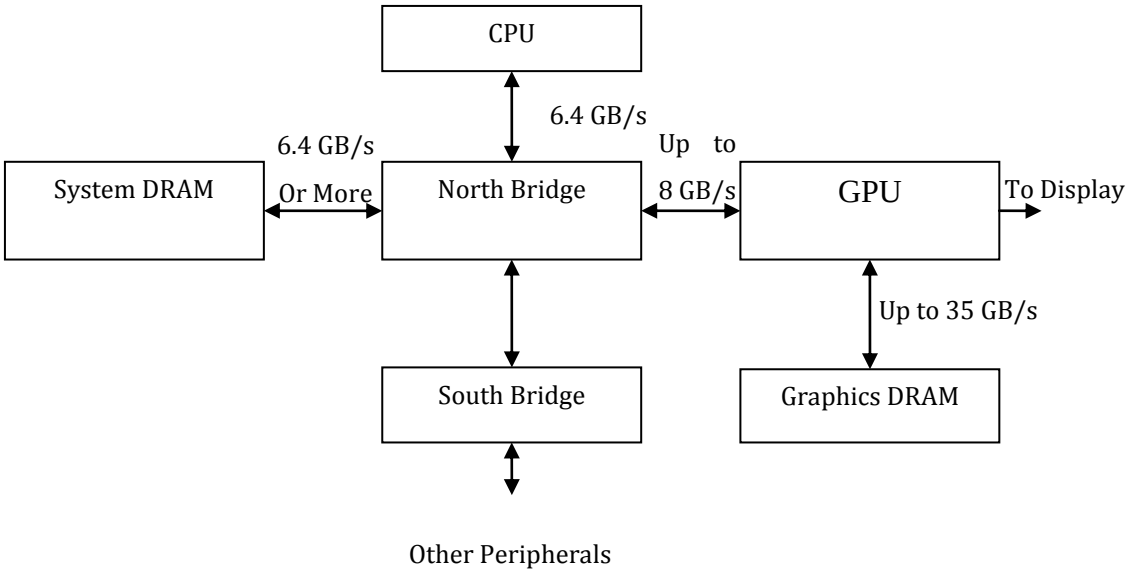

Figure 2-2: The overall system architecture of a PC

The graphics operations are computationally intensive and performing such operations inside the CPU itself brings the performance of entire system down. The initial rationale for designing a separate graphics accelerator was to have decent graphical output without bringing the performance of the entire system down. But there is a fundamental limitation on how much total performance enhancement can be squeezed out by offloading the graphical operations to the GPU and making the GPU faster. Such limitation is commonly known as the Amdahl's law. There was another benefit of offloading. It freed the CPU and anything could be done with the graphical data inside the GPU. The result was an unsurpassed realism in the graphical renderings. The Figure 2-2 shows the place of the GPU in the complete PC architecture. The GPU is connected with the display and there are high bandwidth pathways from CPU to GPU and GPU to the system memory and the graphics memory. Modern discrete graphics cards contain their private graphics memory that is connected with GPU via a very high bandwidth bus. The graphics memory is used to store scene specific data, textures, shaders.

Recently, there is a movement in opening up the GPU instruction sets to make them more programmable and to enable faster and more realistic rendering possible [47]. Such opening up

of the GPUs has resulted in a movement of harnessing tremendous amount of computing power offered by GPU for purposes other than graphics. Such works encompasses from database query handling, video compression and decompression, real time simulation, and there are efforts to extend the power of GPU to number theory research as well. There are mainly three programmable stages in modern GPU architectures: vertex program, geometry program and fragment program. The vertex program operates on individual vertices passed from GPU as part of a 3D scene. The program can perform lighting, texturing and transformation on every vertex. The next stage is the primitive assembly and can be programmed using geometry program. The geometry program operates on each primitive. After a primitive passes through geometry program, it is rasterized and converted to fragments. The fragment program can then operate on individual fragments and perform lighting, texturing and finally shade them. There are also works to unify the complete 3D rendering process into a fully procedural model so as to maintain the locality of computation and minimize the bandwidth consumption [48].

2.4.4 Lighting Models

The purpose of computer graphics is mainly to create an illusion of reality inside a computer. Modeling the interaction of lights with the objects in a 3D scene is important in creating such an illusion. There are different lighting models of varying degrees of mathematical and computational complexities. We have to trade off among models according to our need for realistic scene and computational cost we are willing to incur. The lighting models are used to calculate color at every pixel on a 2D image. The lighting models are categorized into three broad categories [49]: i) empirical model, ii) transitional model and iii) analytical model. The transitional and analytical models are mainly useful in generating real-world scenes. They compute the light-material interactions analytically utilizing the theory of underlying optics to some extent. These models are computationally very intensive and are unsuitable for interactive applications. The empirical models however use geometric approximations to model the light-material interactions and are computationally less intensive. Scientific visualization is not concerned with visual reality per se because purpose is to make invisible data visible. The classical definition of ‘realistic’ does not apply in this case. We simply want to create an illusion of visibility for something that actually is invisible. For our purpose, we restrain ourselves to use simple empirical lighting models that provide enough visual ‘reality’ from scientific visualization perspective. Two of the empirical models are adequate for our purpose: Gouraud

lighting model and Phong lighting model. These two models are computationally simple and provide enough visual realism for the visualization.

2.5 Molecular Dynamics

The increase in computational and storage capacities of computers and corresponding decrease in the price have made the computer simulation of materials at atomic and bulk level possible. These simulations utilize diverse set of algorithms to simulate the systems depending on the system size, available computational resources and the required precision of the simulation. The atomistic simulations using MD algorithms assume atoms are in a potential field and they interact with each other according to QM equations. The solutions of these QM equations determine the potential distribution in the simulation space. The forces acting on each atom is determined by strength of the gradient of the potential field and the positions of the atoms are determined by integrating the Newton's equation of motion. In FPMD, the interatomic forces are derived from a fully self-consistent solution of the electronic structure problem within density functional theory. Unlike the methods based on empirical and simplified potential models, such quantum mechanical method is computationally very intensive and is more accurate than the empirical methods. QM method uses pseudopotentials that are computed for each individual atom using the atom's approximated electronic wavefunctions. The pseudopotentials have been found to accurately approximate the properties of the atoms, irrespective of the chemical environments they are in. It is called the transferability of the pseudopotential. Such transferability of the pseudopotential of an atom to different chemical environments is mainly due to the existence of a nuclear core where "the charge density is practically independent of the chemical environment" [50].

The ultimate goal of a material scientist is to understand what happens at the atomic level in a system and be able to relate that to the bulk level phenomena. There are various techniques devised for serving that purpose [51] and many of these techniques are used in the experimental settings. Some materials exist in laboratory conditions, while others exist only as theoretical models and their physical existence have not been experimentally verified. Many works point toward the existence of materials in the Earth's lower mantle that normally do not exist in the laboratory conditions because of the extreme conditions that exist at the lower mantle. The extreme pressure and extreme temperature conditions of the mantle is unsustainable in normal

laboratory environment and computational tools are used to study the bulk level properties of such materials using their theoretical models. Once the bulk properties are understood theoretically, naturally occurring variants of the minerals are studied by creating extreme pressure and temperature conditions in laboratory [52-54].

The simulations of bulk materials use lower precision algorithms than the simulations of atomistic materials. The low precision methods have been successfully used in simulating system with billions of atoms [55]. The outputs from such large scale simulations are less precise than the outputs from QMMD simulations. Such outputs help reveal bulk level properties of the system. When QMMD simulation is run for reasonably long time, the aggregate properties computed from the outputs of such simulation reflect both the bulk level properties and the micro-level properties such as coordination distribution very accurately. This applies even when the system contains less than one thousand atoms. This is not necessarily true for simulation using less precise simulation methods. So, detail analysis of the outputs from QMMD simulations is very important and useful.

2.6 Spatial-Temporal Analysis

We want to understand the underlying structure among the points by statistically analyzing the dataset. Such statistical analysis is common among statisticians and economists to analyze economic, population, forestry, and ecological data. But there have been little systematic use of such approach to analyze the atomistic data recently. Some initial works in statistical mechanics involved use of such approach in formulating the statistical theory of particles. Later with no reliable access to the complete trajectory information of the individual particle in a system, such first principle statistical approach gave way to higher level analyses. Reliable and complete trajectory information for each individual particle is accessible for analysis with QMMD simulation. In this context, a formal statistical analysis from first-principle is useful.

The statistical methods include analysis of distribution functions, nearest neighbor analysis, covariance analysis. The spatial and temporal data, such as information about the distribution of various types of trees in a geographical region, distribution of animal species in an ecological unit, price fluctuation within a certain period of time, are limited in size. They are analyzed to understand the pattern of distribution of multiple types of points. In the trees data, we may want to find the distribution of one type of trees with respect to another type of trees and see

if there are types of trees that grow together. In case of species, we may want to find the distribution of one species with respect to another species to find out if there is any affinity between species. The null hypothesis of such analysis is complete spatial randomness, i.e. we assume there are no affinities among various types of trees or species. With analysis we want to either confirm the hypothesis or figure out how far the dataset deviate from the hypothesis and quantify such deviation. If a system satisfies the null hypothesis, then there will always be a constant number of particles per unit volume in the system. If the data points show affinities between each other, then the number of particles will vary from the constant value.

There are multiple types of atoms in the atomistic dataset and distribution of various types of atoms around each other contains interesting structural information. Since we assume there are correlations among various types of atoms, we need a mechanism to quantify such correlation. Even though general correlation function is useful, it is mathematically and computationally intractable. Since the system we are interested in are equilibrated and made homogeneous, radial distribution function is useful for such quantification. It provides a complete spatial map of the atoms. The pattern of the RDF provides useful information about the correlation among atoms of different species. The structural information contained in the RDF is analyzed using various coordination environments, nearest neighbor clusters, common neighbor clusters and local ring structures to explore the properties of the system.

The tendency of an atom to move in a certain preferred direction is analyzed using the covariance matrix of the trajectory for the atom. The covariance matrix is a 3x3 matrix and it has three eigenvalues and three eigenvectors. The eigenvalues and eigenvectors of the covariance matrix are also called principal components. The principal component analysis (PCA) has been used extensively to understand multivariate datasets descriptively. The three eigenvectors of the covariance matrix gives the three orthogonal directions in which the data has the maximum variance. The three eigenvalues gives the amount of variance along those directions. Visualization of tensors using ellipsoids is commonly used in diffusion tensor imaging.

2.7 Analysis, Visualization and Exploration

The existing molecular visualization systems provide many great features for rendering, but they lack in the integrated approach to the exploratory visualization. The analytical tools are crucial to enable exploratory visualization of molecular structures in a data specific manner. An

atomistic dataset does not contain structural information and each dataset has dataset specific properties that enable extraction of the relevant structural information. Such lack of a mechanism to extract those properties in the visualization systems is a big drawback.

The analysis and visualization are regarded as two separate components and are treated separately. The analytical algorithms are used to understand the underlying structures of the system and the visualizations are used to see those structures. The analytical algorithms are usually not integrated into the visualization because they are computationally intensive and are not suitable for interactive visualization. Interestingly, such lack of integration of analytical tools is not the problem only in the atomistic visualization, rather is felt across all the data mining community [56]. Also, visualization itself is only a means not an end; it is just a tool [57] and an unusable visualization algorithm is almost worthless except sometimes for its artistic value [58]. Such lack of integration of analysis with the visualization renders the tool almost useless. The visualization systems designed for atomistic systems with exploration in mind are also non-existent. The approach we advocate in this work enables exploration of various properties of the atomistic dataset at space-time multiresolution and represents a more complete approach to the visualization than the currently existing visualizations. This scheme also allows the extraction and the visualization of detailed structural and dynamical information.

The spherical model of atoms has become so ingrained into the consciousness that whenever atomistic visualization is mentioned, an image with spheres of many different colors and cylinders connecting many different atoms together automatically comes into the mind. Such automatic association of attributes with the raw data points somehow hinders an objective look of the dataset simply as set of points with attendant attributes. Bertin [3] was the first to advocate the separation of information from its presentation and was able to analyze the relationship of information with different types of visual signs that could be used to represent the information. Similar approach is needed in atomistic visualization as well. By envisioning atomistic positional dataset as a set of points with multiple attributes attached to each point, various spatial and temporal analytical methods can be readily applied. Application of such methods and integration of visualization algorithms with them enables more effective means of visual exploration of the atomistic dataset.

Chapter 3 Exploratory Atomistic Visualization

Images have limitations in how much information they can convey and are also immune to the interactive manipulation. When the relevant information is hard to identify, it cannot be visualized using images. In such case, it must be possible to explore the parameter space to understand various properties and extract meaningful information. The atomistic dataset is one of such datasets that must be explored to generate the understanding. There are no predefined set of information that can be simply visualized. The data must be looked from multiple perspectives. The responsibility of finding the useful information is then delegated to the user with the visualization acting as an interface to the dataset.

3.1 Molecular Visualization vs. Atomistic Visualization

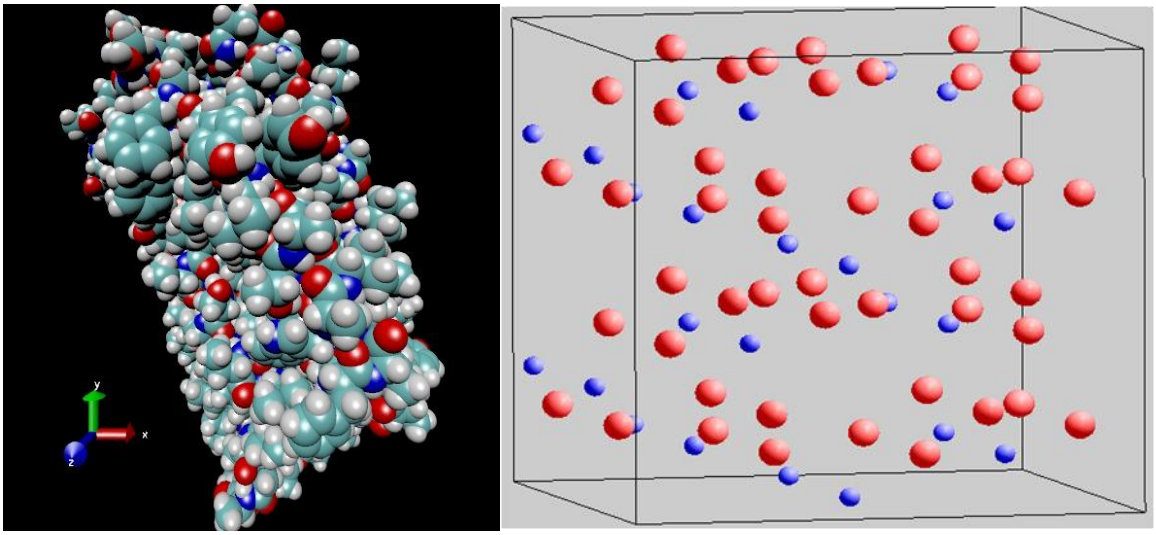

Figure 3-1: (left) Molecular dataset, and (right) atomistic dataset

There are no commonly agreed upon definition of atomistic visualization. In this chapter, atomistic visualization is defined from the perspective of this work, differentiated with the molecular visualization and put into the proper context of the exploratory visualization. Let us first try to understand the concept of *atom* and *molecule* as defined by IUPAC:

Atom: Smallest particle still characterizing a chemical element. It consists of a nucleus of a positive charge (Z is the proton number and e the elementary charge) carrying almost all its mass (more than 99.9%) and Z electrons determining its size.

Molecule: An electrically neutral entity consisting of more than one atom. Rigorously, a molecule must correspond to a depression on the potential energy surface that is deep enough to confine at least one vibrational state.

International Union of Pure and Applied Chemistry (IUPAC) is a world authority on chemical nomenclature, terminology, standardized and other critical chemical information. From the definition above as adopted by IUPAC, *atom* is defined as a single entity where as *molecule* is defined as a composite entity consisting of more than one atoms.

A molecular dataset and an atomistic datasets (Figure 3-1) can be differentiated along the similar lines. A molecular dataset contains information about various groups of atoms and their inter-connectivity. The visualization of such datasets involves mining of information from atoms and their connections and such visualization method is commonly called the "**molecular visualization**". The atomistic dataset on the other hand contains information about individual atoms and does not define how atoms form group with each other. The visualization of such datasets involves extraction of connectivity information from the atomistic dataset and utilizing that information to visualize the dataset meaningfully. Here, we define such visualization technique as the "**atomistic visualization**". It operates at a lower level in analytical hierarchy than molecular visualization. The atomistic datasets are richer in information content than the molecular datasets. They are generated during atomistic molecular dynamics simulations and generally contain sets of positional snapshots of the simulated system. The set of all the system snapshots generated during the MD simulation implicitly contains complete information about the system dynamics and structural changes it goes through. The objective of atomistic visualization is to make dynamical and structural information hidden in the atomistic dataset visually manifest.

3.2 Terminologies

Before going into the details of the dataset, definitions and concepts that are used in successive sections are defined here.

Lattice Vector

Let $a \in R^3$ is 3D vector. If a^i, where $0 \leq i < 3$, gives length of the simulation space on each side i, then a is said to be a *lattice vector*.

Basis Vectors

Vectors v_1, v_2 and v_3 are said to be *basis vectors* for an atomic system, if position of atoms in the system can be specified by linear combinations of these three vectors.

K Matrix

A K-Point matrix is a 3x3 matrix with the basis vectors forming columns of the matrix. i.e. if K is the K matrix, then,

Equation 3-1

$$K = [v_1^T v_2^T v_3^T]$$

Periodic Boundary Condition

In a system, suppose there is an atom at position $r \in X$. If all the self consistent computations during the simulation is done with the assumption that there are copies of the atom at positions $(r + av_1 + bv_2 + cv_3) \in R^3$, where v_1, v_2 and v_3 are the basis vectors of the system and $a, b, c \in R$, then the system is said to follow *Periodic Boundary Condition.*

Reduced Coordinate

For a position p, if each component of p, $0 \leq p_i \leq 1$, then p is said to be in *reduced coordinates.* A position p in reduced coordinates is converted back to a normal 3D coordinate by multiplying the position with the corresponding K matrix as:

Equation 3-2

$$p_{3d} = K.p$$

3.3 Ergodic Hypothesis

According to the ergodic hypothesis, the time average value of an observable, which is determined by the dynamics, is equivalent to an ensemble average, which is an average over a large number of systems all of which have identical thermodynamic properties but are not identical on the molecular level [59]. This hypothesis is generally accepted to be valid even though it has not yet been mathematically proven to be so. We also assume this hypothesis to be

valid. This hypothesis provides us with a theoretical justification that if we simulate a system long enough and the parameters are evaluated as the time averaged quantities from the simulated system, those parameters effectively represent the ensemble average of the original system. The underlying assumption of this hypothesis is that when the system is run for sufficiently long time, it goes through the complete set of possible configurations. This assumption simplifies the way time averaged and space averaged parameters are treated during the visualization.

3.4 Molecular Dynamics

Molecular dynamics (MD) simulations of Earth materials used to be performed for crystalline systems with periodic boundary conditions, where symmetry constrained the size of the simulation box and the atomic configuration, and only a few free positional parameters needed to be considered explicitly. For example, in Magnesium Oxide, its conventional simple cubic structure requires only eight atoms whose positions are fixed by its cubic symmetry. Simulations of the disordered systems such as liquids and defective crystals are increasingly becoming common and precise due to rapid advances in computational resources and capabilities. Such simulations require larger atomic systems that partially or completely violate the crystal symmetry. They also have to keep track of large number of degrees of freedoms.

A point in space has only one degree of freedom if it is constrained to move in a single direction, it has two degrees of freedom if it is allowed to move in a fixed plane, and it has three degrees of freedom if there are no restrictions on its movement [60]. There are no restrictions in atomistic system simulated using FPMD except the ones imposed by the quantum mechanical equations. Such restrictions are implicit and they can be ignored for the analytical purpose. In such case, each atom in the system, represented as a single point in space, has three degrees of freedom because it can move along any of the three directions. The MD simulations do not keep track of rotational degrees of freedom of each atom because spherical atoms are invariant under rotations. For example, the data for the liquid Magnesium Oxide system with 216 atoms has 648 discrete positional degrees of freedom because each atom in the system is free to move in all three directions with no geometric constraints. Unlike in a perfect crystalline phase, the atomic positions in liquid phase are strongly correlated, but they lack long-range order. Even the short-range (local) order is highly temporal and transient fluctuations are expected to occur. MD simulation of such liquids can be treated according to the ergodic hypothesis.

An atomistic system is initially defined by a set of properties that represent atomic positions, atomic types, and constraints to be satisfied. Let an atomistic system contains n_a number of atoms, then

Equation 3-3

$$P = \{p_i \in X \subset R^3 \,|1 \leq i \leq n_a\},$$

is the set of the atomic positions in the system, which can be considered the discrete degrees of freedom of the system. *X* is the simulation space.

Λ is the set of atomic species in the system, and $|\Lambda|$ is the number of atomic species in the system. Larger the number of atomic species in the system, more complex will be the interactions among the atoms in the system.

C is the set of constraints that the system has to satisfy. In FPMD, this is implicitly defined by the atomic species, number of valence electrons and orbital structures of the atoms and quantum mechanical equations guide these constraints.

Then, $A = \{P, \Lambda, C\}$ is the atomic system. For example in an FPMD simulation based on canonical *NVT* ensemble, the number of atoms in the supercell (n_a), the volume (V) of the supercell, and the temperature (T) of the system are fixed. The smallest crystal supercell is equivalent to a unit cell. The complete crystal structure can be inferred from the unit cell by the symmetry groups of the crystal, whereas liquids do not follow any symmetrical structure. The objective in liquid state simulation is to simulate the largest possible and computationally feasible supercell. Simulations of large supercells result more accurate output. A system is assumed to satisfy periodic boundary condition. That means the simulation is performed in a three-dimensional torus and unwrapped into a rectangular 3D space. It effectively replicates the system infinitely and approximates the bulk matter simulation.

3.5 Simulation Progress

An initial snapshot, $P(0)$, along with a pseudopotential field and wave function characteristics of the atomistic system in the simulation space are specified at the beginning of the simulation. A potential field distribution in the space is computed using the quantum

mechanical (QM) equations by taking electronic and atomic properties of the atoms into consideration. The gradient of the computed potential field determines the magnitude and the direction of the force acting on each atom in the system.

During the simulation, a new system configuration, $A^{'}$, is generated at a fixed time interval, Δt. The collection of $A^{'}$'s over a given simulation time duration, τ, describes the system dynamics, D, i.e.,

Equation 3-4

$$D = \{ A^{'}(t) | 0 \leq t \leq \tau \},$$

Here, note that $A^{'}(0) = A$. According to the ergodic hypothesis, properties calculated from D represent the ensemble average of the system when τ is sufficiently long.

Let $\overrightarrow{F_i}(t)$ be the force acting on atom i at time t. Newton's law of momentum gives the acceleration experienced by the atom i. If $\overrightarrow{p_i}(t)$ is position of the atom i and m_i is mass of the atom at time t, then position of the atom i in time $t + \Delta t$ is computed by integrating the Newton's equation of motion with time interval Δt.

Equation 3-5

$$m_i \frac{d^2 p_i}{dt^2} = F_i(t)$$

Where,

Equation 3-6

$$F_i(\mathrm{t}) = \sum_{\substack{j=1 \\ j \neq i}}^{j = n_a} f_{ji}\ (t)$$

The interatomic forces, $f_{ij}\,(t)$, are derived from a fully self-consistent solution of the electronic structure problem within density functional theory. This process is repeated for the required number of time steps. The individual atoms are free to move anywhere obeying the quantum mechanical restrictions and there are no additional external restrictions, such as how

close two atoms can come to each other or to maintain certain structural characteristics intact as is the case in other less precise molecular dynamics simulations.

3.6 Description of the Dataset

Let A be the system with n_a number of atoms and Λ be the set of species in the system. Each of the atoms in the system are of species $\alpha \in \Lambda$. Let $X \in R^3$ be the simulation space and $p_i(t) \in X$ be the position of atom i at time t.

Let $P(t)$ be the time-varying positional dataset generated from FPMD simulation of the atomistic system, A. The dataset contains set of positional snapshots of the system at different time instants during the simulation. Individual snapshot specifies positional configurations of the n_a atoms in the system.

Suppose simulation is performed over n_{step} timesteps with each timestep accounting for Δt units of time. Then, the simulation covers τ ($n_{step} \times \Delta t$) time and the complete dataset consists of n_{step} snapshots of the atomic system. Thus data can be expressed as:

Equation 3-7

$$S = \{P(t) | 0 \leq t \leq \tau\}$$

where

Equation 3-8

$$P(t) = \{p_i(t) | 1 \leq i \leq n_a\}$$

is atomic configuration at time, *t*. Here, I think it is important to stress one more time that the positional dataset from a molecular dynamics simulation of the atomistic system is just a set of points and each point is decorated with additional species information.

3.7 Spatio-Temporal Analysis

In a disordered system, single snapshot of the atomic configuration becomes irrelevant as the properties of the system are guided by the average configuration of the system. During the analysis, the dataset is assumed to be stationary and isotropic in both space and time, but nothing is assumed about the underlying point process model generating the dataset. The stationarity

assumption is valid because when a disordered system is simulated long enough, it reaches an equilibrium stage becoming stationary in time. The system goes through equilibration stage at the beginning of the simulation and the initial portion of the dataset has to be ignored for this assumption to be valid. A finite system is not stationary in strict sense but a system satisfying periodic boundary conditions represents an infinite system and can be stationary in space. The assumption that the dataset is isotropic is mainly to make the mathematical modeling easy. It makes the system rotation invariant and pair correlation function becomes dependent on the distance between the two atoms.

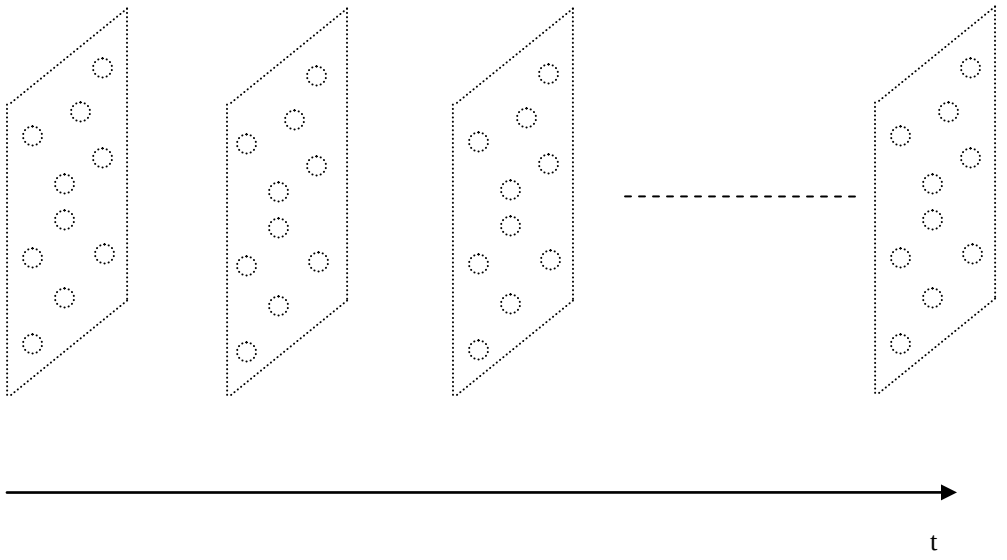

Figure 3-2: Spatio-temporal point process

Let S be the system that we want to analyze defined as above. The periodic boundary condition assumption justifies the use of toroidal edge corrections during distribution function computation. For simplification, an atom is stripped off any other attendant information, such as species, atomic number, covalent radius etc. and the sole objective of the analysis becomes to find any spatio-temporal structure in the dataset of such time-varying points. The points are later assigned the attendant properties for detailed analysis in terms of those properties.

Let V be the volume of X and there are n_a points in the system. Then, ρ, the number density of the system, is defined as:

Equation 3-9

$$\rho = \frac{n_a}{V}$$

Before going into the details of further analysis, let us define the concepts of complete spatial randomness and probability distribution function. Throughout this discussion we will focus on three-dimensional point process only because our dataset is three dimensional in nature.

3.7.1 Complete Spatial Randomness

Let ρ be the number density of a system satisfying *complete spatial randomness (CSR)* [1] [61]. Then, the system has ρ atoms per unit volume. So, on average a volume element ΔV contains $\rho\,\Delta V$ number of atoms atoms within it. This means the particles in the systems do not interact with each other and are uncorrelated. One example of such a system includes a gaseous system. The particles in a gaseous system do not interact with each other and are independent. They are distributed completely randomly in space due to the lack of correlation. Such systems are also called homogeneous Poisson process and intensity of the Poisson process is the number density, ρ, of the system. The intensity is assumed to be constant at every point in the space, X. If intensity of the process is different at different points in the space, the process is inhomogeneous Poisson process [62]. The systems satisfying CSR are not interesting to analyze but while analyzing other spatial systems, CSR is taken as a null hypothesis. How far a system deviates from CSR is used to quantify the inhibitory or the clustering interactions among the particles in the system.

3.7.2 Probability Distribution Function

A probability distribution function is a measure that assigns a number to an event. The number is the probability of the event. Probabilistic distribution of the points is computed by considering individual snapshot as realizations of the random point process (Figure 3-2). Such cumulative consideration can explain the probabilistic structure of the system.

The randomness is a means of avoiding the details that arise due the intractable amount of interactions among the entities in a system. For example, when a coin is tossed, large number of physical phenomena affect the outcome of the toss. When the coin toss is modeled as sampling of a random space and the outcome is treated as a random variable, the entire gamut of

physical phenomena vanishes. The outcome of the coin toss can then be probabilistically explained. By treating an outcome as a random variable, we effectively isolate the outcome from the set of other interactions, and by assigning a probability to the outcome, we put the outcome back into its natural environment in a mathematically tractable way. The atomistic dataset is one of such cases where by treating individual data-point as a random variable, complication of understanding the quantum interactions that went in determining the position of all the particles is made more tractable.

3.7.3 K-Function

For a point process with intensity, λ, the K-function is defined as [62]:

Equation 3-10

$$K(r) = \lambda^{-1} E[number\ of\ events\ within\ distance\ r\ of\ an\ arbitrary\ event]$$

For a multivariate point process with *k* different types of points, the K-function is defined similarly as:

Equation 3-11

$$K_{ij}(r) = \lambda_j^{-1} E[number\ of\ points\ of\ types\ j\ within\ distance\ r\ of\ an\ arbitrary\ type\ i\ event]$$

Let us define the K-function for a monotype point process first. Suppose $A \subset X$ and $B \subset X$ are two volume inside the simulation box and $N(.)$ is a counting measure. $N(A)$ is the number of particles inside the volume A. Then, Ripley [63] defines the K-function as:

Equation 3-12

$$E\big(N(A)N(B)\big) = \lambda v(A \cap B) + \lambda^2 \int_0^\infty v_r(A \times B) dK(r)$$

Where

$$v_r(A \times B) = \int_A \sigma_r[\{\boldsymbol{y} - \boldsymbol{x} | \boldsymbol{y} \in B, d(\boldsymbol{x}, \boldsymbol{y}) = r\}] dv(\boldsymbol{x})$$

σ_r is the uniform probability on the surface of the sphere of radius r centered at the origin. The first term in Equation 3-12 comes if there are particles common to both A and B. If there are no common particles, then this term becomes zero. The second term counts the pair of particles, one in A and another in B. Let us suppose A and B are two disjoint sets. Then, we can safely remove the first term and the Equation 3-12 becomes:

Equation 3-13

$$E\big(N(A)N(B)\big) = \lambda^2 \int_0^\infty v_r(A \times B) dK(r)$$

Ripley gives three separate interpretations of the K-function of which we are mostly interested in two of them.

- $\lambda K(r)$ is the expected number of particles within r of any arbitrary point, and
- $g(r) = \lambda^2 \frac{\frac{dK}{dr}}{c(r)}$, where $g(r)$ is the radial distribution function for a stationary and isotropic system and $c(r)$ is the area of the surface of a ball of radius r

The first interpretation is the same as the one we used at the beginning of this section to define the K-function itself. We will come to the second interpretation in the next chapter when we explain the radial distribution function. For computational purpose, Ripley also gives an unbiased estimator for the K-function as:

Equation 3-14

$$\widehat{K}(r) = \frac{V \sum k(\boldsymbol{x}, \boldsymbol{y})}{n_a^2}$$

The $\frac{1}{k(\boldsymbol{x},\boldsymbol{y})}$ in the equations is a correcting factor introduced to correct the error due to the finite size of the sample space. When there are more than one types of particles, this straightforward approach does not work. In such a case, instead of simply analyzing the distribution of particles in general, more information can be gathered if distribution of one type of particle with respect to another type of particles can be analyzed. Lotwick et al. [64] extended Ripley's K-function to analyze the point process with multiple types of points and provided an unbiased estimators for the K-functions useful for analyzing such point processes in terms of two

separate estimators. Again, due to the periodic boundary conditions, the estimator is simplified. Let's say there are points of type i and type j with n_i and n_j points respectively. Let us define a function, $a(x, y, r)$ be zero if distance between x and y is greater than r otherwise 1. Then the estimator given by Lotwick et al. becomes:

Equation 3-15

$$\widehat{K_{ij}}(r) = \frac{V}{n_i n_j} \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} a(x_p, y_q, r)$$

These estimators are used to estimate the radial distribution functions.

Chapter 4 Structural Analysis

The structural information are used to explore the spatial properties of the underlying system. The results from the simulations represent strongly correlated dynamical scattered data and quantification of the correlation is the first step for structural analysis. Radial distribution function (RDF) is the starting point for the quantification of the correlation. The RDFs are categorized into total RDF and partial RDFs. Each RDF provides information to be used subsequently.

Coordination environments are useful in understanding the local structures. The environments are expected to be very complex for the disordered systems like liquids. In particular, the coordination and bonding environment are non-uniform in both space and time and the questions such as - What are different coordination types? How stable is a coordination state if it exists? What are possible coordination states for different atomic pair types? - naturally arise during the process of understanding. Similarly the cluster analysis and the ring structure calculations provide further insight into the structure of the system under consideration.

In this chapter, first the physical significance of RDF is explained in detail. Then subsequent structural analysis algorithms are explained with respect to the RDF. The pseudocode for computing all the parameters are explained in the subsequent chapter.

4.1 Radial Distribution Function

The radial distribution function (RDF) is one of the many pair-correlation functions possible in a dynamical system. It is used to understand the average radial distribution of the particles. The underlying assumption during the derivation of RDF is that the system is isotropic and stationary, as it is already mentioned in the previous chapter.

The RDF describes average radial packing of the atoms and is important for three main reasons. First, RDF is one of the frequently used methods for concisely characterizing structure of liquids [65-67] and also tells us about the structure of complex systems such as colloidal suspensions. In such systems, the particles are in constant motion and a single snapshot of the system shows only the instantaneous disorder. The structure has meaning only in the average sense and the RDF is often considered an effective way of describing such average structure.

Second, the RDF can be measured experimentally using neutron and X-rays diffraction techniques allowing us to make a direct comparison between the experiment and the simulation. Third, in the case of the pair-wise potential function, thermodynamic quantities particularly the energy and pressure can be expressed directly in terms of RDF.

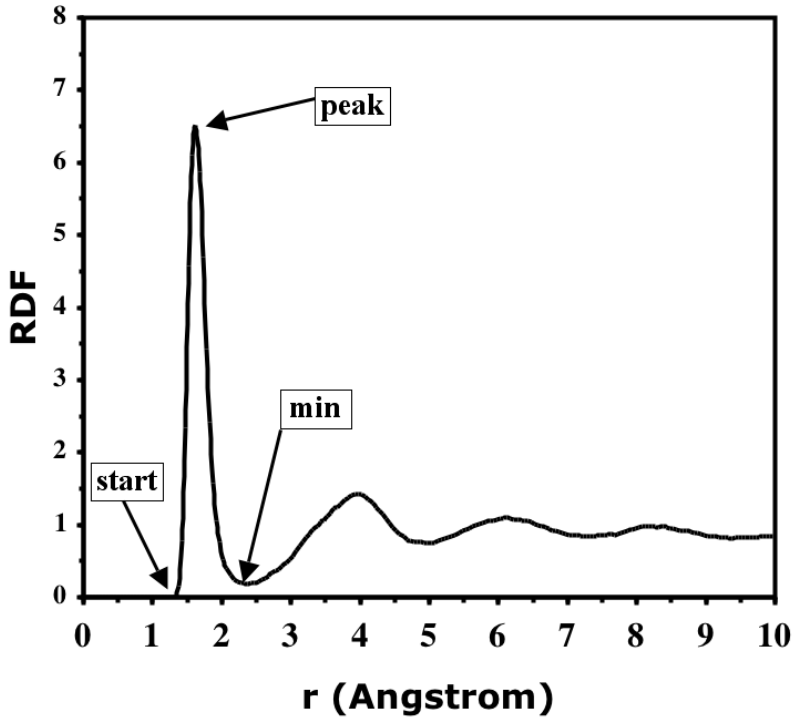

Figure 4-1: Radial distribution function of a liquid

Formal definition of RDF and its significance in describing structure of an atomic system is given in Section 4.1.1.

4.1.1 Definition and Significance

In this section, RDF is defined and its significance in characterizing the structure of the underlying system is elaborated.

Definition 4-1: Radial Distribution Function

Let g(r) be the RDF of a system, then for any particle i in the system, there are g(r) number of atoms per volume at a radial distance r from the atom i.

In deterministic systems such as solid or crystalline systems, RDF is not as useful as it is for disordered probabilistic systems such as liquids. In solid or crystalline systems, each atom sits at its equilibrium position in a lattice. Thermal vibrations do not have enough energy to dislocate the atom from its current lattice position. So, RDF shows peaks at fixed distances and is zero everywhere else.

On the other hand, probabilistic systems such as liquids show interesting patterns in their RDF (Figure 4-1). At small distances, RDF is zero because two atoms cannot come closer to each other than a certain minimum distance prohibited because of the large repulsive force between two atoms at those distances. Atoms are modeled as soft spheres because of the existence of such a strong force. RDF shows non-zero value and increases sharply to reach a peak value for the distance greater than that minimum distance. As the radial distance increases the RDF fluctuates around unity and at large distances it remains almost constant at unity.

Such characteristic of the RDF shows that liquids have short range structures and lack long range structures. When the system is not in a perfect liquid state, RDF shows multiple peaks at different distances before it settles around unity. When there are multiple peaks instead of just one peak, then the system is not already in a completely liquid state and there are some residual crystalline substructures within the system. For a system with more than one type of atoms, such simplistic description of RDF is insufficient. We have to differentiate between the RDF that describes radial distribution of the particles irrespective of their types and the RDFs that describe radial distribution of the atoms of certain type around atoms of another type. The former is called total RDF (TRDF) and the latter is called partial RDF (PRDF). Different PRDFs shows different characteristics.

4.1.2 Total Radial Distribution Function

Total RDF (TRDF) considers the particles as attributeless points and describes their average radial distribution around each other as such. It is defined as:

Definition 4-2: Total Radial Distribution Function (TRDF)

TRDF gives the number of atoms per volume that can be found at a shell of radius r with width Δr. TRDF is similar to RDF as defined in Section 4.1.1. RDF and TRDF do not differentiate according to the type of particles.

TRDF is never used in analyzing multi-species system and only used while analyzing the elemental systems. In elemental systems, there is one RDF as there is only one species.

4.1.3 Partial Radial Distribution Function

Partial RDF (PRDF) is always defined for a pair of atomic species, say $\alpha \; and \; \beta$. Then, PRDF is defined as:

Definition 4-3: Partial Radial Distribution Function (PRDF)

Let $g_{\alpha\beta}(r)$ be the PRDF. Then, the PRDF gives the number of atoms of type β per volume at a distance r within a shell of thickness Δr from an atom of type α.

Unlike TRDF, PRDF explicitly takes types of the particles into consideration. Such consideration enables the detail analysis of the multi-species system and is useful when distribution pattern of one atomic species around another atomic species is more important than general particle distribution. In reality also, such finer analyses result more interesting information about the underlying system than TRDF approach. For example, in Magnesium Silicate system, distribution of Oxygen atoms around Magnesium or Silicon atoms are of paramount importance than distribution of Magnesium atom around Silicon or Magnesium atom around Magnesium itself in explaining various properties of the silicate liquid.

4.1.4 Cutoff Distance

When the RDF is non-zero at a distance, there is some finite probability of finding another atom at that distance. No two atoms can come closer to each other than a certain minimum distance and that minimum distance is guided by the strong repulsive force that gets dominant at smaller distances. So, the cutoff distance gives us the minimum allowable distance between two atoms. This distance varies according to the types of atoms considered.

Definition 4-4: Cutoff Distance

Cutoff Distance (r_{start}) is the distance where a RDF first becomes non-zero, i.e. $g(r) = 0 \; |_{r<r_{start}}$ and for $\Delta r \to 0, g(r) > 0 |_{r=r_{start}+\Delta r}$.

4.1.5 Distance to the First Minimum

In a crystalline solid, all the particles have predetermined fixed positions in the system. RDF for such system shows peaks at fixed distances. RDF is zero for all other distances as there is probability of finding any particle at distances other than to the lattice points. As temperature is increased, RDF shows slightly flattened peaks. But the peaks are still distinct from each other. When temperature is increased further, the peaks become broader and broader and after a certain temperature, the peaks start to merge with each other. The r_{min} value is the distance where first peak and second peaks merge with each other. When we look at all the particles that are within this distance, most of those particles were within the first peak and some of the particles were under the second peak before the two peaks merged. At the same time, some of the particles that were within the first peak move away from the r_{min} position to fall under the second peak. On average, we can still say that the particles within r_{min} distance represent the particles that were originally under the first peak in the crystalline system.

Definition 4-5: Distance to First Minimum

Distance to First Minimum (r_{min}) is the distance where the RDF is minimum after its first peak. Formally, let us say r'_{min} is the set of distances where, i.e. $\frac{dg(r)}{dr} = 0$ and for $\Delta r \to 0$, $g(r \pm \Delta r) \geq g(r)$. Then $r_{min} = min(r'_{min})$.

From a purely statistical perspective, a homogeneous Poisson process has uniform distribution of events in the space. The RDF of such a process is constant and has value unity (‘1’). Any change from constant valued RDF suggests there is certain amount of correlation among the events in the system. When RDF is larger than unity, there is an amount of clustering. When RDF is smaller than unity, there is an amount of repulsion. As can be noticed, directional dependence of the spatial distribution has been ignored when computing RDF. Here, the dataset is assumed to represent a stationary and isotropic system. For a stationary system, probabilistic statements about the system in any region A are invariant under arbitrary translation of A. For an isotropic system, those statements are invariant under arbitrary rotation also [62].

4.2 Coordination Environment

The first large peak in RDFs for a liquid system hints at a short range structure. The RDFs for a liquid system also contain peaks of different heights after the first peak before RDF finally settles around unity. These intermediate peaks hint at some medium range structures. In this section, the concept of Coordination Environment (CE) is explained in detail. It is one of the methods for analyzing the short range structures in the immediate neighborhoods of the particles.

4.2.1 Definition

The coordination environment (CE) of a particle in the system is the distribution of other particles around it. The CE is distinguished by two different criteria: i) by the number of particles in the environment, and ii) by the distribution of types of particles in the environment. The first criterion gives us the total coordination number of the particle and the second criterion gives us the partial coordination number. Total coordination number and partial coordination number will be differentiated later in the chapter.

There are more particles at greater distance from a particle and the particle has larger coordination number (CN). Similarly, there are few particles at smaller distances and the particle has smaller coordination number. Now, we are ready to define the concept of coordination distance.

Definition 4-6: Coordination Distance

Let r be the distance between two particles i and j. If the particles i and j are thought to be coordinated whenever $r \leq r_c$, then r_c is said to be the coordination distance.

CE of a particle *i* is now formally defined as:

Equation 4-1

$$C_i = 4\pi\rho \int_0^{r_c} r^2 g_i(r) dr$$

Here, $g_i(r)$ is the RDF for the particle i. In practice, RDF is not computed for individual particles. It is averaged over number of particles and used to compute average coordination number for the system as:

Equation 4-2

$$C = 4\pi\rho\int_0^{r_C} r^2 g(r)dr$$

Equation 4-2 computes total average coordination number of the system. This equation can be modified to compute total instantaneous coordination number of the system:

Equation 4-3

$$C(t) = 4\pi\rho\int_0^{r_C} r^2 g_t(r)dr$$

In Equation 4-3, $C(t)$ is the total instantaneous coordination number of the system and $g_t(r)$ is the instantaneous TRDF of the system at time *t*.

Similarly, these equations are modified to compute average partial coordination number (APCN) and instantaneous partial coordination number (IPCN) of the system.

Equation 4-4

$$C_{\alpha\beta} = 4\pi\rho x_\beta \int_0^{r_C} r^2 g_{\alpha\beta}(r)dr$$

Equation 4-4 gives APCN of the system for atoms of type β with respect to atoms of type α. x_β is the fraction of atoms of type β in the system with respect to the total number of atoms in the system, i.e.:

Equation 4-5

$$x_\beta = \frac{N_\beta}{N_a}$$

where N_β is the number of atoms of type β and N_a is the total number of atoms in the system.

The cutoff distance, r_c, used for CN computation above is how far we want to go radially away from a particle to compute the CN. There is no theoretical restriction on what shall be the

cut-off distance. But practically, the cut-off distance is computed from TRDF or PRDF depending on whether we want to compute the TCN or the PCN.

There might be occasional confusion between coordination environments and bonding environments. The coordination environment of an atom is more general concept than bonding environment of the atom. The bonding environment is defined for one particular distance, known as bond length, and only for those atomic pairs that can be chemically bonded. On the other hand coordination environment can be defined for any distance and between any types of atomic pairs. It is simply an abstract geometrical concept. If we look at the relationship between the expression for K-function (Equation 3-10) and the expression for the average coordination number (Equation 4-2), we find a relationship between the coordination number and K-function as:

Equation 4-6

$$C(r) = \rho\, K(r)$$

So, even though in case of atomistic datasets, the cutoff distance used for calculating the coordination number has physical significance, the Equation 4-6 shows that coordination function can be analyzed to understand the spatial distribution of the points.

In a multi-species system, coordination environment depends on the considered atomic species and the order in which they are considered. That is, which species becomes the central atom and which atomic species become the surround atoms. For a given material system containing $|\Lambda|$ species, there are $|\Lambda|^2$ distinct atomic coordination environments. Thus coordination is multivariate information, which can be represented by an asymmetric square matrix of order $|\Lambda|$:

Equation 4-7

$$\begin{bmatrix} CN_{11} & CN_{12} & \cdots & CN_{1|\Lambda|} \\ CN_{21} & CN_{22} & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ CN_{|\Lambda|1} & CN_{|\Lambda|2} & \cdots & CN_{|\Lambda||\Lambda|} \end{bmatrix}$$

Here, the diagonal terms represent same species coordination whereas the off-diagonal terms represent different species coordination. For example, $MgSiO_3$ system shows nine different coordination environments with three same species coordination of Mg-Mg, Si-Si and

O-O, and six different species coordination types of Mg-O, Mg-Si, Si-Mg, Si-O, O-Mg and O-Si. Out of these nine coordination environments, only Si-O and O-Si are directly involved in bonding.

For a crystal, the bulk coordination numbers are well defined and their values are the same irrespective of the locations of the atoms of given species involved in coordination under consideration. However, coordination environment is not the same everywhere for a defect crystal of at surface and interfaces. The surface coordination number is always less than the bulk coordination number and is also dependent on which Miller index the surfaces use. In a BCC crystal, the bulk coordination number is 8, whereas for the 100 surface, the surface coordination number is 4. Similarly, the coordination number near the vacancy site or grain boundary may be smaller than the bulk value.

The coordination environment is even more diverse in liquids. In this case coordination is a function of both space and time. For instance, Silicon-Oxygen coordination numbers of all the Silicon atoms in a Magnesium Silicate liquid are not the same. Since both Silicon and Oxygen atoms in the liquid are constantly moving, the coordination number of a given Silicon atom does not need to remain constant with time. In this sense, it is generally preferable to calculate an average value. However, tracking individual coordination states of a given type in space and time gives important insight in to the structural and dynamical behavior of the system under consideration.

For a given time instant, the atomic configuration is composed of various coordination states. It is important to see the changes in these states over the time. We have to take its average behavior to understand the system's properties. We need to extract the multivariate coordination data and visualize them to fully understand the coordination environments. With this background, now coordination environment for an atom, *i*, are categorized into total coordination environment and partial coordination environment.

4.2.2 Total Coordination Environment

Total Coordination Environment (TCE) for an atom *i* is defined as the total number of atoms and the distribution of the atoms (labeled by *j*) that fall within a spherical region of radius r_c with the particle *i* located at the centre of the sphere, i.e.:

Equation 4-8

$$nn_i = \{1 \le j \le n_a : d(i,j) \le r_c\}$$

where r_c is taken to be distance to the first minimum in the $g(r)$.

This information is useful in a monoatomic system or in a system where different types of points are equivalent and interact the same way irrespective of their type information. In general that is not the case. We extend this definition to the case of partial coordination environment for an atom.

4.2.3 Partial Coordination Environment

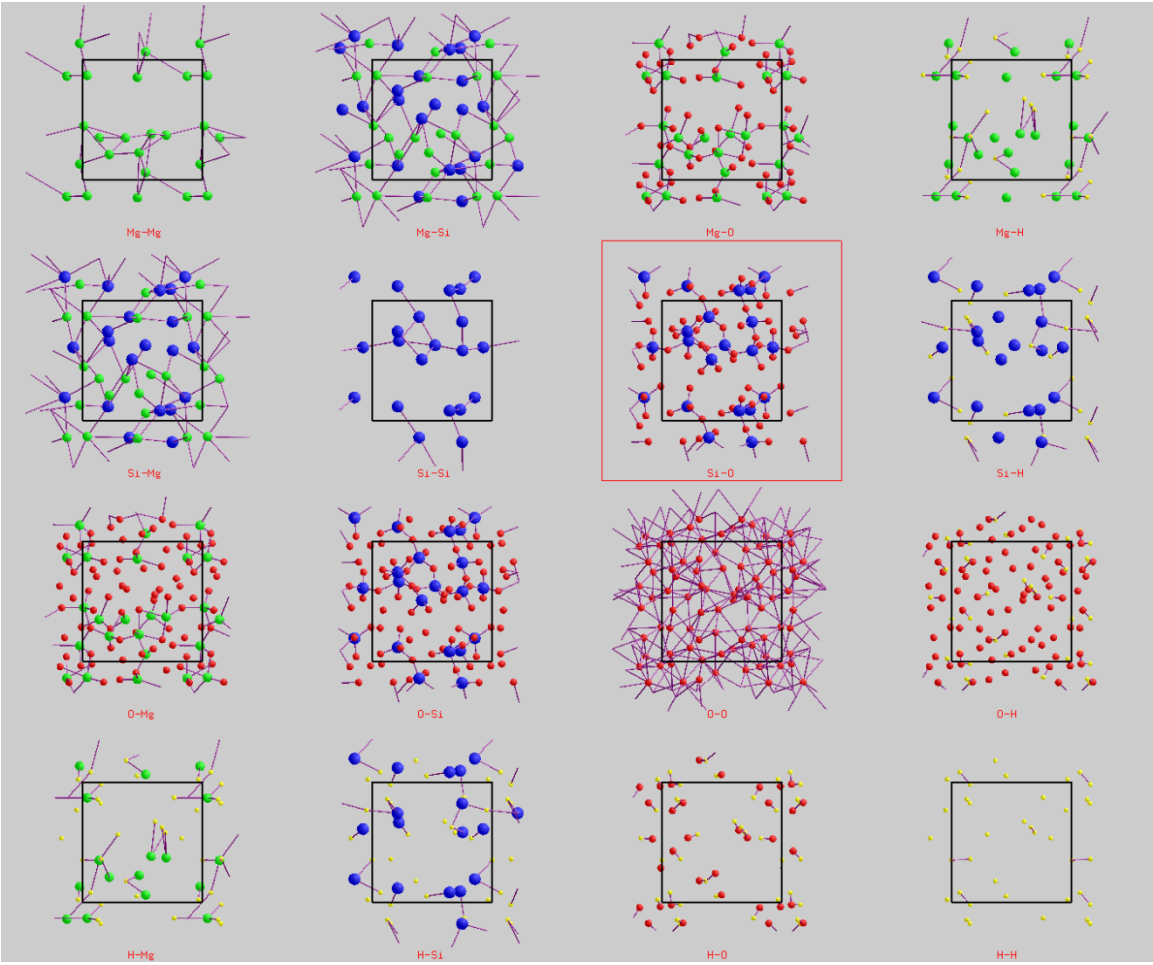

Figure 4-2: Complete set of partial coordination environments of hydrous magnesium silicate.

Suppose the type of an atom *i* is α and we want to define coordination of this atom with respect to atoms of type β. In this case, instead of taking all atoms within the cutoff distance into account, only the atoms of type β are taken into account. Also, r_{min} in the corresponding partial radial distribution function is used as the cutoff distance, $r_{cutoff} = r_{\min}^{\alpha\beta}$. The partial coordination environment is then defined by:

Equation 4-9

$$nn_i^{\alpha\beta} = \left\{1 \le j \le n_a : d(i,j) \le r_{\min}^{\alpha\beta} \wedge type(j) = \beta\right\}$$

For each possible pair of species, a unique partial coordination environment can be defined. For example, when there are four different species there are 16 different partial coordination environments (Figure 4-2).

4.2.4 Stability of Coordination Environment

The stability of a coordination state for a particle *i* is defined as a fraction of time the coordination state exists in the system. CE consists of two components: number of coordinated atoms and the set of coordinated atoms forming the CE. So, there are two kinds of stabilities we want to analyze: stability of the coordination number and stability of the coordinated neighbors. It is necessary to distinguish between these two types of stability because an atom can have same coordination number even when it gets coordinated with different sets of neighbors at different time instants.

Let $c_i(t)$ be the instantaneous coordination number for an atom *i* at time *t*. Over the n_{steps} snapshots, atom *i* is tracked to see for what fraction of the time it is coordinated with $c_i(t)$ atoms. Let this fraction be $f'[c_i(t)]$. Then, $c_i(t)$ and $f'[c_i(t)]$ are visually represented using color and shape of the atom.

For an atom *i*, the set of its nearest neighbors at time *t* is given by $nn_i(t)$. Then, the set of all the atoms that get coordinated to atom *i* during an extended period of *N* steps, is given by:

Equation 4-10

$$NN_i = \bigcup_{j=1}^{j=n_{steps}} nn_i(j\Delta t).$$

Equation 4-10 is used to define Coordination cluster as:

Definition 4-7: Coordination Cluster

Coordination cluster of an atom i is defined as the set of all nearest neighbors over the selected temporal interval (Figure 4-3).

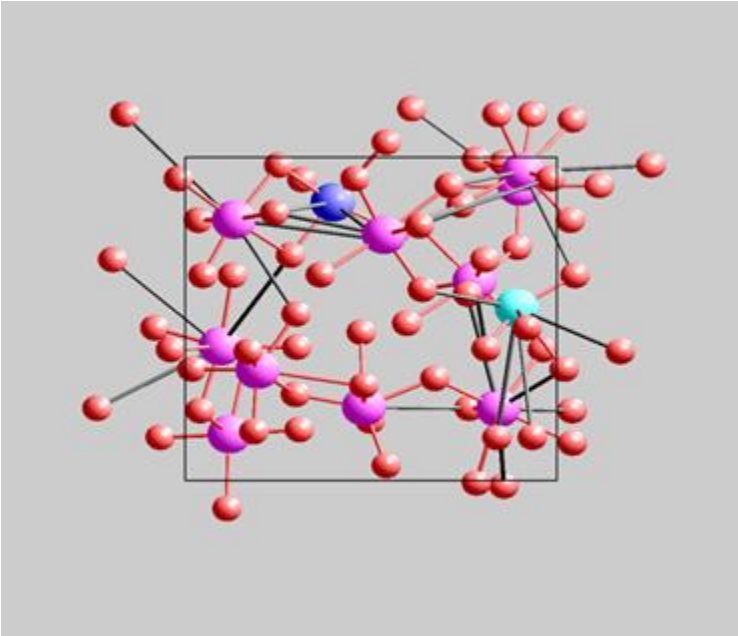

Figure 4-3: Coordination cluster

Let $f_i(k)$, where $k \in NN_i$, is the fraction of time the atom k is coordinated with atom i. Then, when the atom k gets coordinated with atom i, $f_i(k)$ is used to represent the stability of coordination between atoms i and k (Figure 4-4). This concept is extended to include partial coordination environment. In this case, only atoms of type β are taken into account when computing the set of nearest neighbors. Using the type-specific nearest neighbors $nn_i^{\alpha\beta}(t)$, the set of all the atoms of type β that get coordinated to atom i that is of type α, is given by:

Equation 4-11

$$NN_i^{\alpha\beta} = \bigcup_{j=0}^{j=N} nn_i^{\alpha\beta}(j\Delta t)$$

.

In many cases, the local coordination environment can also be expressed as polyhedral units. For instance, four-fold coordination means a tetrahedron with the type α atom being at the centre and surrounding type β atoms at the vertices of the tetrahedron. The stability of a polyhedral surface is not easy to visualize because we are actually trying to visualize the cumulative effect of more than one interatomic distance in one object at the same time. For this purpose, the stability of the surface, s_p, is defined as:

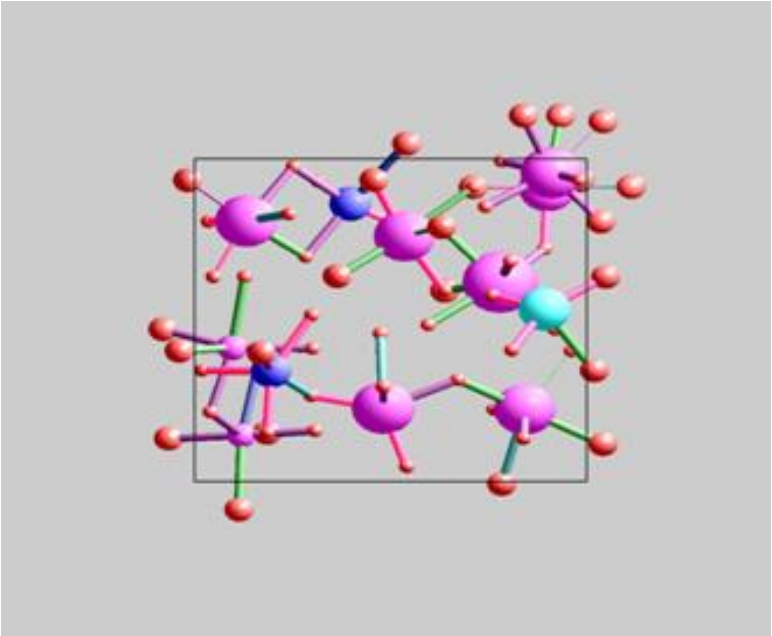

Figure 4-4: Coordination stability and bond stability

Equation 4-12

$$s_p = \prod s_{bond}$$

where s_{bond} is the stability of individual bonds defined as the fraction of the time it exists in the dataset. As each of the bonds has to exist at the same time to form the polyhedral structure, we multiply stability of individual bonds to compute the stability of the polyhedron as a whole (Figure 4-5). One way to represent the polyhedral stability is to modulate the polyhedral surface

itself using s_p as the modulating quantity. Such modulation of shape tells us about the overall stability of the polyhedral object as a whole, but it does not tell us why the object is stable or unstable. Also, as only size variation is quantitative [3], change in shape will not be able to convey the quantitative information that we want to convey. This also does not tell us anything about which atom(s) is (are) more likely to detach from the surface or which atoms are likely to be bound with the surface. So, instead of modulating shape, we modulate the size of the coordinated atom to visualize the probability of their coordination (Figure 4-5). The probability of coordination of atom *k* with another atom *i* is given by $f_i(k)$.

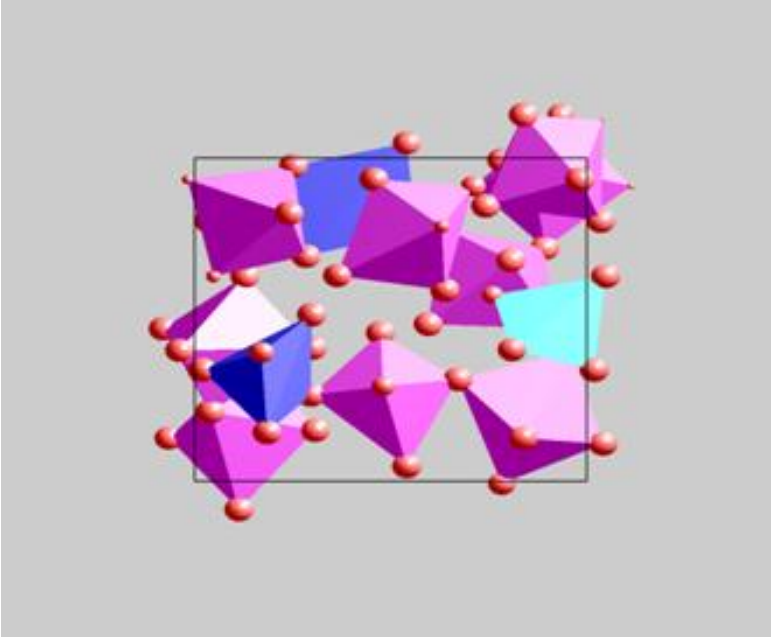

Figure 4-5: Polyhedral representation of coordination environment, corresponding polyhedral and bond stability

4.3 Cluster Structure

The CE describes the structure in the immediate neighborhood of an atom and is valuable in explaining different characteristics of a system. In this section, two kinds of cluster structures, nearest neighbor cluster and common neighbor cluster, are explained. The nearest neighbor (NN) clusters are useful in describing the structures existing in neighborhood farther away from the immediate neighborhood of an atom. The common neighbor (CN) clusters are useful in describing the neighbor around a pair of atoms.

4.3.1 Nearest Neighbor Cluster

NN clusters are useful in looking at the longer range structures than coordination environments. Before we define NN cluster, we need to define a nearest neighbor. Formal definition of the nearest neighbor is:

Definition 4-8: Nearest Neighbor

An atom j is said to be a nearest neighbor of another atom i, if and only if $d(i,j) \leq r_c$, where r_c is the coordination distance.

Now, we define NN cluster as:

Definition 4-9: Nearest Neighbor Cluster

The nearest neighbor cluster for the atom i is defined as the set of atoms that are reachable from the atom i by using coordination distance, r_c, and their coordination relationship with each other. NN cluster is analyzed by selecting $r_c = r_{min}$. As it has already been mentioned, the coordination distance is not restricted to this particular value. The critical distance can be varied to look at the distance dependence of the NN cluster in a system.

4.3.1.1 Types of Nearest Neighbor Clusters

NN cluster can be computed in multiple ways, for example, considering same species type, considering multiple species types and considering all the atoms as simple points without their species information. In the first case, the cluster shows the framework formed by that species in the system and cluster's dynamics shows the dynamics of the structural framework formed by that species. In the second case, the cluster shows the structure formed by collection of coordination environments among the selected set of species. Finally, in the last case, the atoms are separated from their species information and are treated simply as particles in three-dimensional space. The cluster formed in this way is usually not useful except in the elemental systems.

4.3.2 Common Neighbor Cluster

CN cluster analysis is another important approach to classify inherent local structures through direct interpretation of the characteristic shape of RDF in terms of characteristic short-

order range. Given a pair of atoms, CN is the set of all the atoms that are nearest neighbors of both the atoms in the pair. This method is able to decompose the first and second peaks of the RDF by characterizing local environment surrounding each atomic pair that contributes to the peak of the RDF in terms of the number and properties of common nearest neighbors of the pair under consideration. It can be used to identify atoms in particular environment such as FCC (Face Centered Cubic), HCP (Hexagonal Close-Packed), BCC (Body Centered Cubic) or icosahedral. Different types of pairs are associated with different types of local order. For example, 2421 pair is characteristic of FCC. The numbering scheme is explained later in the section.

Definition 4-10: Common Neighbor

Let i and j are two atoms. An atom k is said to be a common neighbor to i and j, if $d(i,k) \leq r_c$ and $d(j,k) \leq r_c$, where r_c is the coordination distance. Or in other words, the atom k is a common neighbor to atoms i and j if it is a nearest neighbor of both atom i and atom j.

Definition 4-11: Common Neighbor Cluster

For a pair of atoms i and j, let $cn_{ij} = \{k \mid d(i,k) \leq r_c \wedge d(j,k) \leq r_c\}$ be the set of common neighbors. Then, $cn \cup \{i,j\}$ along with all the coordination relationship among these atoms within the critical distance r_c is called the common neighbor cluster for atoms i and j.

CN cluster for a pair of vertices in a graph can be defined as the subgraph induced by the neighbors common to both of those vertices. From this perspective nearest neighbor cluster is union of the coordination environments of the nearest neighbors whereas common neighbor cluster is the intersection of the coordination environments of the pair of vertices.

CN cluster also represents a way of decomposing the RDF according to the environment of the pairs. A PRDF describes local environments of a pair of species of the atoms. We analyze the local structure of different pairs of atoms using different RDFs. For each atomic pair falling under a given cutoff window, a set of four indices αjkl are obtained to specify the local environment of the pair. The first index α indicates whether or not the pair of atom are nearest-neighbors of each other i.e. coordinated with each other. It is 1 if the pair falls under the first peak and the atoms are referred to as bonded pairs, and is 2 if the pair falls under the second

peak. The second index *j* is the number of nearest-neighbors common to both the atoms. The third index *k* is the number of bonds between these common neighbors. The fourth index *l* is the number of bonds in the longest continuous chain formed by the *k* bonds between common neighbors. This last index did not have a systematic definition and the notation for the CN pairs was slightly different in other previous studies. The main purpose of the fourth index is to resolve the ambiguity associated with the arrangement of the bonds. Once each pair has been assigned one of the *αjkl* types, the distribution distances can be calculated in the usual way and a radial distribution function for each type of pairs obtained. These will be referred to as the CN components, $g_{\alpha jkl}(r)$, of the RDF and are normalized in such a way that the sum of these components gives the total RDF. This method can thus be used to interpret various features in RDF.

For a multi-component system, CNA becomes more complicated because many different types of atomic pairs have to be taken into consideration. A cutoff is found for each of the PRDFs. For a binary system like Magnesium Oxide, the CN components can be labeled as $g_{MgMg\alpha jkl}(r)$, $g_{OO\alpha jkl}(r)$, $g_{MgO\alpha jkl}(r)$. For the FCC-structured Magnesium Oxide, no common neighbor clusters exist under the first peak of Mg-O RDF where as 1421 pair type exists for Mg-Mg RDF.

4.4 Ring Structure

There are different definitions of rings available in the literatures [68-70] and results about ring statistics differ according to the definitions [71-73]. The ring statistics have been used to analyze the structural properties of different atomic systems [73-75] and software systems are also available for offline ring analysis [76, 77]. Also, there are debates on what constitutes a ring in a system and how to characterize them [68, 69]. Statistics about the ring-size and distribution of rings of different sizes are thought to be useful in describing the bulk-level properties of an atomic system [71, 78].

Amorphous solids and liquid systems lack in long range structure and the coordination environments are analyzed to understand their short range structures. Zachariasen, was the first to propose that atoms in glasses must form frameworks that were non-periodic and isotropic with energy comparable to or greater than that for crystals while trying to reconcile difference in properties among crystals and glasses [78]. The atoms in the crystals are arranged in a regular

and periodic fashion, whereas the atoms in the glasses and the liquids are arranged in an irregular and non-periodic fashion. The three-dimensional arrangements of the rings formed by the atoms in glasses and liquids are thought to be useful in describing the topology of those systems.

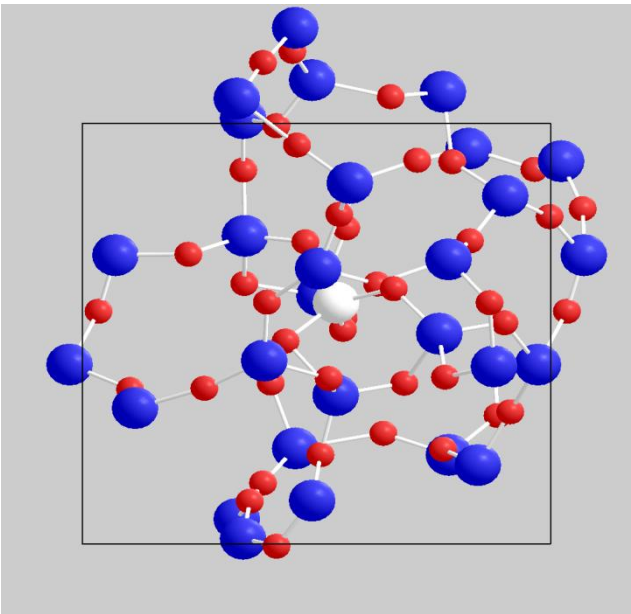

Figure 4-6: Ring structure

Finding all the rings in a system is computationally intractable [79] because the general problem of ring computation contains Hamiltonian cycle problem as a special case, and there are possibly exponential number of rings in a system [68] and brute force technique is infeasible. For these reasons, the maximum ring size is restricted to a physically relevant size instead of trying to compute all the rings. Such restriction makes algorithm suitable for interactive visualization possible. Also, in case of material systems, only the *primitive rings* are of interest. This further reduces the time complexity of the problem because large rings are not usually primitive.

Now, before going into the details of local ring structure, some terms are formalized below. Let us suppose $G = \{V, E\}$ is the graph with vertex set V and edge set E. Let us also define an index set, N.

Definition 4-12: Path

Let us suppose there are two vertices x and y in V and there are sequence of vertices, $p = v_1, v_2, \ldots, v_n$ such that $(v_i\ v_{i+1}) \in E$ for all $1 \leq i \leq n-1$, $x = v_1$, and $y = v_n$, then p is a path between vertices x and y. Length of the path is the number of edges in the path.

Definition 4-13: Cycle

Let c is a path between two vertices x and y, then c is a cycle if vertices x and y are the same. Length of the cycle is the number of edges in the cycle.

Definition 4-14: Simple Cycle

If every vertex in the cycle except one vertex appears exactly once and the one that appears more than once appears exactly twice, then the cycle is called a simple cycle.

From the definition of a simple cycle, we see that there are two distinct paths between any two vertices in the cycle and length of these paths are not necessarily equal. We are interested in the shortest path among such multiple paths. The shortest path between two vertices can be defined with respect to the graph, G, or with respect to a cycle. Now, primitive ring is defined as:

Definition 4-15: Primitive Ring

Let r is a simple cycle and $v_i, v_j \in r$. If the shortest path between v_i and v_j with respect to the simple cycle r is not greater than the shortest path between those two vertices with respect to the graph G, then the cycle r is a primitive ring.

This definition makes sure there are no shortcuts between any two vertices in a cycle. Now, we are ready to define local ring structure for a vertex, say v_i, as:

Definition 4-16: Local Ring Structure

Local ring structure, $R(v_i)$, for a vertex v_i is a set of primitive rings, r such that $v_i \in r$. Formally:

$$R(v_i) = \{r \mid v_i \in r\}$$

This is not a standard definition but this is the typical definition used to calculate ring statistics in a material system and this definition is used.

Chapter 5 Dynamical Analysis

Dynamical analysis is done to explore the data at different time scales. The spatio-temporal analyses are performed by combining such dynamical analyses with the structural analysis to visualize the data at different space and time scales. For instance, we may want to know how and to what extent the constituent atoms in the given system move. Normally, atomic positions are animated, or alternatively, trajectories of the atoms in the system are displayed completely. While such representations do not discard any information, the uncorrelated and highly entangled trajectories are difficult to interpret. There are better ways of understanding the dynamics. Various displacement information are extracted to understand the diffusive nature of the atoms and anisotropy of their diffusion. Quantification and visualization of such displacements contain information about the diffusive nature of the underlying system.

5.1 Transport Coefficients

Diffusion process can be characterized by calculating diffusion coefficient as follows:

Equation 5-1

$$D = \lim_{t \to \infty} \frac{\langle [r(t)]^2 \rangle}{6t}$$

where

Equation 5-2

$$\langle [r(t)]^2 \rangle = \frac{1}{n} \sum_{i=1}^{i=n} \left| \overrightarrow{r_i(t + \Delta t)} - \overrightarrow{r_i(t)} \right|^2$$

is the root mean square displacement (RMSD), and $\overrightarrow{r_i(t)}$ and $\overrightarrow{r_i(t + \Delta t)}$ are the positions of i^{th} atom at the start and end of the time interval t.

The partial MSD is then calculated by averaging over atoms of a given species. Translational symmetry of the system is accounted for so that the MSD value is not restricted by

the size of the supercell. Note that the MSD for a given time t can also be calculated by averaging over time origins (t_0).

The time-correlation functions are used to analyze the dynamical properties of the system [66, 80]. Since MD simulation allows direct access to the complete trajectories of each atom in the system, computation of time-correlation functions is also straightforward. The integrals of the time-correlation functions are used to compute transport coefficients of the system such as diffusion coefficient.

5.2 Trajectories

The set of positions an atom occupies is the trajectory of that atom. In a solid system, an atom remains close to its original lattice position. Each atom remains within a well-defined region around itself and the trajectories of different atoms do not overlap. In a liquid system on the other hand, atoms move far from their mean positions resulting dispersed and overlapping trajectories. We can find out more about the system by analyzing the trajectories of individual atoms. As an atom moves within the supercell satisfying the periodic boundary conditions, its path is traced using a color-coded line. The pixel color at a given position for a given atom is varied according to the elapsed time or the distance traveled by the atom relative to some reference position.

5.3 Atomic Displacements

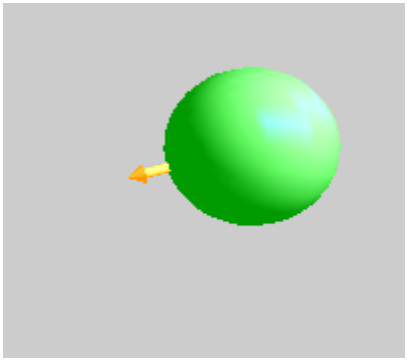

Figure 5-1: Instantaneous diffusion sphere

Various sets of displacement data are generated to characterize the extent and pattern of the atomic movement during simulation. In general, displacements are defined as:

Equation 5-3

$$\Delta r_i = r_i^t - r_i^0, \text{ for } i = 1, 2, \ldots, n_a$$

Here, r_i^t and r_i^0 are the positions of i^{th} atom in a given configuration at a particular time and a reference configuration respectively. The data may represent differences of a given atomic configuration (r_i^t) from the initial ($r_i^{t=0}$) or previous ($r_i^{t-\Delta t}$) or next ($r_i^{t+\Delta t}$) configuration. The reference configuration can be the crystalline configuration (r_i^{crys}) as well.

The other relevant differences are those involving the mean positions,

Equation 5-4

$$c_i = \frac{1}{n_{steps}} \sum_{j=1}^{n_{steps}} p_i^j$$

The mean positions are taken over the selected set of snapshots, where n_{steps} is the number of selected steps, p_i^j is position of i^{th} atom at j^{th} simulation step. All these data represent discrete vector data, which are rendered using spheres and lines. Magnitude of the vector is represented by the size of the sphere placed at each atomic site (p_i) in 3D space:

Equation 5-5

$$r_i^s = f\left|\Delta r_i\right| + a,$$

where f is the scaling factor and a is the minimum radius given to each sphere. The direction is represented by a line segment that points away from the surface of the sphere:

Equation 5-6

$$line\left(|\Delta r_i| p_i, |\Delta r_i| p_i + l \frac{\Delta r_i}{|\Delta r_i|}\right),$$

where Δr_i is the orientation vector, p_i is the center of the sphere, and l is the length of the line. In the case, where we are interested at the farthest positions the atoms reach during the

whole simulation period from the mean positions, the centroid spheres are drawn at c_i's with the radii defined by:

Equation 5-7

$$r_i = \max_{j=1:n_{steps}} \left\{ d(c_i, p_i^j) \right\}$$

Here, $d(c_i, p_i^j)$ is the distance between c_i and p_i^j.

5.4 Principal Component Analysis

The positional set for the individual atoms describes their diffusion and movements completely. The displacement analysis approach (Section 5.3) gives the extent of movement of individual atom. But such displacement analysis fails to convey the anisotropy of the atomic movement. The atoms do not move equally in each direction. There are preferred directions of movement for each atom. Principal component analysis (PCA) [81] is used to analyze the trajectory of each atom to find out more about those preferred directions of movement.

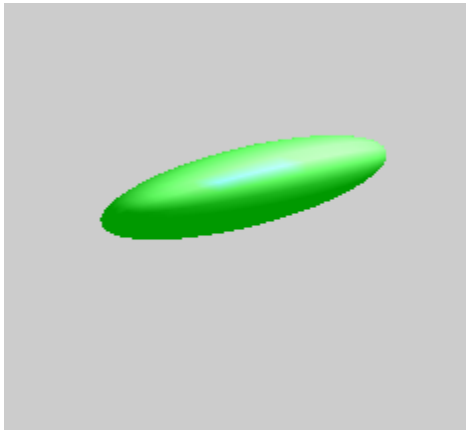

Figure 5-2: Diffusion ellipsoid

Traditional PCA assumes data points are independent and with no obvious correlation among each other. Time series data do not satisfy this condition. Data points closer in time and space are highly correlated by virtue of their closeness. For this reason, PCA is not usable for such datasets. But PCA is very rarely used just as an inferential tool and by relaxing the requirement of probabilistic independence among the data points, PCA provides valuable descriptive information about the data points [81]. Main thrust of PCA is to reduce the

dimensionality of a dataset that consists of a large number of interrelated variables and retain as much as possible of the variations present in the dataset. We also want to find out the direction and amount of maximum drift for an atom from the tangle of trajectories and see if there are any systematic patterns in their movements.

One frequently used method of computing PCA is by using covariance matrix as the starting point. Let us suppose the population covariance matrix for each atom, *i*, for the given dataset is given by Σ_i.

Equation 5-8

$$\Sigma_i = \begin{pmatrix} \sigma_i^{xx} & \sigma_i^{xy} & \sigma_i^{xz} \\ \sigma_i^{yx} & \sigma_i^{yy} & \sigma_i^{yz} \\ \sigma_i^{zx} & \sigma_i^{zy} & \sigma_i^{zz} \end{pmatrix}$$

Population covariance matrix can be estimated by using the sample covariance matrix, S_i, with n data points as [82]:

Equation 5-9

$$\frac{n}{n-1} S_i$$

This is commonly known as an unbiased estimator. The diagonal terms of the covariance matrix are the variance along three orthonormal axes, *x-axis, y-axis* and *z-axis*, and off-diagonal terms are the covariance between the two axes on the superscript. To compute the covariance matrix, mean position of the particle is computed as,

Equation 5-10

$$\vec{\mu}_i = \frac{\sum_{j=1}^{j=N} \vec{p}_i(j\Delta t)}{N}$$

and covariance and variances of the positional components are then computed as,

Equation 5-11

$$\sigma_i^{kl} = \frac{\sum_{j=1}^{j=N} \left(p_i^k(j\Delta t) - \mu_i^k\right)\left(p_i^l(j\Delta t) - \mu_i^l\right)}{N},$$

where *k* and *l* are two of the three positional components, *x, y, and z*. We see covariance matrix is a symmetric matrix. The eigenvalues and eigenvectors of the covariance matrix give the amount and direction of the maximum variation in the data. For a positive semi-definite and symmetric matrix, eigenvalues are real. A square matrix of size three has three eigenvalues and corresponding three eigenvectors. A natural representation for the eigenvalues and eigenvectors for each atomic movement is an ellipsoid. Let the three eigenvalues and corresponding three eigenvectors be λ_1, λ_2, *and* λ_3, and v_1, v_2, *and* v_3, respectively. Let us define a principal component matrix, $\mathcal{A}$, as:

Equation 5-12

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} \lambda_1 v_1^1 & \lambda_1 v_1^2 & \lambda_1 v_1^3 \\ \lambda_2 v_2^1 & \lambda_2 v_2^2 & \lambda_2 v_2^3 \\ \lambda_3 v_3^1 & \lambda_3 v_3^3 & \lambda_3 v_3^3 \end{pmatrix}$$

Now, let's transform the position set, $\boldsymbol{p_i}(\boldsymbol{t})$, by $\boldsymbol{\mathcal{A}}$ as:

Equation 5-13

$$\boldsymbol{p_i'}(\boldsymbol{t}) = \boldsymbol{\mathcal{A}}\boldsymbol{p_i}(\boldsymbol{t})$$

The transformed positions, $\boldsymbol{p_i'}(\boldsymbol{t})$, is defined with respect to the frame reference specified by the three orthogonal eigenvectors. These positions have the maximum variation along the $\boldsymbol{v_1}$ axis and successively decreasing variations along $\boldsymbol{v_2}$ and $\boldsymbol{v_3}$ axes.

We use λ_1, λ_2, *and* λ_3 as the major, medium and minor axes of the ellipsoid and use eigenvectors, $\boldsymbol{v_1}$, $\boldsymbol{v_2}$ *and* $\boldsymbol{v_3}$ to construct a rotation matrix to orient the ellipsoid. Principal components are uncorrelated with each other, so they form an orthogonal frame of reference. When the set of eigenvectors are not unit vectors, their values are normalized to provide an orthonormal frame of reference for the ellipsoids.

Chapter 6 Algorithms and Data Structures

In this chapter, algorithm, pseudocode and data structure are explained from implementation perspective. Following three datasets are used for the performance evaluation.

Table 6-1: Descriptions of the Sample Datasets

Dataset	D1	D2	D3
n_a	72	168	84
$\|\Lambda\|$	2	4	4
n_{steps}	7000	7000	7000
a (10^{-10} m)	[10.3, 10.3, 10.3]	[12.7, 12.7, 12.7]	[10.11, 10.11, 10.11]

6.1 Analytical and Exploratory Algorithms

In this section, the analytical and exploratory methods are described from the implementation perspective. The pseudocodes and performance information for individual methods are also given.

6.1.1 Radial Distribution Function (RDF)

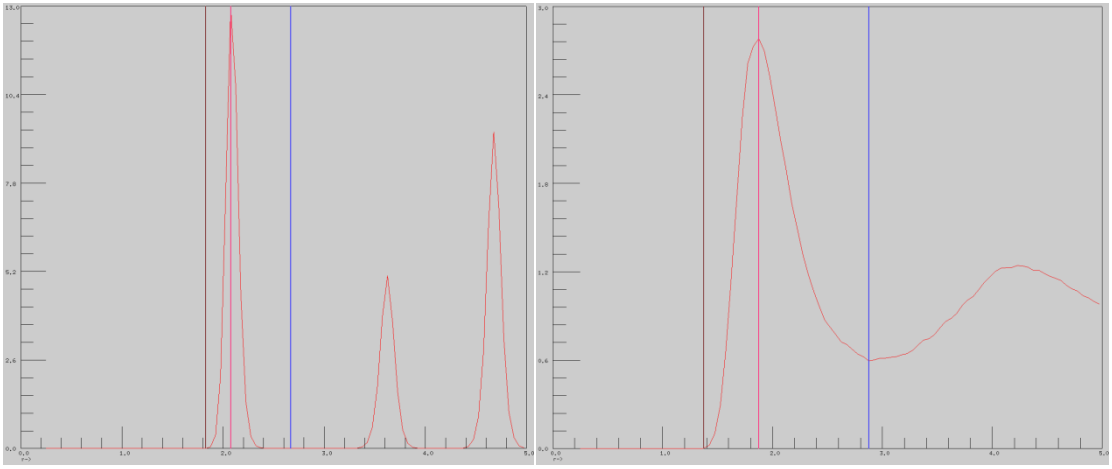

Figure 6-1: RDF plots for crystalline (left) and liquid (right) systems

RDF is the most important quantity computed and is also the computationally most intensive quantity required. RDF describes radial distribution of atoms with respect to each other. The RDF is computed by constructing a series of concentric spheres set at a fixed distance

(Δr) apart around each of the atoms in the system considered in turn and counting the number of atoms found in each shell. Figure 6-2 shows concentric circles for illustration purpose. In the figure, we are computing RDF for atom *i.* At a radius *r*, we draw a spherical shell of thickness *Δr*.

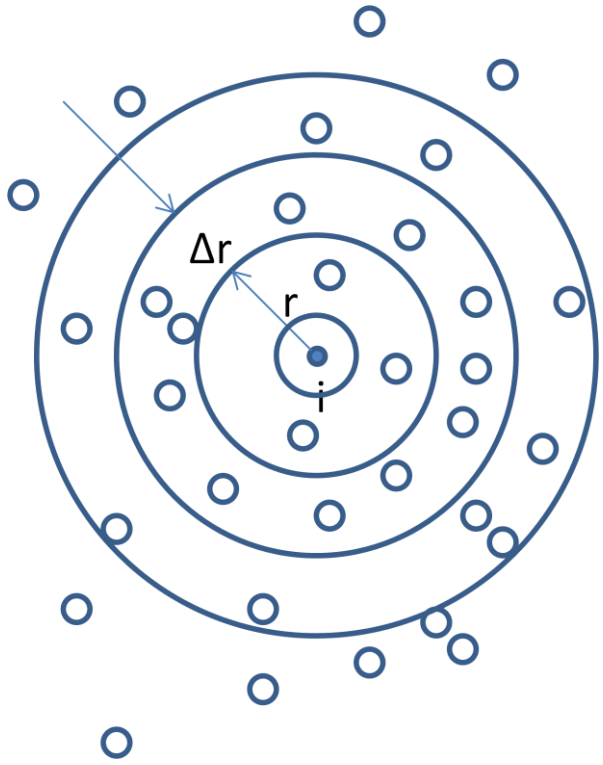

Figure 6-2: Computation of RDF using concentric spheres

In an uncorrelated system, where there is no structure, the expected number of atoms at a distance, *r,* is the average number of atoms per unit volume ($\rho = \frac{N}{V}$) times the volume of the shell of radius r and width Δr ($4\pi r^2 \Delta r$). As r increases, the expected number increases proportionately to the volume of the spherical shell. To quantify correlations, the observed number of atoms in the shell of radius *r* is normalized by the number that would be observed if the atoms were uncorrelated. Thus, the RDF can be expressed as [83]:

Equation 6-1

$$g(r) = \frac{n(r)}{4\pi\rho r^2 \Delta r}$$

Here, $n(r)$ is the mean number of atoms in a shell of radius r and thickness Δr. To formulate an equation for $n(r)$, let us first define an indicator function for a predicate p, $I(p)$, as:

Equation 6-2

$$I(p) = \begin{cases} 1, if\ p\ is\ true \\ 0, if\ p\ is\ false \end{cases}$$

Now, $n(r)$ is defined as:

Equation 6-3

$$n(r) = \frac{1}{n_a} \sum_{i=1}^{i=n_a} \sum_{\substack{j=1 \\ j \neq i}}^{j=n_a} I(r \leq d(p_i, p_j) \leq r + \Delta r)$$

$d(p_i, p_j)$ is a distance function that computes Euclidean distance between two points.

All atoms in the system can be treated similarly, leading to an improved determination of RDF as an average over many atoms. Furthermore, the RDF is also averaged over an extended simulation period by replacing $n(r)$ in Equation 6-1 with its time-averaged counterpart as:

Equation 6-4

$$\widehat{n(r)} = \frac{\int_0^\tau n_t(r) dt}{\int_0^\tau dt}$$

Here, $n_t(r)$ is time-indexed version of $n(r)$.

As the RDF is probabilistic by nature, the system has to be simulated for a sufficiently long time for the RDF to be physically significant. Otherwise RDF fluctuates rapidly and the following characteristics and descriptions may not be applicable. Also as the system is assumed to be ergodic, time average distribution also represents the ensemble average of similar systems.

Liquid has strong short range order and lacks long range structures. Two atoms too far from each other are statistically uncorrelated. Due to strong short range order, there are peaks in RDF at short distances and the RDF stabilizes around unity at large distances. In crystalline solids, there exists strong long range structure also, but the structure is periodic. Strong short

range order and lack of long range order in liquid and strong long range periodic structure in crystalline solids enable us to use a distance, d_{max} during computation. Two atoms farther than this distance are assumed to be statistically uncorrelated and are not used for the computation.

Algorithm 1: Radial Distribution Function

Input:
d_{max}: maximum distance upto which RDF is to be computed
n_{bins}: number of bins
V: volume
N: number of atoms in the volume V
$\Delta r = \frac{d_{max}}{n_{bins}}$
$\forall i \in N$
    $\forall j \in N \; and \; i != j$
        if $d(i,j) < d_{max}$ then
            $index = \left\lceil \frac{d}{\Delta r} \right\rceil$
            $rdf_{index} = rdf_{index} + 1$
        endif
for $index = 1 \; to \; n_{bins}$
    $rdf_{index} = rdf_{index} \times \frac{V}{4\pi r^2 n_a^2 \Delta r}$
endfor

The maximum distance is then discretized into number of grids, n_{bins}. The RDF is computed by counting the atoms falling in individual bins. Once all the atoms have been taken into account, then the array is normalized to compute RDF.

Let $\Delta r = d_{max}/n_{bins}$, then all the atoms at a distance $0 < r \leq \Delta r$, are put in bin_0, atoms at a distance $\Delta r < r \leq 2\Delta r$ are put in bin_1. In general, atoms at a distance $i*\Delta r < r < (i+1)*\Delta r$ are put in bin_i. In pseudocode, the bin is the array rdf and atoms farther than d_{max} are discarded. The pseudocode shown does not average over multiple time steps and also does not take atomic species into account and only computes the instantaneous TRDF.

Extending this algorithm to calculate time averaged RDF is straight forward, but extending this pseudocode to calculate PRDFs is not so straight forward.

Let us suppose we want to calculate PRDF for two species α and β. Then, for all the particles of species α, we have to compute distances to all the atoms of species β. Let's say there are n_α and n_β particles of species α and species β respectively. Also, N_α and N_β are the index sets for species α and species β. Then the above pseudocode is modified as follows to calculate the PRDF for β species with respect to α species.

Algorithm 2: Pseudocode for compute partial radial distribution function

```
Input:
d_max : maximum distance upto which PRDF is to be computed
n_bins : number of bins
V: volume
N: number of atoms in the volume V
Δr = d_max / n_bins
∀i ∈ N_α
    ∀j ∈ N_β
        if d(i,j) < d_max then
            index = ⌈d/Δr⌉
            prdf_index = prdf_index + 1
        endif
for index = 1 to n_bins
    prdf_index = prdf_index × V / (4πr²n_α n_β Δr)
endfor
```

The normalizing factor in the case of the partial RDFs is same as the one given in the unbiased estimator for K-function in Equation 3-15. When the two species are same, then the normalizing factor becomes same as the one for total RDF. Also, if there is only one species in the system, the PRDF and TRDF become the same.

6.1.1.1 Performance Evaluation

Table 6-2: Time to compute the RDFs (in seconds)

n_{steps}	d_{max} = 5.0 A						d_{max}=10.0 A					
	n_{divs}=100			n_{divs}=200			n_{divs}=100			n_{divs}=200		
	D1	D2	D3	D1	D2	D3	D1	D2	D3	D1	D2	D3
1	0.00	0.02	0.02	0.00	0.02	0.02	0.00	0.03	0.02	0.00	0.03	0.00
100	0.36	1.64	0.44	0.34	1.72	0.47	0.42	1.89	0.55	0.42	1.90	0.58
500	1.55	8.53	2.23	1.55	8.50	2.14	1.99	9.83	2.58	1.95	9.51	4.80
1000	3.23	15.97	4.44	3.03	17.45	4.18	3.91	18.56	5.19	3.88	19.28	5.39
2000	6.45	34.02	8.86	6.44	33.94	8.95	7.78	38.06	10.39	7.77	38.98	10.38
3000	9.59	51.00	13.19	9.69	51.00	13.12	11.63	58.39	15.78	11.16	55.22	18.88
5000	15.94	85.00	21.98	16.03	84.94	21.95	19.39	98.61	26.09	18.34	97.27	27.24
7000	22.48	111.86	28.80	22.78	118.61	30.75	27.09	133.36	39.27	25.86	136.41	38.11

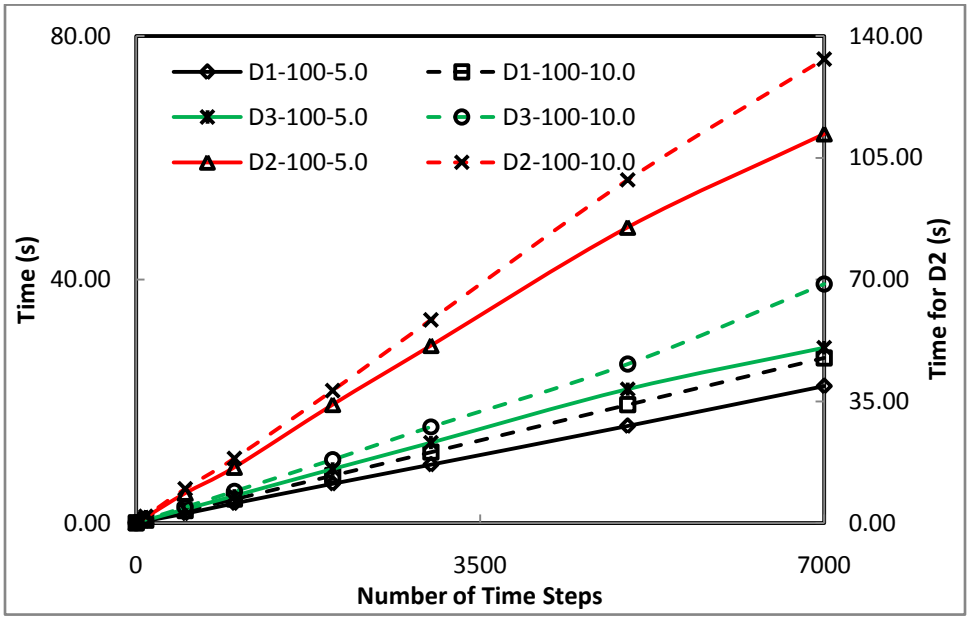

Figure 6-3: Performance chart for rdf computation with 100 bins

The performance data are collected for three different datasets, D1, D2, and D3. RDF computation depends upon the number of time steps, n_{steps}, number of atoms, n_a, number of bins, n_{divs}, and maximum distance, d_{max}, used. The chart shows two separate groups. The lower group of data corresponds to D1 and D3, and the upper group of data corresponds to D2. D1 contains 75 atoms of 2 atomic species, D3 contains 84 atoms of 4 atomic species and D2 contains 168 atoms of 4 atomic species.

The effect of the number of species in the system on computation time of the radial distribution function is negligible. On the other hand, the computation time is proportional to the square of the number of atoms and is linearly proportional with the number of time steps used for the computation. The computation time is also linearly proportional with the number of bins and the maximum distance.

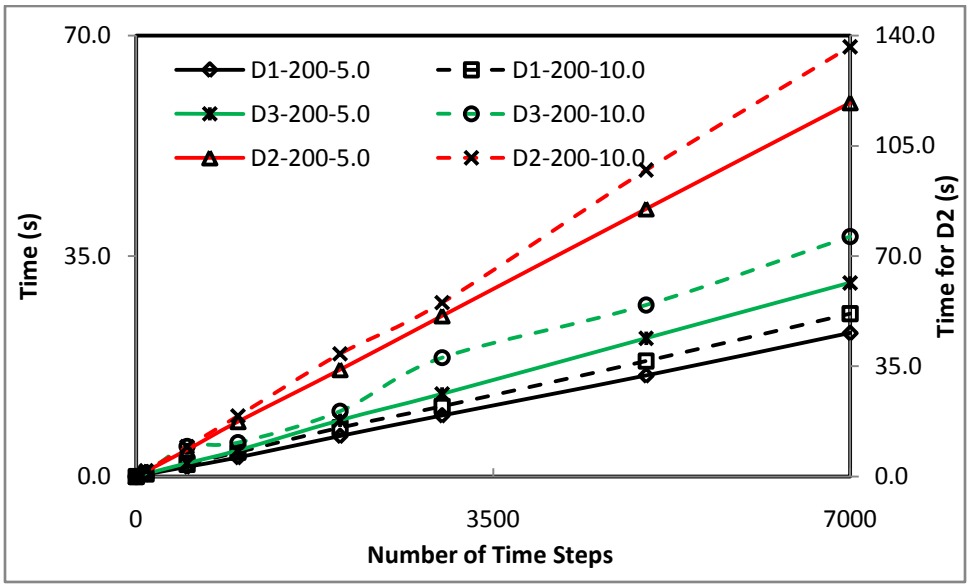

Figure 6-4: Performance chart for rdf computation with 200 bins

The performance of the RDF computation degrades slightly with the number of bins used. In case of D1 for $d_{max} = 5.0$, the computation time increased from 22.48 seconds to 22.78 seconds (1.33% increase) when the number of bins was increased from 100 to 200 for 7000 time steps. Similarly, for $d_{max} = 10.0$, the computation increased from 133.36 seconds to 136.41 seconds (2.3% increase) for D2 when the number of bins was increased from 100 to 200. On the other hand, the increase is substantial when the maximum distance is increased. The computation time increased from 30.75 seconds to 38.11 seconds (23.9% increase) when the cutoff distance was increased from 5.0 to 10.0 for D3 with 200 bins and 7000 time steps.

6.1.2 Coordination Environment

The coordination environment of an atom is the distribution of other atoms around this atom within a coordination distance, r_c. The algorithms for representing the CE are presented in this section. CE is computed using the original positional dataset and the coordination distance.

Algorithm 3: Pseudocode for Coordination Environment Computation

```
Input:
r_c: Coordination Distance
i : Atom Id
Output:
ce : Set of coordinated atoms
Pseudocode:
for all atoms a != i
begin:
     if distance(i, a) < rc then
          ce.add(a)
     endif
end
return ce
```

The set of coordinated atoms, `ce`, are within the coordination distance from the atom `i`. Two different schemes are used to visualize `ce` for the atom `i`: i) as a color coded sphere, and ii) color coded coordination polyhedron. When the CE is visualized using a color-coded sphere, the atom is rendered using a color selected from a color map. In this case, the color represents the size of `ce,` that is the coordination number. When the coordination environment is visualized using a polyhedron, the atoms in the set form the vertices of the polyhedron and the polyhedron is computed as the convex hull of the atoms in the set. The polyhedron is colored just like color-coded coordination sphere by using the color-map. Only some coordination environments are visualized as polyhedra. For example in Magnesium Silicate systems, Silicon-Oxygen coordination environments are usually visualized as polyhedra. Such polyhedral are called coordination polyhedra. The coordination polyhedra are computed using QuickHull algorithm [84] as the convex hull of the positions of the coordinated atoms. Convex hull of a set of points is the smallest convex set that contains the points.

Algorithm 4: Pseudocode for QuickHull algorithm [84]

```
Create a simplex of d+1 points
For each facet F
     For each unassigned point p
          If p is above F
               Assign p to F's outside set
For each face F with a non-empty outside set
     Select the furthest point p of F's outside set
     Initialize the visible set V to F
     For all unvisited neighbors N of facets in V
          If p is above N
               Add N to V
     The boundary of V is the set of horizon ridges H
     For each ridge R in H
          Create a new facet from R and p
          Link the new facet to its neighbors
     For each new facet F'
          For each unassigned point q in an outside set of a
facet in V
               If q is above F'
                    Assign q to F''s outside set
     Delete the facets in V
```

QuickHull algorithm uses the "Beneath Beyond" randomized algorithm given by following algorithm [originally in [85] cited from [84]].

Let H be a convex hull in R^d, and let p be a point in $R^d - H$. Then F is a facet of conv(p $\cup$ H) if and only if

(1) F is a facet of H, and p is below F; or

(2) F is not a facet of H, and its vertices are p and the vertices of a ridge of H with one incident facet below p and the other incident facet above p.

For polyhedral surface representation of the local structure around an atom of a given type, e.g. Silicon, we first find all the atoms of another type, e.g. Oxygen, within the specified coordination distance. The positions of those coordinated atoms are then passed to the QHull library. The library computes a convex-hull for the positions and returns a set of triangle describing the convex hull. This hull is the coordination polyhedron. The colors are assigned to a polyhedron for easy classification of the coordination environment according to the number of vertices in the polyhedron. The pseudocode for computation of coordination polyhedron is given below:

Algorithm 5: Pseudocode for Coordination Polyhedra Computation

```
Input:
i : Atom Id to compute coordination polyhedra for
r_c: Coordination Distance
Output:
P : Coordination Polyhedron
Pseudocode:
Begin:
     ce = ComputeCoordinationEnvironment(i, r_c)
//compute polyhedron using the coordination environment
     P = qhull(ce)
     return P
End
```

The cation coordination polyhedron such as SiO_n, where n is the number of Oxygen atoms coordinated to a Silicon atom, can be regarded as the fundamental unit in characterizing the atomic structure in response to temperature, pressure and composition [86]. There are five regular polyhedra commonly known as Platonic solids: i) Tetrahedron, ii) Cube, iii) Octahedron, iv) Dodecahedron, and v) Icosahedron. All of these polyhedral solids have fixed lengths and angles with respect to the center. In a distorted system, polyhedral are not regular and are distorted from their regular counterparts. The degree of polyhedral distortion is characterized by using the following two parameters [87, 88]:

Equation 6-5: Quadratic Elongation

$$\lambda_P = \sum_{i=1}^{n} \left(l_i / l_0\right)^2 / n,$$

Equation 6-6: Bond Angle Variance

$$\sigma = \sum_{i=1}^{n} \left(\theta_i - \theta_0\right)^2 / (n-1).$$

Here, l_0 and θ_0 are bond lengths and angles in regular polyhedra. l_0 is computed as the average of all the bond-lengths forming a polyhedron. We assume that the average bond-length closely approximates the actual bond length in the undistorted polyhedron. In case of bond angle variance, there are fixed angles for regular polyhedra. For example, in tetrahedral coordination, n = 4 and θ_0 = 109.47 °, and in octahedral coordination, n = 6 and θ_0 = 90 °. In case of irregular polyhedra, there are no standard bond-angles. Liquid systems are full of such polyhedra. There is no known way of calculating the bond-angle variance for liquid systems.

6.1.2.1.1 Mixed Coordination Environment

The CEs are crucial in understanding various structural properties. It is imperative for a visualization system to allow visualization of coordination environments among multiple pairs of species to provide context into their environments. This raises certain implementation issues. Let's suppose there are $|\Lambda|$ species in the system. Then, there are $|\Lambda|^2$ possible pairs of species. It is already known from the previous discussion that coordination environments among different pairs of species are different in general. Table below shows a possible scenario.

Individual cell of the table specifies coordination environment of the species defined by the row with respect to the species defined by the column. When multiple coordination

environments are visualized, only one coordination environment from a row can be visualized. One row corresponds to one type of species and for an atom of certain species, only one coordination environment can be encoded in the color-coded sphere. Converse of this is multiple coordination environments from one column can be selected for simultaneous visualization. So, theoretically $|\Lambda|$ different coordination environments can be visualized simultaneously restricted only by our ability to perceive so much information in one image (Figure 6-5).

Table 6-3: Coordination Environment Matrix

S1 / S2	1	2	3	4	…	$\|\Lambda\|$
1	ce_{11}	ce_{12}	ce_{13}	ce_{14}	…	$ce_{1\|\Lambda\|}$
2	ce_{21}	ce_{22}	ce_{23}	ce_{24}	…	$ce_{2\|\Lambda\|}$
3	ce_{31}	ce_{32}	ce_{33}	ce_{34}	…	$ce_{3\|\Lambda\|}$
4	ce_{41}	ce_{42}	ce_{43}	ce_{44}	…	$ce_{4\|\Lambda\|}$
…	…	…	…	…	…	…
$\|\Lambda\|$	ce_{11}	$ce_{\|\Lambda\|2}$	$ce_{\|\Lambda\|3}$	$ce_{\|\Lambda\|4}$	…	$ce_{\|\Lambda\|\|\Lambda\|}$

This condition is modeled using a directed graph, *D* (Figure 6-6). Vertices of the graph are the atomic species in the system and the directed edges are the coordination environments between the corresponding pairs of species. For example, let's suppose there is an edge, *e*, in the directed graph, *D*, between vertices representing species pair, S_1 *and* S_2. Then, this edge represents the coordination environment of species s_1 with respect to species s_2. This graph can also contain loop in it when coordination environment of a species with respect to the same species is selected. For the graph to be of a practical value, each edge will have only one outgoing edge and can have many incoming edges. In Figure 6-6, each vertex has one outgoing edge, so this is a valid graph for coordination environment representation.

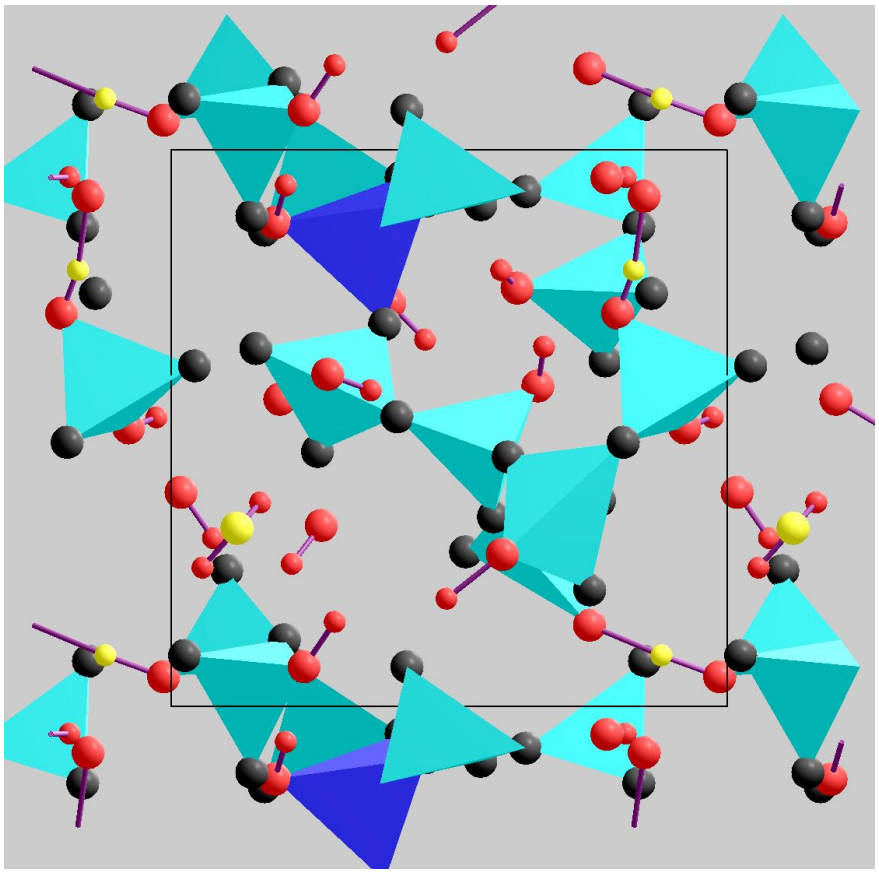

Figure 6-5: Mixed coordination environment

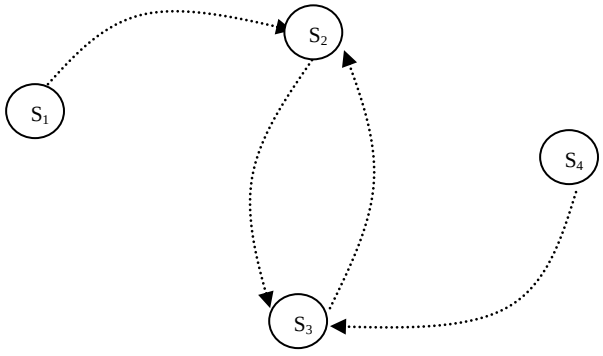

Figure 6-6: Coordination environment modeled as a directed graph

6.1.2.2 Performance Evaluation

The cutoff distances for each species pair is selected from the respective PRDFs and is used to compute the coordination environments, coordination stability and bond stability. The computation time is linearly dependent with the number of time steps. Number of species in a dataset also plays a greater role in the computation time than it did in case of RDF computation shown by greater separation between the lines for D1 and D3 in the performance chart.

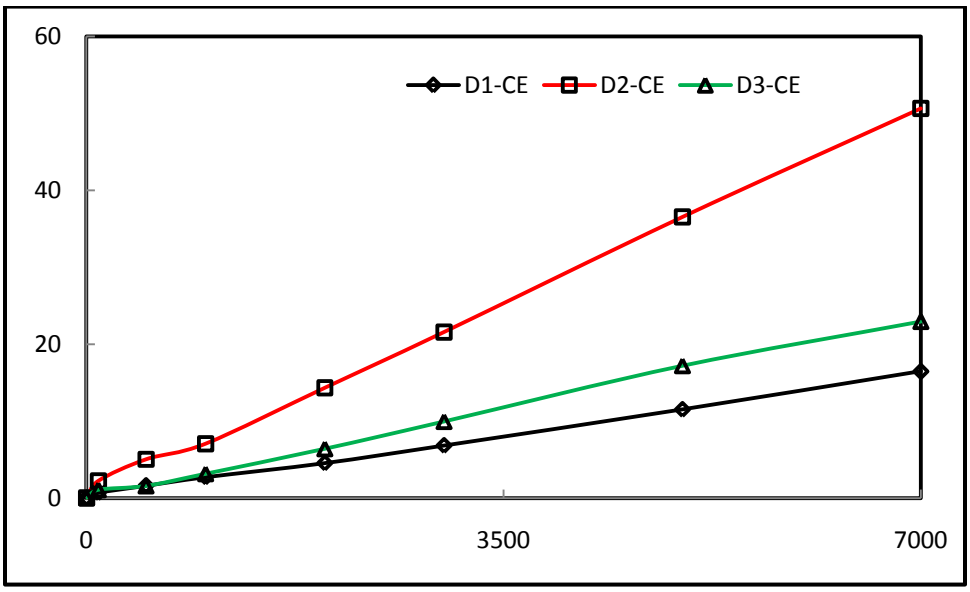

Figure 6-7: Performance of coordination environment computation

6.1.3 Common Neighbor Cluster

A graph data structure is constructed where each atom is represented as a vertex. If the distance between two vertices is less than cutoff distance, an edge is constructed. CN cluster is computed using the graph.

6.1.3.1 CNC Algorithm

The algorithm (Algorithm 6) looks at only those pairs of atoms that actually have at least one nearest neighbor atom in common. Suppose V and E are vertex set and edge set for a graph, $G(V, E)$. Formally, let's say $V = \{ i \mid 1 \leq i \leq v_{max} \}$ and $E = \{(i,j) \mid i,j \in V\}$ and we want to compute the CNC for this graph. V is the vertex set and E is the edges in the graph.

Observation 6-1

If the shortest distance between two vertices is more than two, then they cannot have any neighbors common with each other.

Observation 6-2

Nearest neighbor relation in an undirected graph is symmetric. If vertex *i* is a nearest neighbor of vertex *j*, then vertex *j* is a nearest neighbor of vertex *i* as well.

Observation 6-3

For a vertex i, if n_i is the set of its nearest neighbors, then there are $\frac{|n_i|(|n_i|-1)}{2}$ different pairs of vertices that have the vertex i as their common neighbor.

Algorithm 6: Pseudocode for Common Neighbor Cluster computation

```
Input:
g:Adjacency list representation of graph
Pseudocode:
Step 1: Find all the pairs with at least one common neighbor
For all the atoms, k
     For all the nearest neighbors, m of k
          If m is marked with q as its nearest neighbor, then
               Insert m as common neighbor between pairs k and
          q
          End If
     Mark m with k
End For
Step 2:Find subgraph spanned by the set of common neighbors
For all the pairs, p, with at least one common neighbor
     Find subgraph V of g induced by p
     (V, p) is the common neighbor cluster formed by p.
End For
```

First observation is obvious from the definition of the common neighbors and it obviates the need to look at the vertices pairs that are farther than two edges apart from each other and

thus simplifies the algorithm. Second observation simply says that it is enough to calculate cluster for a pair (i, j) rather than calculating for both pairs (i, j) and (j, i) reducing some computation. Figure 6-8 illustrates the third observation. Any pair of nearest neighbors of the vertex i will have the vertex i as its common neighbor. Also, the third observation lets us pair up a new nearest neighbor of a vertex with the already found nearest neighbors.

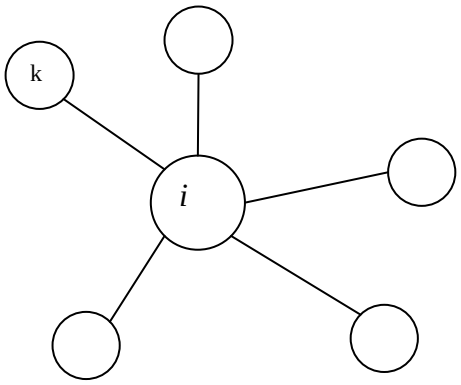

Figure 6-8: A vertex and its nearest neighbors

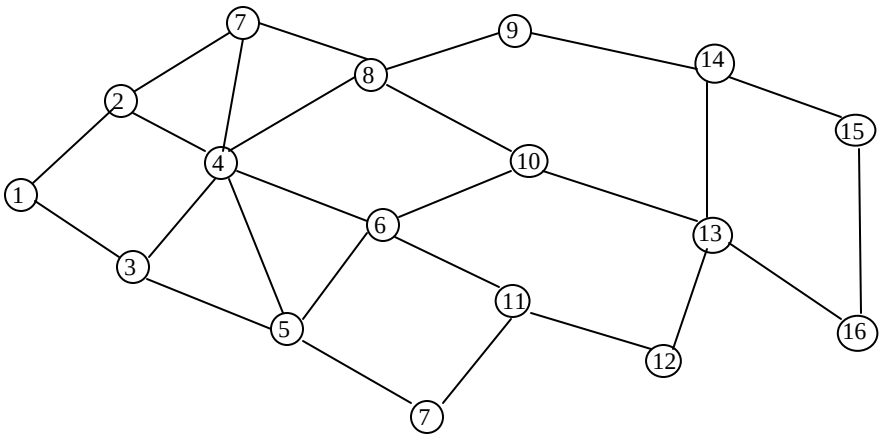

Figure 6-9: A simple graph

For illustration, let's suppose we have a graph as shown in Figure 6-9 and we want to compute the cluster formed by each pair of vertices in the graph.

Let's start from the vertex *1*. The algorithm keeps track of nearest neighbors for each vertex that are encountered so far. The vertex *1* has two nearest neighbors, the vertex *2* and the vertex *3*. This information is saved in the nearest neighbor list for both vertices *2* and *3*. From the vertex *1* we go to the vertex *2*. It has three nearest neighbors, *1, 4* and *7*. *2* is added in the nearest neighbor lists of *1*, *4* and *7*. At this stage, nearest neighbor lists of the vertices *1*, *2*, *3*, *4* and *7* have one neighbor in each of them. Now we move to the vertex *3*. It has three nearest neighbors, the vertices *1*, *4* and *5*. Here, we find the common neighbors for a pair of vertices *2* and *3*. Before adding *3* to the nearest neighbor list of *1* and *4*, we see that the nearest neighbor lists for those vertices are non-empty that means we already know some of the nearest neighbors for *1* and *4*. From this it is easily seen that the vertex *3* and the nearest neighbors of *1* and *4* have some common neighbors. The vertex *1* has *2* in its nearest neighbor list and the vertex *4* also has *2* in its nearest neighbor list. So, the vertex pair, *2* and *3*, has two vertices, *1* and *4*, in common. This information is saved and then the vertex *3* is inserted in the nearest neighbor list of *1* and *4*. We have not found any nearest neighbors of *5*, so we just add *3* in its nearest neighbor list. We now move to the vertex *4*. It has six nearest neighbors and some of the nearest neighbors for three of the vertices *2*, *3* and *7*. The vertices *2* and *3* both have the vertex *1* as their nearest neighbor. So, the vertex-pair, *1* and *4*, has the vertices *2* and *3* as their common neighbors. Similarly, the vertex-pair, *3* and *4*, has the vertex *5* as a common neighbor. Another vertex-pair, *2* and *4*, has the vertex *7* as a common neighbor. Once we are done with computing common neighbors, the nearest neighbor lists of each of the nearest neighbors of the vertex *4* are updated.

The graph is completely traversed in this way. At the end of the traversal, we have information about the common neighbors between all the pair of vertices not farther than two edges. On the second stage of the algorithm, subgraph induced by the common neighbors is found. The subgraph characterizes the common neighbor cluster. The subgraph and the corresponding vertex-pair is the common neighbor cluster.

In the Algorithm 6, Step 1 finds all the pairs of atoms having at least one neighbor common to each other. The algorithm goes through the adjacency list of every vertex, v. For all the vertices, i, in the adjacency list, it checks whether we already know some of its other nearest

neighbors. If we do, then we have found a common neighbor to some pairs of vertices and this vertex is added to the common neighbor of each of those pairs. At the end, the vertex, v, is added the nearest neighbor list of vertex, i. At the end we have common neighbor lists for all pairs that have at least one neighbor common between them. The structure of the subgraph spanned by the set of common neighbor determines the structure of common neighbor cluster. Step 2 then finds the subgraph spanned by the set of common neighbors. The pair of atom with the subgraph spanned by the set of common neighbors is the common neighbor cluster.

6.1.3.2 Data Structure

The CN clusters are stored in a separate data structure so that the clusters can be explored according to different criteria such as the number of common neighbors and the number of bonds between the common neighbors.

```
struct ClusterList{
public:
      unsigned int numCommonNeighbors;
      ClusterInfoList *cHead;
      struct ClusterList *next;
};

struct ClusterInfoList {
public:
      unsigned int numBonds;
      ClusterInfo *sib;
      struct ClusterInfoList *next;
};

struct ClusterInfo{
public:
      unsigned int startAtom, endAtom;
      bool areBonded;
      unsigned int numCommonNeighbors, numBonds;
      vector<unsigned int> listCN;
      bool *cluster;
      struct ClusterInfo *next;
};
```

`ClusterList` holds the information about the number of common neighbors in the cluster. As more than one pair can have same number of common neighbors and all the cluster having same number of common neighbors are added to the `ClusterInfoList`. `ClusterInfoList` holds the information about the number of bonds in the cluster among the

common neighbors. All the clusters having same number of bonds among the common neighbors are added to `ClusterInfo`. `ClusterInfo` contains the detail information about the cluster. It contains the pair of atoms (`startAtom` and `endAtom`), number of common neighbors between them (`numCommonNeighbors`), number of bonds among the common neighbors (`numBonds`), list of common neighbors (`listCN`) and the actual cluster formed by the common neighbors (`cluster`). The `cluster` simply stores the information about the connection among the common neighbors. If two common neighbors are bonded, then the corresponding entry in the `cluster` is '`true`', otherwise the entry is '`false`'.

6.1.3.3 Performance Evaluation

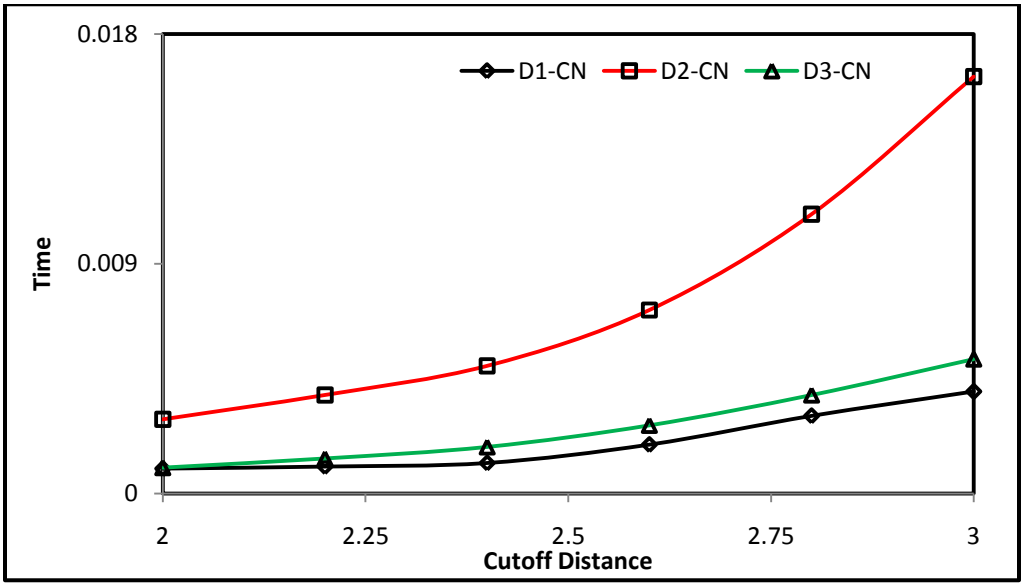

Figure 6-10: Performance of CN algorithm

The performance of the CN algorithm is calculated in terms of the cutoff distance used. The computation time increases quadratically with the cutoff distance used (Figure 6-10). The coordination number of an atom increases quadratically with the cutoff distance and such increase in the coordination number results quadratic increase in the number of edges in the graph. We believe the quadratic increase in computation time comes due to such relationship between the cutoff distance and coordination number.

The reported times include time to construct and free the graph and also the time to construct and free the CN cluster data structure. The cluster data structure does not store the

position information about the pair of atoms and the common neighbors to reduce the duplication of data because the position can be obtained from the original positional dataset.

6.1.4 Nearest Neighbor Cluster

NNC is constructed in breadth first manner for the selected snapshot. Starting from a specified atom, we visit all the atoms that are reachable from this atom (Figure 6-11). The NN cluster is visualized as the new atoms are found instead of first saving the cluster separately. The cluster can also be computed using depth-first search.

Algorithm 7: Pseudocode for Nearest Neighbor Cluster Computation

```
Input:

g:Graph

id: NNC is to be computed for this atom

q: A queue for bread-first search through the graph

Pseudocode:

  1. clear(mark)
  2. mark(id)
  3. enqueue(q, id)
  4. while not empty(q)
       a. A = dequeue(q)
       b. for all k ϵ nearest-neighbors of A in g
            i.  nnc.insert(k)
           ii.  if not marked(k)
          iii.  mark(k)
           iv.  enqueue(q, k)
            v.  endif
       c. end for
  5. end while
```

The white spheres in the figure are the atoms for which the cluster was computed. The number of distinct clusters for a selected cutoff distance can be found by counting the number of white spheres in the Figure 6-11. In this particular system, there is only one cluster when r_{min} from Si-O RDF is selected as the cutoff distance and there are 39 clusters when r_{peak} was used as the cutoff.

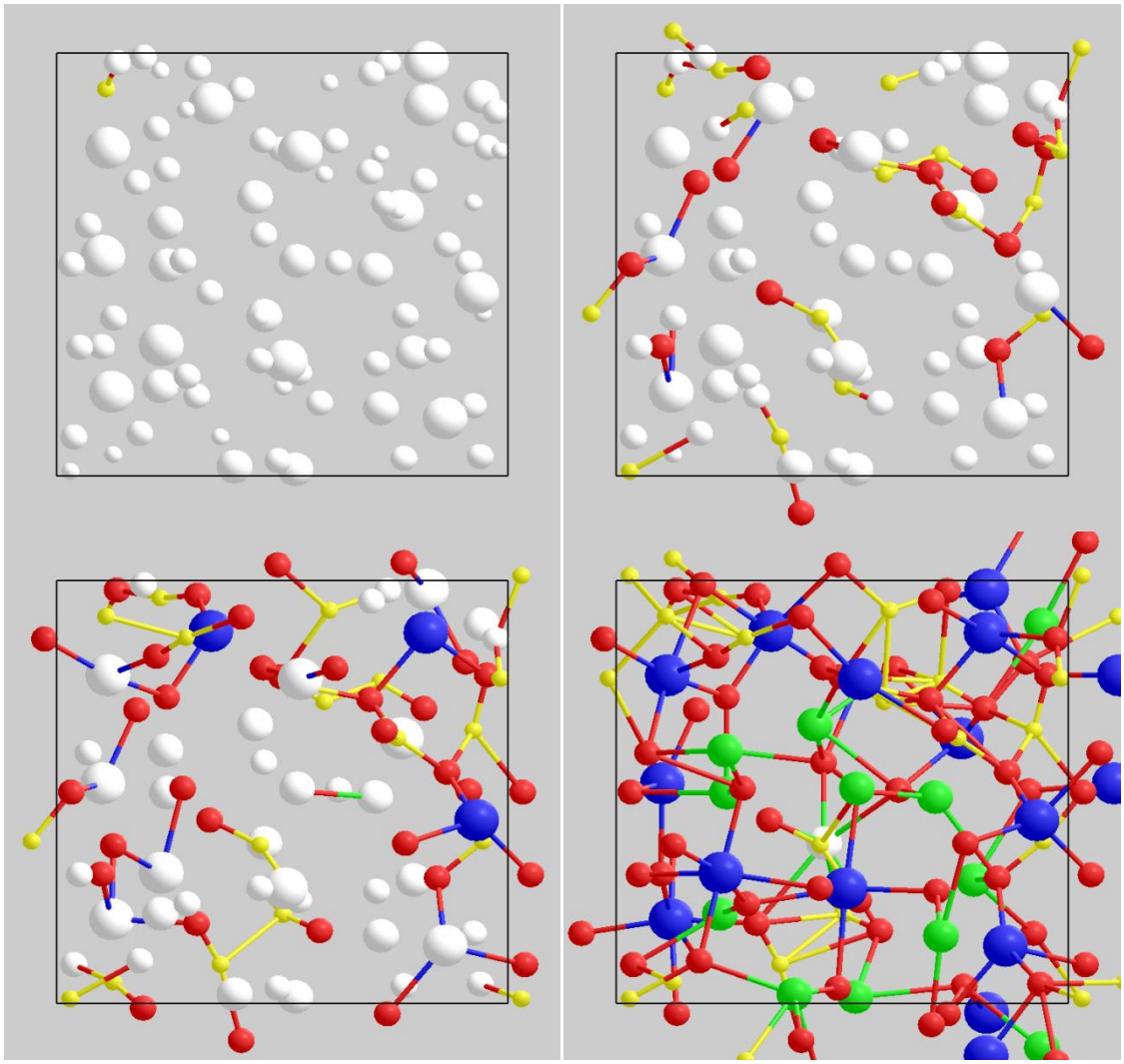

Figure 6-11: Nearest neighbor clusters for different cut-off distances (clockwise from left top) r_{start}, r_{avg}, r_{peak}, r_{min}

6.1.4.1 Performance Evaluation

Performance of NNC algorithm shows that it can compute the cluster in interactive time for a system with size up to 216 atoms. The performance measurement is done with respect to the coordination distance used rather than the size of the vertices and edges in a graph. The

number of vertices is constant for a dataset and the number of edges in the graph is directly proportional to the coordination distance. As we discussed in CN case, the number edges vary with the square of the cutoff distance. The performance chart shows an increasing trend with the increase in coordination distance and the number of vertices and the trend seems to follow the quadratic relationship.

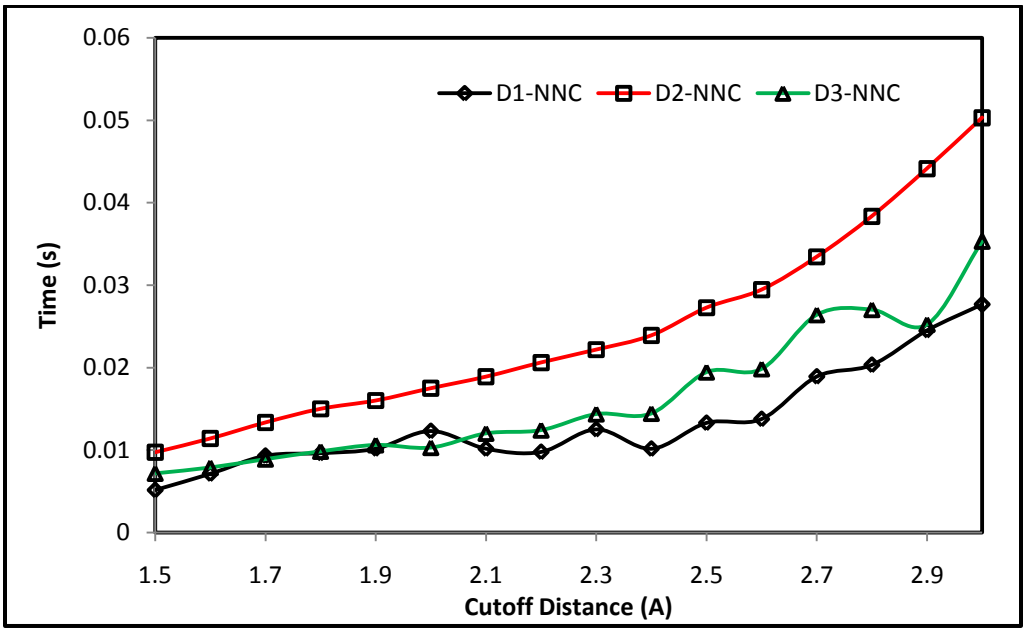

Figure 6-12: Performance of NNC algorithm

6.1.5 Ring Structure

Since the general problem of determining all the rings in a graph contains Hamiltonian cycle-problem as a special case, the general problem is of computing ring structure is computationally intractable. But it is unlikely larger rings are primitive because the large rings usually have shortcuts. This observation is utilized to use the ring computation algorithm in the visualization system.

We need atomic id, i, and a critical distance, r_c, for the ring computation. Instead of all the atoms, only one atom is used for the ring analysis because as the performance studies show, computing primitive rings for all the atoms takes unreasonably long for interactive visualization. The critical distance is selected based either on TRDF or if the rings are to be computed between a pair of species, say α and β, then from PRDF for corresponding pair of species.

Each ring is stored as a list of atom indices and positions. The large rings are rarely primitive and they only increase the computational time. The computational cost is reduced by limiting the size of the system. The system is replicated to find the largest possible rings. This process makes multiple copies of an atom and assigns a unique id to each atom. The number of times the system is replicated is fixed and the algorithm tries to find ring up to the maximum size in the replicated system. An atom has unique position in the original system whereas in the replicated system, the same atom will be located in multiple positions. Thus rings are kept as lists of atoms arranged in parent-child relationship where each atom is accompanied by its position in the replicated system and its original atomic id. The atomic ids in the original system have to be mapped from the replicated ids. This can be done using the following procedure.

Let r_i be a new index for an atom i in the replicated system and rf is the replication factor. All three parameters are integer parameters. Function to convert original index to set of replicated indices, R_i, is:

Equation 6-7

$$R_i = \{ rf \times i + j \mid 0 \leq j < rf \}$$

and

Equation 6-8

$$r_i \in R_i.$$

So, one index is replicated to rf unique indices. The atomic id i in the original atomic system is computed from r_i as:

Equation 6-9

$$i = \frac{r_i}{rf}$$

The division is an integer division and i is the atomic id of the atom in the original atomic system before replication.

6.1.5.1 Ring Structure Algorithm

A separate logical cluster data structure (Figure 6-13) for each atom in the system is also constructed from the graph of the complete system before proceeding with the ring computation [71]. The cluster data structure for an atom stores the information about all the atoms reachable at different levels from the atom.

The data structure has two separate components: the graph and the shortest distance maps among all the vertices in the graph. The graph is stored in an adjacency list representation for reduced memory consumption. The shortest distance map is computed using Dijkstra's shortest distance algorithm for undirected graphs. We first give a brute force approach to the ring computation (Algorithm 8).

Other algorithms reported in [68-70] that are more efficient than the above algorithm. Even though the basic concept are same in the brute force and these improved algorithms, efficient usages of the shortest distance map, they give better performance.

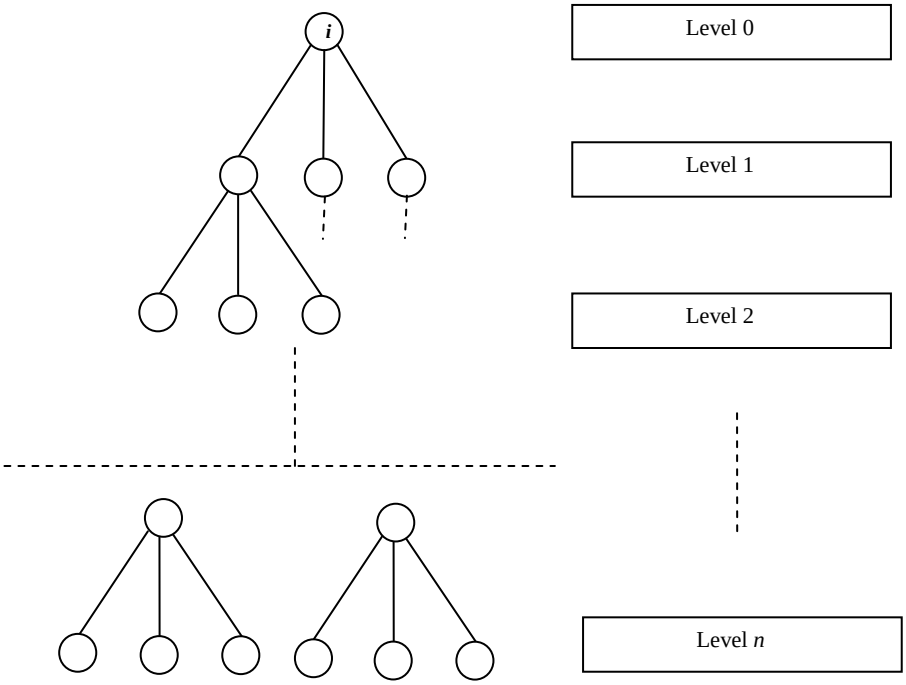

Figure 6-13: Cluster data structure for a vertex *i*

Algorithm 8: Pseudocode for brute force ring structure algorithm

```
Input:
id: rings formed by the cluster of id is to be computed
s: Shortest distance map for all the atoms
n: maximum size of the ring
Output:
rings: rings of sizes up to n
Pseudocode
1. for i = 2 to n
     a. find all the atoms, l_i, in c_id at level-i
     b. for all atoms in l_i
          i. find all pairs of atoms, (r, s) such that either r
             and s are at the same level or s is one of the
             children of r.
     c. endfor
     d. for all pairs (r, s)
          i. r1 = all the atoms found by following parent atoms
             in the cluster structure starting from r
         ii. r2 = all the atoms found by following parent atoms
             in the cluster structure starting from s
        iii. ring = reverse(r1) ∪ s ∪ r2
         iv. if (primitive_ring(ring)) then
                1. rings.add(ring)
          v. endif
     e. endfor
2. next i
3. return rings
```

Main idea behind this algorithm is that whenever there in one vertex at the same level in two different places in the cluster structure, there must be two unique paths to that vertex starting from the root. A ring can then be formed by joining those two paths together. For large rings, there are many different paths between the vertex and the root. When there are n different paths,

then any pair of paths can make a ring. So there are $\binom{n}{2}$ different rings possible. All the rings formed this way may not satisfy the primitiveness criterion. All the rings are checked for their primitiveness. By definition, a ring is primitive if another shorter path between any two vertices in the ring other than the shortest path in the ring itself between those two vertices does not exist. The number of possible paths in a graph increases exponentially with the ring size. Large number of rings have to be checked for their primitiveness when the ring size is large. The computational cost of checking whether a ring is primitive or not is reduced by limiting the number of pair of vertices checked.

We use the mid-node theorem [68] that states:

For any primitive even $2n$-ring, between any check-node and its mid-node, the two equal paths with n links along the ring must be the shortest.

The following routine checks the primitiveness of a ring formed by two paths, `r1` and `r2`.

```
bool MinimalRing(vector<int> r1, vector<int> r2) {
  if(r1.size() != r2.size()) return false;
  for(size_t i = 1, j = 1; i < r1.size()-1 && j < r2.size()-1;
++i,  ++j)
    if(r1[i] == r2[j]) return false;
  int dist = r1.size() - 1;
  for(size_t i = 0, j = r2.size()-1; i < r1.size() && j >= 0;
++i, --j)
    if(shortestPaths[r1[i]][r2[j]] < dist)
      return false;
  return true;
}
```

Due to the exponential increase in the number of possible paths and in the size of the cluster structure itself, the brute force algorithm becomes computationally infeasible. At each level, there will be exponential number of atoms and with large rings, information that need to be stored there also becomes exponential. All the data need not be stored for efficient computation of the primitive rings set and there is a lot of redundancy in the cluster data structure. So, the cluster data structure is split into two separate data structures, graph and a shortest distance map, as mentioned above, to reduce the redundancy. This reduction in the redundancy makes

computing the all the paths more difficult. But in overall, this restructuring of the cluster data structure results an algorithm better than the brute force algorithm.

Algorithm 9: Improved ring structure algorithm

```
Input:
   1. g, graph
   2. i, id of the atom for which local ring structure is
      to be determined
   3. r_size, size of the ring
Output:
   1. rings: set of all the rings of size r_size
Pseudocode:
   1. s_map = ShortestDistanceMap(g)
   2. m_i = MidNodes(i, s_map, r_size/2)
   3. for all m in mid
        a. paths = AllPaths(g, s_map, m)
        b. for each pairs of paths, pp in paths
             i.  r = FormRing(pp);
            ii.  if PrimitiveRing(r), then
                   1. rings.add(r)
           iii.  endif
        c. end for
   4. end for
   5. return rings
```

Optimized algorithm given in Due to the exponential increase in the number of possible paths and in the size of the cluster structure itself, the brute force algorithm becomes computationally infeasible. At each level, there will be exponential number of atoms and with large rings, information that need to be stored there also becomes exponential. All the data need not be stored for efficient computation of the primitive rings set and there is a lot of redundancy in the cluster data structure. So, the cluster data structure is split into two separate data structures, graph and a shortest distance map, as mentioned above, to reduce the redundancy. This reduction

in the redundancy makes computing the all the paths more difficult. But in overall, this restructuring of the cluster data structure results an algorithm better than the brute force algorithm.

Algorithm 9 first computes shortest distance maps, s_{map}, for all the vertices in graph, g (step 1). Then all the vertices at a distance $\frac{r_{size}}{2}$ from vertex i are, m_i, are computed (step 2). If any of the vertices in m_i has more than one distinct path to the vertex i, then there is possibly a primitive ring. So, all the paths from each middle vertex are found (step 3.a). Each pair of paths is a possible ring, so for each pair of paths, a ring is formed (step 3.b.i) and checked for its primitiveness (step 3.b.ii). If the ring is primitive, then it is added into the set of primitive rings (step 3.b.ii.1).

Above algorithm finds only the even-sized rings. In systems that we are interested in, the graphs are usually bi-partite. So, all the rings are even-sized. For a general graph where there can be odd-size rings, Steps 2-4 is replaced with the following logic.

```
Replaced code:
   2. m1 = MidNodes(i, smap, rsize/2)
   3. for all m in m1
        a. nnm = NearestNeighbors(g, m)
        b. for all nn in nnm
             i.  if(smap(i, nn) == (rsize+1)/2)
                   1. p = AllPaths(g, smap, nn)
                   2. p2.add(p)
            ii.  endif
        c. end for
        d. p1 = AllPaths(g, smap, m)
        e. for each pair of paths, (i, j), s.t. i∈ p1, j∈ p2
             i.  r = FormRing((i, j));
            ii.  if PrimitiveRing(r), then
                   1. rings.add(r)
           iii.  endif
        f. end for
   4. end for
```

Let's say we want to find all the odd sized rings of size r_{size} for a vertex i. Let m is a vertex at a distance $r_{size}/2$ from vertex i. Then a nearest neighbor of m, say nn, that is at a distance $(r_{size} + 1)/2$ from i is in the odd sized ring. After vertex m and vertex nn are found, two sets of paths, p_1 and p_2, are found. p_1 is a set of paths from m to i and p_2 is a set of paths from nn. Any pair of path, (x, y) where x is from p_1 and y is from p_2, form a ring. If this ring is a primitive ring, it is an odd-sized ring in the local ring cluster of vertex i.

6.1.5.2 Ring Visualization

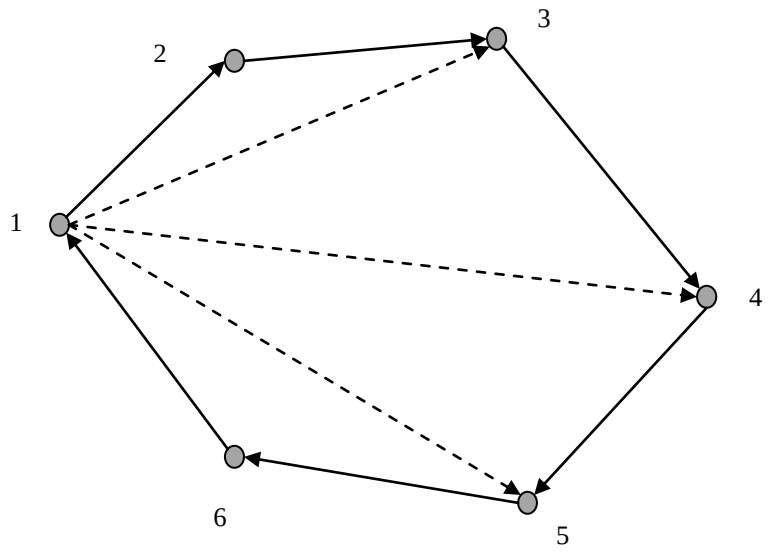

Figure 6-14: Ring triangulation

Algorithm 10: Ring Triangulation Algorithm

```
Input: R, size
  1. For i = 2 to size − 1
       a. v1 = R[1].p − R[i].p
       b. v2 = R[1].p − R[i + 1].p
       c. n = (v1 × v2) / |v1 × v2|
          // n is normal for triangle below
       d. DrawTriangle(R[1].p, R[i].p, R[i + 1].p, n)
  2. End For
```

The primitive rings are visualized as a surface generated by a set of triangles. A triangulation algorithm generates the triangles. Let's we have a ring, $R = \{r_j \mid 1 \leq j \leq |R|\}$, for vertex i, such that r_k is parent of r_{k+1} for $k \geq 1$ and $r_{|R|}$ is parent of r_1. The triangle is calculated by taking set of three vertices, where one of the vertex is the vertex for r_1 and remaining two vertices are r_k and r_{k+1} where $2 \leq k \leq |R| - 1$. For example, let us say R consists of a list of atoms, e.g. $\{1, 2, 3, 4, 5, 6\}$ as shown in the Figure 6-14. R is visualized using the set of triangles generated (as shown in Figure 6-14) according to the following triangulation algorithm. The normal for each triangle is calculated separately.

There can be more than one ways of triangulating the rings. The algorithm shown above treats individual rings independently and the rings are triangulated separately. A triangulation algorithm that takes the complete set of rings into consideration can produce more pleasing visual result. Such algorithm can also provide more meaningful topological information about the ring structure of the system.

6.1.5.3 Performance Evaluation

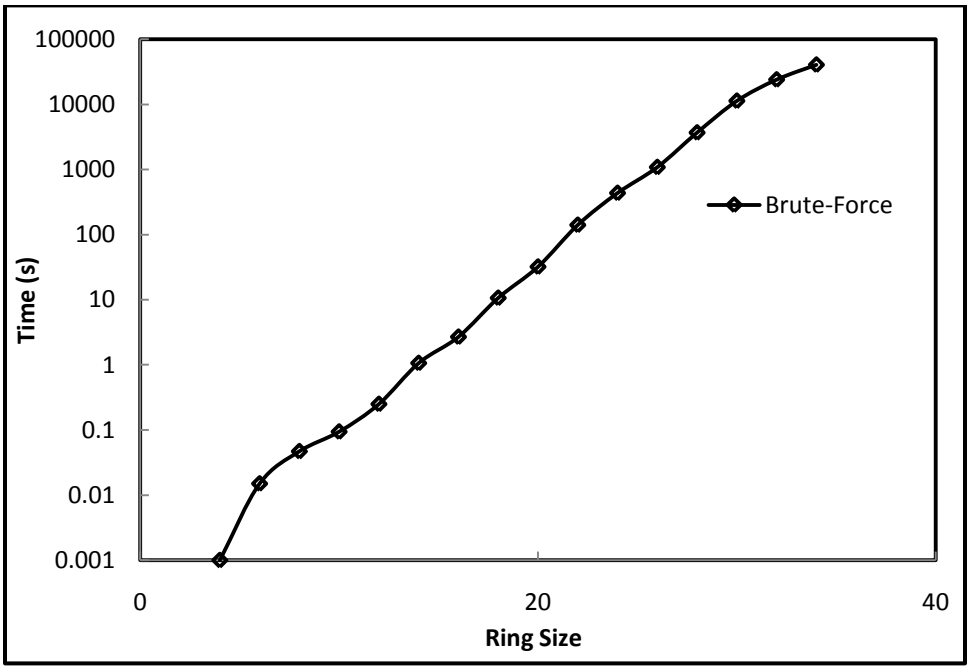

Figure 6-15: Performance of brute force algorithm

The brute force ring computation algorithm executes in real-time for very small rings and is acceptable for interactive visualization. But as the maximum size of the ring increases, the

running time of this algorithm grows unacceptably. In this case, the system size is not restricted and as a result, the system size grows polynomially and the size of the cluster data structure grows exponentially. Due to such exponential growth in the cluster datastructure, this algorithm works interactively only for rings of sizes up to six. The computation time become un-interactive for rings of sizes more than six because the computation time of the algorithm increases exponentially with the ring size (Figure 6-15). The exponential factor comes due to the number of possible paths and the number of possible rings that needs to be tested for their primitiveness. Unless the number of possible rings is reduced to a polynomial order, this algorithm will remain exponential.

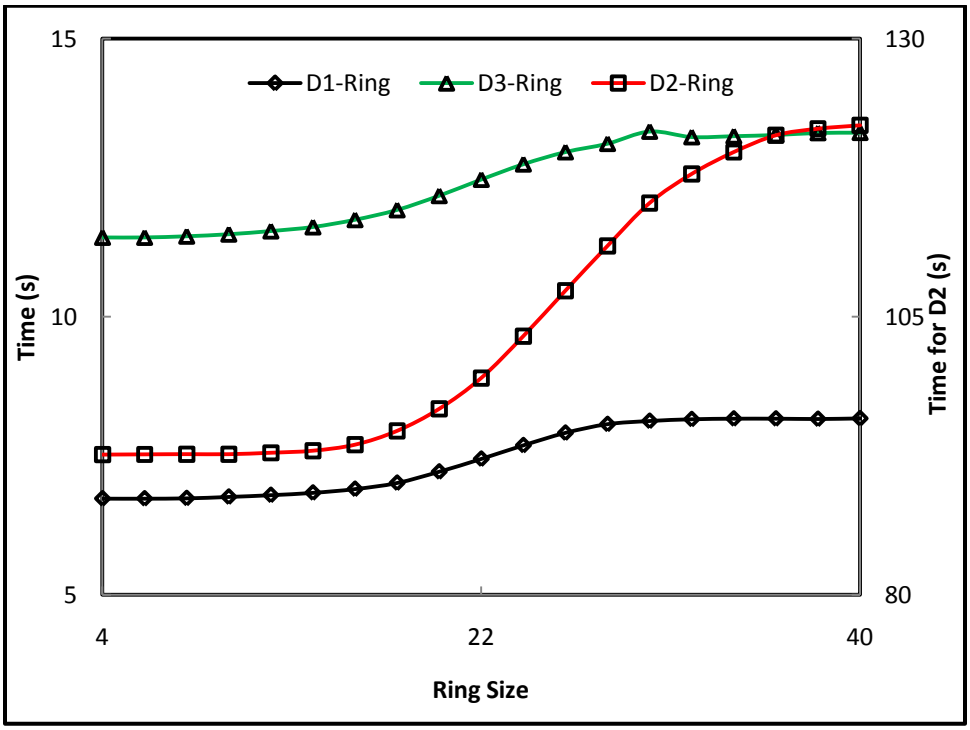

Figure 6-16: Ring structure algorithm performance

The brute force algorithm does not utilize one obvious fact about the ring computation algorithm. Large rings have high possibility of having shortcuts and thus fail to satisfy the primitiveness criterion. The improved algorithm removes such large rings very early in the stage thus saving a lot of computation time. It utilizes the fact that the root node will have one vertex at a distance half the size of the ring. If there are no vertices at such a distance then there are no rings of this size. This condition is satisfied less frequently for large rings and thus saves a lot of

computation. The improved algorithm checks for this condition when it looks for the mid nodes in the rings.

The performance of the improved algorithm gave interactive computation times for small ring sizes (Figure 6-16). The computation time initially increases exponentially and levels off after a certain ring size. The initial exponential term in the ring computation comes from the exponential number of possible paths that can be taken to reach one atom from another atom. In a fixed size system, two atoms cannot be farther than a certain distance. Hence, after a certain ring size, the number of possible mid nodes decreases and the computation time starts to level off. When the ring size is increased further, the algorithm cannot find any mid nodes and saves all the computation time required to check for the primitiveness of the ring. The computation times reported here is the time required to compute rings for all the atoms in the system.

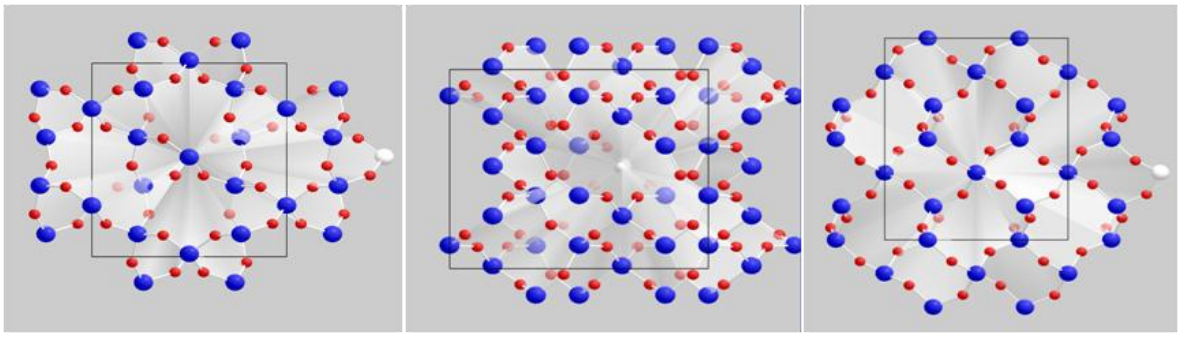

Figure 6-17: Three views of the visualization of the rings for quartz mineral

Like other computationally intensive algorithms, ring algorithm is also implemented to run in separate thread from the main application thread. To make it useful for the interactive visualization, the rings are computed only for the selected atom and only for current time step not for all the atoms in the system for the complete dataset. It can be inferred from the above chart that the computation takes less than a second (0.113495 s for D1, 0.727522 s for D2 and 0.15848 s for D3) to compute all the rings for a selected atom. These extra restrictions make the interactive visualization of rings possible. Figure 6-17 shows three views of the rings computed for a Quartz system and visualized using the ring visualization algorithm presented in previous section. In this system, there are 6 rings with 12 atoms (6 Silicon atoms and 6 Oxygen atoms) and 40 rings with 16 atoms (8 Silicon atoms and 8 Oxygen atoms).

6.1.6 Diffusion Sphere

Spheres are used to visualize the amount of displacement each particle in the system goes through. Such displacements are computed using the methods described in Section 5.3. These spheres are called Diffusion Spheres. We use two different diffusion spheres, instantaneous diffusion sphere and average diffusion sphere. The center and the radius for diffusion spheres are computed differently for different diffusion visualizations.

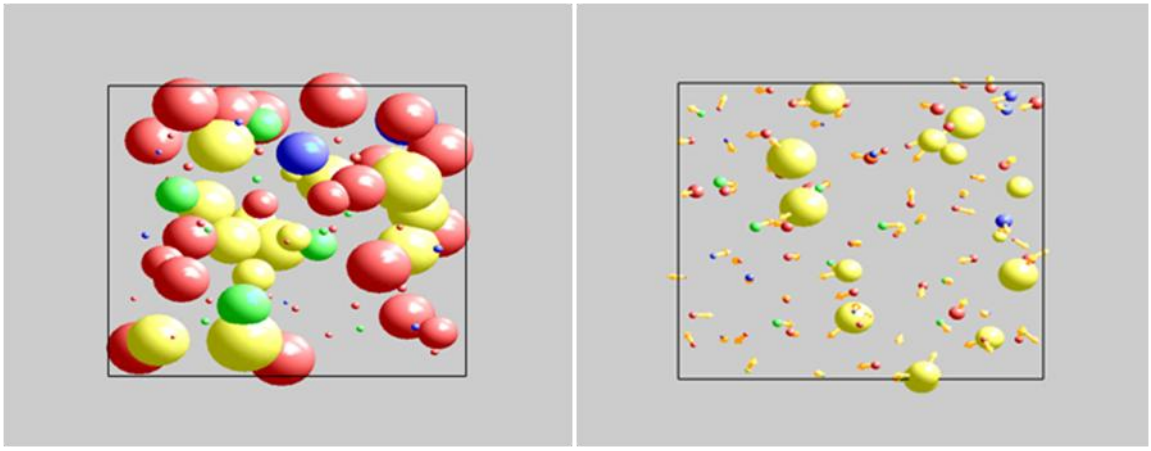

Figure 6-18: Average and instantaneous diffusion spheres

For average displacement, there is no one direction of displacement and center of the sphere is the mean position of the position set. The radius of the sphere is the distance between the mean position and the farthest position in the position set from the mean position. For instantaneous displacement and finite-time span displacements, there is a direction of net displacement. The direction is indicated using an arrow on the surface of the sphere. The length of the arrow shown on the surface of the sphere is of fix length.

6.1.6.1 Performance Evaluation

The performance measurement shows overall trend of increasing rendering time with the number of atoms in the system. In case of average diffusion sphere, the computation and rendering time increases linearly with respect to the number of steps used. The computation and rendering time remains constant for the instantaneous diffusion sphere. In both the cases, most of the time is spent in rendering the spheres unlike in case of previous algorithms where analytical portions were dominating. The time to render spheres is very negligible because of the size of the system we are analyzing. The system with the largest number of atoms has 216 atoms.

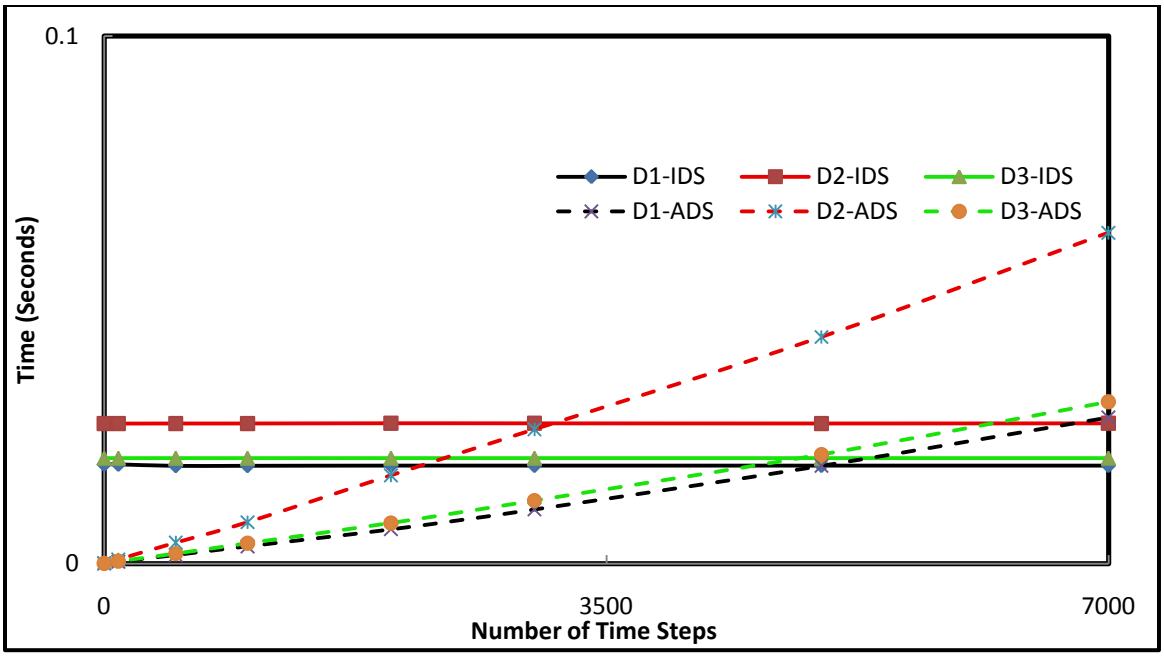

Figure 6-19: Performance of diffusion sphere modules

6.1.7 Diffusion Ellipsoid

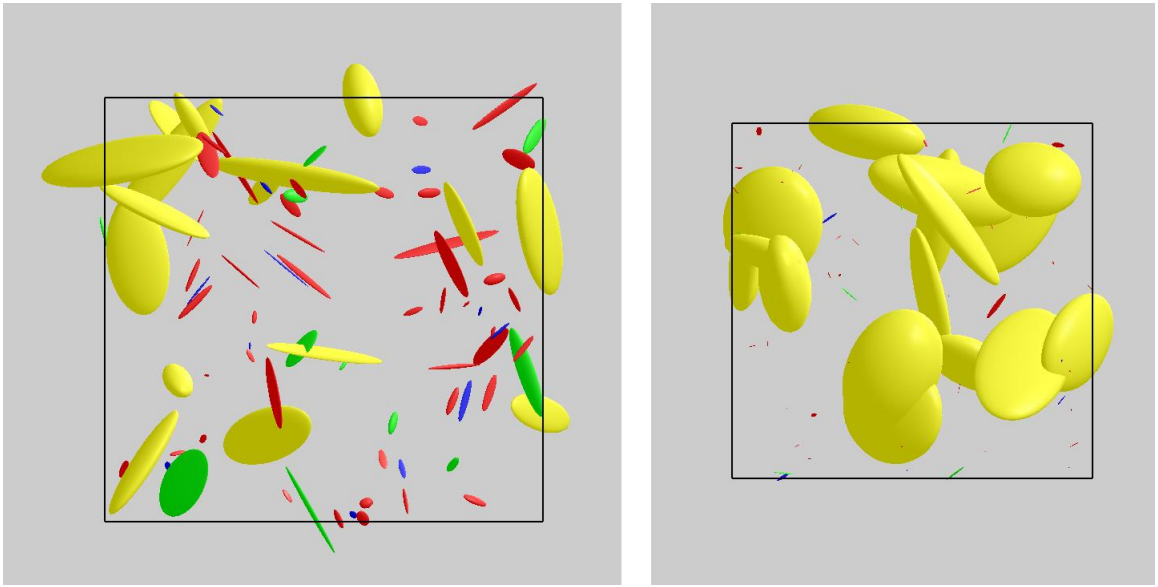

Figure 6-20: Diffusion ellipsoid representation of atomic displacement

The ellipsoids are used to visualize the eigenvalues and eigenvectors of the covariance matrices (Section 5.4). The eigenvalues are the lengths of the three orthogonal axes of the ellipsoid and the eigenvectors form an orthonormal frame of reference to orient the ellipsoid. The mean position of the set of positions is the center of the ellipsoid. The spherical structure in

diffusion sphere is isotropic and fails to capture the anisotropy in the atomic movement. The ellipsoid representation gives a more informative visualization of the aggregate atomic movement.

6.1.7.1 Performance Evaluation

The time taken to compute and render ellipsoids shows a linear trend with respect to the number of time steps. The linear trend is due to the time taken to compute covariance matrices. The computation and rendering time also increases linearly with respect to the number of atoms in the system as indicated by the separation between the lines in the following chart.

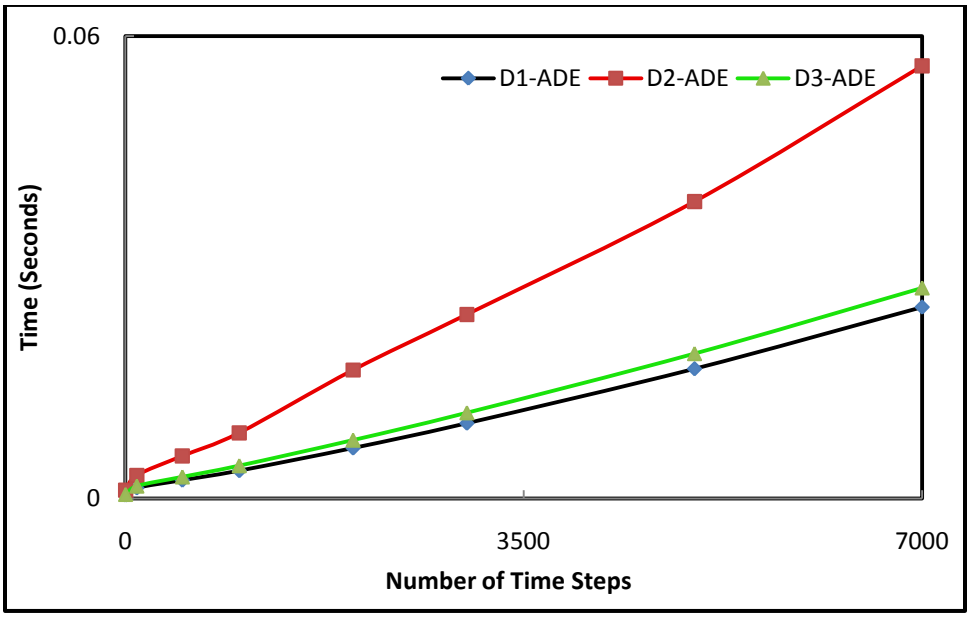

Figure 6-21: Performance of the diffusion ellipsoid module

6.1.8 Pathline

The pathlines are used to visualize the raw movement of individual particle as well as to visualize the cumulative movement and stability of a complete structural unit, such as a polyhedron (Figure 6-22). The figure shows pathlines to visualize the movement of individual particles (left), and the collective movement of a group of atoms forming a complete structural unit (middle and right). In the figure, the structural unit is a polyhedron. In case of the polyhedron, the yellow line shows the movement of the central atom, where as the blue and red lines show the movement of the coordinated atoms. When the path for a coordinated atom is blue, the central atom and the coordinated atom are bonded at that instant. When the path is red

at any instant, the central atom and the coordinated atom are not bonded at that instant. The paths of the coordinated atoms in the middle figure contain large proportion of red pathlines that indicates the polyhedron is not very stable. On the other hand, the pathlines on the polyhedron in the figure on the right are all blue that indicates the polyhedral structure is stable.

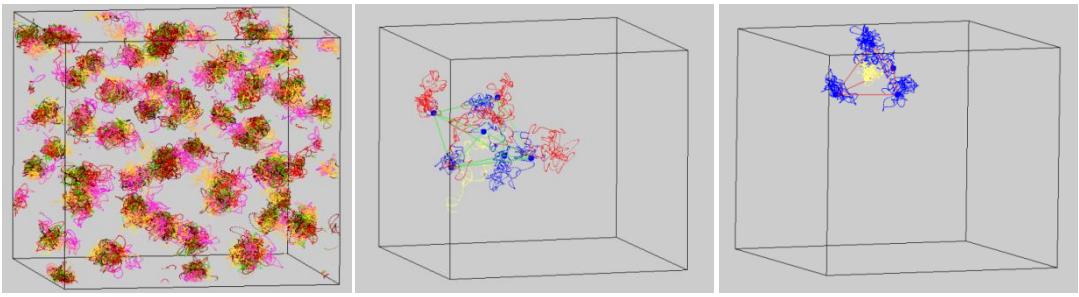

Figure 6-22: Pathline visualizing movement of atoms and complete structural unit

6.1.8.1 Performance Evaluation

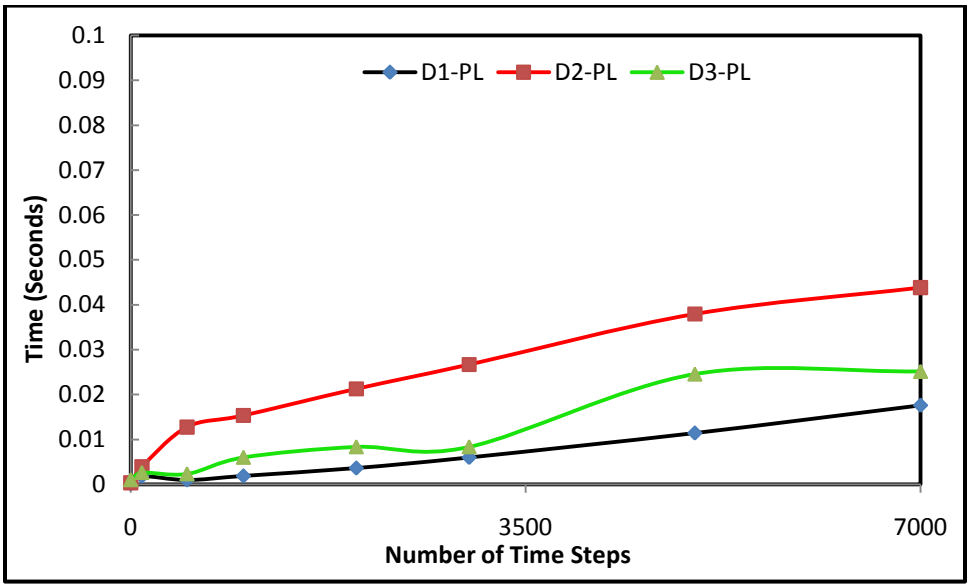

Figure 6-23: The relationships between time taken to render pathlines for individual atoms

The pathline rendering time increases linearly with the number of snapshots used for the pathline and also increases linearly with the number of atoms in the system as indicated by the separation between lines in the chart (Figure 6-23).

6.2 Rendering Algorithms

Atomistic visualization system renders large number of atoms and bonds. In practice atoms are modeled as spheres and bonds are modeled as cylinders. Rendering a system containing hundreds of atoms and hundreds more of bonds normally requires transferring thousands of polygon information from CPU to GPU. For modern computer systems, such numbers is not big but when a system containing thousands of atoms and bonds is visualized, number of polygons becomes a bottleneck. One way around such problem is to utilize the programmability of modern GPU to render spheres and cylinders. The rendering algorithms used to render different shapes are explained in the following sections.

6.2.1 Quadric Rendering

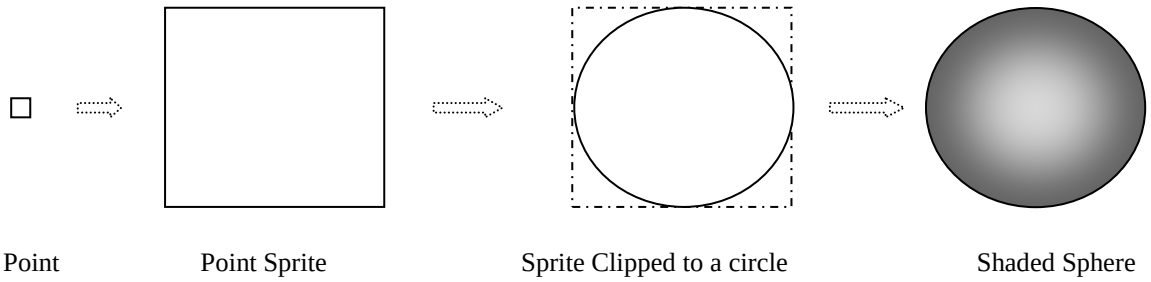

Figure 6-24: Quadric shading from point sprite

Hardware accelerated algorithm for quadric rendering is implemented and tested. The algorithm renders high quality resolution independent ellipsoid and spheres. Irrespective of the size of the viewport, the quality of the ellipsoids and spheres remains same. In polygon based modeling of spheres, quality of spheres degrades with the size of the viewport.

The vertex shader receives the center position of the quadric, size of the quadric, projection matrices and viewport information from the application. Then it computes the size of the point that completely covers the quadric. Fragment shader then computes per pixel normal information and performs lighting computation for each pixel. This results very high quality output. The output is resolution independent because the algorithm uses implicit function for the rendering and since, the light calculations are done separately for each pixel covered, the output quality is also very good. The shader sources are shown in the following figures.

```
#version 110
varying vec4 center;
varying float radius;
varying mat4 MT, IMT, IT;
uniform mat4 T;
uniform mat4 D;
uniform vec3 vp;
void main() {
     mat4 PMT = gl_ProjectionMatrix*gl_ModelViewMatrix*T;
     vec4 r1 = vec4(PMT[0][0], PMT[1][0], PMT[2][0], PMT[3][0]);
     vec4 r2 = vec4(PMT[0][1], PMT[1][1], PMT[2][1], PMT[3][1]);
     vec4 r4 = vec4(PMT[0][3], PMT[1][3], PMT[2][3], PMT[3][3]);
     vec2 bx, by;
     float a, b, c;
     a = dot(r4*D, r4); b = dot(r1*D, r4); c = dot(r1*D, r1);
     float disc = sqrt(b*b-a*c);
     bx.x = (b - disc) / a;
     bx.y = (b + disc) / a;
     b = dot(r2*D, r4); c = dot(r2*D, r2);
     disc = sqrt(b*b-a*c);
     by.x = (b - disc) / a;
     by.y = (b + disc) / a;
     center = vec4(dot(r1*D, r4), dot(r2*D, r4), 0.5, dot(r4*D,
r4));
     float size = max(abs(bx.x-bx.y)*vp.x, abs(by.x-by.y)*vp.y);
     gl_PointSize = size;
     radius = 0.5*gl_PointSize;
     MT = gl_ModelViewMatrix*T;
     gl_Position =ftransform();
     IT[0][0]=1.0/T[0][0]; IT[1][0] = 0.0; IT[2][0] = 0.0;
     IT[3][0] = -T[3][0]/T[0][0];
     IT[0][1]=0.0; IT[1][1] = 1.0/T[1][1]; IT[2][1] = 0.0;
     IT[3][1] = -T[3][1]/T[1][1]; IT[0][2]=0.0; IT[1][2] = 0.0;
     IT[2][2] = 1.0/T[2][2];
     IT[3][2] = -T[3][2]/T[2][2];
     IT[0][3]=0.0; IT[1][3] = 0.0; IT[2][3] = 0.0; IT[3][3] =
1.0;
     IMT = IT*gl_ModelViewMatrixInverseTranspose;
}
```

Figure 6-25: Vertex shader source for quadric rendering

```
#version 110
varying vec4 center;
varying mat4 MT, IMT, IT;
varying float radius;
uniform mat4 T, D;
uniform vec3 vp;
void main() {
      float far = -8.024, near = 8.024;
      vec4 coord = gl_FragCoord;
      coord.x = (2.0*coord.x)/vp.x - 1.0;
      coord.y = (2.0*coord.y)/vp.y - 1.0;
      mat4 invMat = IT*gl_ModelViewProjectionMatrixInverse, ITT;
      coord.z = 0.0; coord.w = 1.0; coord = invMat*coord;
      vec4 c3 = invMat[2];
      float a, b, c;
      a = dot(c3*D, c3); b = dot(coord*D, c3); c = dot(coord*D, coord);
      float disc = b*b-a*c; if(disc < 0.0) discard;
      float z1, z2, sqD = sqrt(disc);
      z1 = (-b - sqD)/a; z2 = (-b + sqD)/a;
      coord  += min(z1, z2)*c3;
      vec4 pe = gl_ModelViewMatrix*T*coord, np;
      ITT[0][0] = IT[0][0]; ITT[1][0] = IT[0][1]; ITT[2][0] = IT[0][2];
      ITT[3][0] = IT[0][3]; ITT[0][1] = IT[1][0]; ITT[1][1] = IT[1][1];
      ITT[2][1] = IT[1][2]; ITT[3][1] = IT[1][3]; ITT[0][2] = IT[2][0];
      ITT[1][2] = IT[2][1]; ITT[2][2] = IT[2][2]; ITT[3][2] = IT[2][3];
      ITT[0][3] = IT[3][0]; ITT[1][3] = IT[3][1]; ITT[2][3] = IT[3][2];
      ITT[3][3] = IT[3][3];
      np = gl_ModelViewMatrixInverseTranspose*ITT*coord;
      vec3 n = normalize(np.xyz);
      struct gl_LightSourceParameters lsp = gl_LightSource[0];
      struct gl_MaterialParameters mp = gl_FrontMaterial;
      vec3 light = normalize(lsp.position.xyz);
      float fract = dot(light, n);
      vec4 color = 2.0*lsp.ambient*mp.ambient;
      if(fract > 0.0) {
            float s = dot(lsp.halfVector.xyz, n);
            if(s > 0.0) s = pow(s, 50.0);
            else s = 0.0;
            color = 2.0*lsp.ambient*mp.ambient +
fract*lsp.diffuse*mp.diffuse + 0.6*s*lsp.specular*mp.specular;
      }
      gl_FragColor = color;
      gl_FragDepth = (pe.z-near)/(far-near);
      gl_FragDepth = (pe.z - range.x)/(range.y-range.x);
}
```

Figure 6-26: Fragment shader source for quadric renderer

6.2.2 Transparency Using Depth Peeling

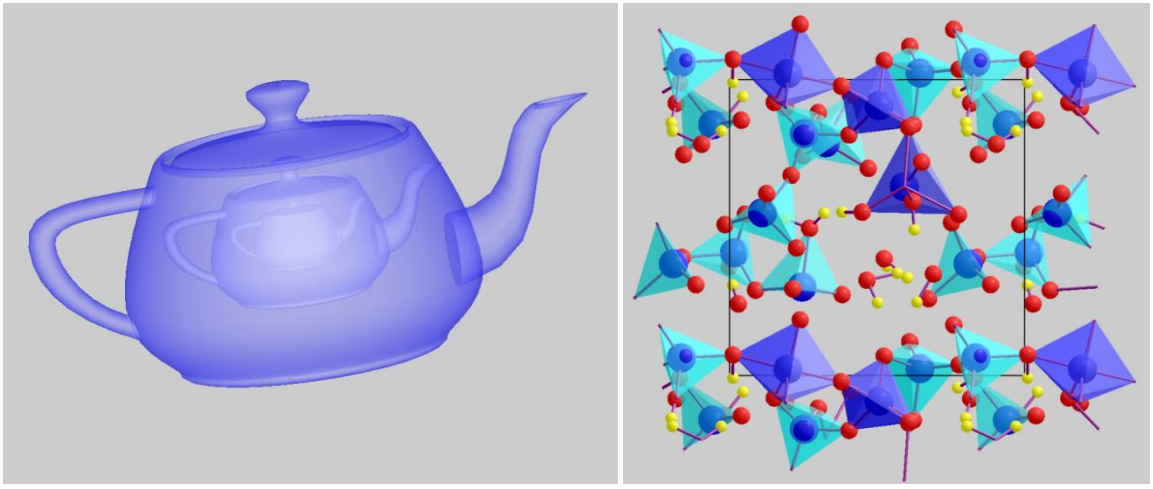

Figure 6-27: Transparency using depth peeling

When there are many different objects on the screen, the clutter makes it hard to comprehend the presented information and transparency increases visibility of the structures hidden behind or hidden inside the other structures. There are different ways of achieving transparency effect. The simplest one is to use alpha blending. In this case, each object is assigned an alpha value according to the amount of desired transparency. The objects are then drawn in back to front order and alpha value on each pixel is used to combine the incoming color and the current color to assign a new color to the pixel. This approach is unsuitable for interactive application like this. Since this algorithm depends on the order of the objects rendered, slight change in model-view matrix warrants reordering of the complete scene. Reordering is tedious. There is a better approach that is order independent and is also suitable to interactive visualization. This is approach is called *depth peeling [89]*.

Depth peeling algorithm is used to achieve transparent effects. Main idea behind this algorithm is to peel away individual layers of the rendered scene and store the layers in multiple textures. After all the layers have been computed, they are blended together to form the final transparent scene. The amount of blending is controlled by the alpha value specified while peeling the texture. Depth peeling obviates the need to sort scene in certain depth order. Such sorting is impractical in an interactive visualization system. User of such system is constantly interacting and changing the viewpoint by rotation, translation or zooming. Such changes in viewpoint warrant reordering and resorting of the complete scene to maintain the depth order,

but this algorithm is insensitive to the depth order of objects in the scene being rendered and does not need resorting and reordering of the objects. For this reason depth peeling algorithm is also called order-independent transparency method. Below is the fragment shader source code for transparency using depth peeling.

```
uniform sampler2DShadow depth;
uniform vec2 v;
uniform float a;
void main(void) {
      vec4 coord = gl_FragCoord;
      coord.xy = vec2(coord.x/(v.x), coord.y/(v.y));
      vec4 depthVal = shadow2D(depth, coord);
      if(depthVal.z == 0.0) discard;
      gl_FragColor = vec4(gl_Color.rgb,a);
}
```

Figure 6-28: Fragment shader source for transparency using depth peeling

"`depth`" is a depth texture that is bound at the application level. At the beginning, depth texture is initialized to smallest depth value in the depth volume. `shadow2D(depth, coord)` returns 1.0 is if the depth value of `coord (coord.z)` is greater than the value of the depth texture, `depth`, at `coord.xy`. Depth texture is configured at the application level to use "greater" function while comparing depth value and the texture content. Following two OpenGL statements configures the depth texture that are used for peeling the layers.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, GL_COMPARE_R_TO_TEXTURE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_FUNC, GL_GREATER);
```

The first statement tells the OpenGL pipeline to compare *r*-component of the texture coordinate with the texture content and second statement tells the OpenGL pipeline to return 1.0 if texel value is greater than the r-value otherwise return 0.0.

The layers are blended together to form the final transparent scene. Blending is done according the equation given below where C_s and C_d are colors of texture and corresponding fragment, and α_s is the opacity value of the texture:

Equation 6-10: Blending Equation

$$C = \alpha_s C_s + (1 - \alpha_s) C_d$$

The layers are peeled in front to back manner and blended in back to front manner. Transparency using this algorithm requires rendering the scene over and over again for $(n_{layers} + 1)$ times. Frame rate for scene, where the transparency is enabled, is less than a scene, where transparency is not enabled.

6.2.3 Vector Rotation

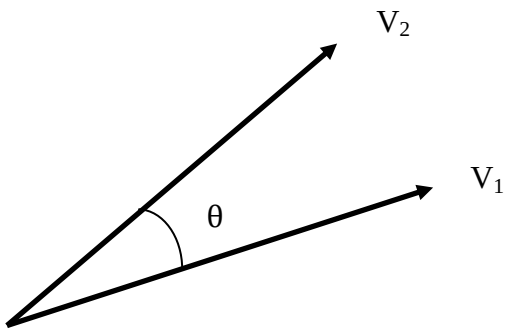

Figure 6-29: Rotating from V1 to V2

A vector, $V_1^{'}$, is rotated into another vector, $V_2^{'}$ [referred in [90] and first derived in [45] by multiplying it with the following rotation matrix.

Equation 6-11

$$\begin{pmatrix} u_x^2 + (1 - u_x^2)\cos\theta & u_x u_y(1 - \cos\theta) - y_x \sin\theta & u_x u_y + u_y \sin\theta \\ u_x u_y(1 - \cos\theta) + u_x \sin\theta & u_y^2 + (1 - u_y^2)\cos\theta & u_y u_z(1 - \cos\theta) - u_x \sin\theta \\ u_x u_z(1 - \cos\theta) - u_y \sin\theta & u_y u_z(1 - \cos\theta) + u_x \sin\theta & u_z^2 + (1 - u_z^2)\cos\theta \end{pmatrix}$$

This rotation matrix containss $\cos\theta$, which is just $\overrightarrow{V_1}.\overrightarrow{V_2}$and $\sin\theta$, which is $||\overrightarrow{V_1} \times \overrightarrow{V_2}||$ and such trigonometric functions are computationally very intensive. They can be avoided by following procedure.

Let,

Equation 6-12

$$\vec{V} = \overrightarrow{V_1} \times \overrightarrow{V_2}$$

Equation 6-13

$$c = \overrightarrow{V_1}.\overrightarrow{V_2}$$

Equation 6-14

$$h = \frac{1-c}{1-c^2} = \frac{1-c}{\vec{V}.\vec{V}} = \frac{1}{1+c}$$

The above rotation matrix can be simplified into another form that has no trigonometric functions and no square roots.

Equation 6-15

$$R(V_1, V_2) = \begin{pmatrix} c + hV_x^2 & hV_xV_y - V_z & hV_xV_z + V_y \\ hV_xV_y + V_z & c + hV_y^2 & hV_yV_z - V_z \\ hV_xV_z - V_y & hV_yV_z + V_z & c + hV_z^2 \end{pmatrix}$$

This rotation matrix is used to orient bond toward appropriate direction. A bond is modeled using a cylinder. Cylinders are easy to construct with orientation toward one of the standard axes. The rotation matrix constructed is then used to appropriately orient the cylinder.

Chapter 7 System Design and Implementation

I presented various methods for the analysis in previous chapters. Here, I discuss the design decisions taken while developing the visualization system.

7.1 Platform, Programming Language and Tools

C/C++ programming language is used for the system development. Standard Template Library (STL) provided in standard C++ is also used. STL provides performance guarantee of the data structures and algorithms in the library. OpenGL is used for graphics rendering supplemented by GLUT and GLUI libraries. OpenGL provides the algorithms for drawing polygonal primitives such as lines, triangles and rectangles. GLUT library abstracts away the underlying graphics context initialization and management, and presents a cleaner interface useful for application development. GLUI is a user interface library built upon GLUT and OpenGL. GLUT and GLUI are open source libraries and are available freely.

OpenGL shading language (GLSL) is used for hardware accelerated graphics rendering. Shader written in GLSL is executed directly in GPU and is also faster compared to a routine written in high level languages executing in CPU. Shader improves the rendering performance and enables highly flexible rendering algorithm to be implemented. Programs written using GLSL are compiled during the runtime and OpenGL drivers come with built-in compiler for GLSL. Qhull [84] algorithm is used to compute coordination polyhedral. It uses the “beneath and beyond” algorithm described in 0 to compute coordination polyhedra. MATLAB™ C/C++ libraries [91] are used to compute eigenvalues and eigenvectors of the covariance matrices required during the principal component analysis for diffusion ellipsoid.

Also, some analytical algorithms, such as RDF calculation and ring structure calculation, take too long for an interactive visualization. These algorithms are implemented in separate threads so that they do not deteriorate the interactivity of the system. Outputs from these algorithms are made available for exploration as soon as new data is computed. Multithreading makes the system appear more responsive and more interactive. A selected subset of the complete positional dataset is stored completely in main memory and a separate coordinator module coordinates the access to the dataset. This way algorithms running in multiple threads do

not corrupt the data. Then, data structures are constructed dynamically from the positional dataset for different set of analyses and visualizations.

7.2 System Design

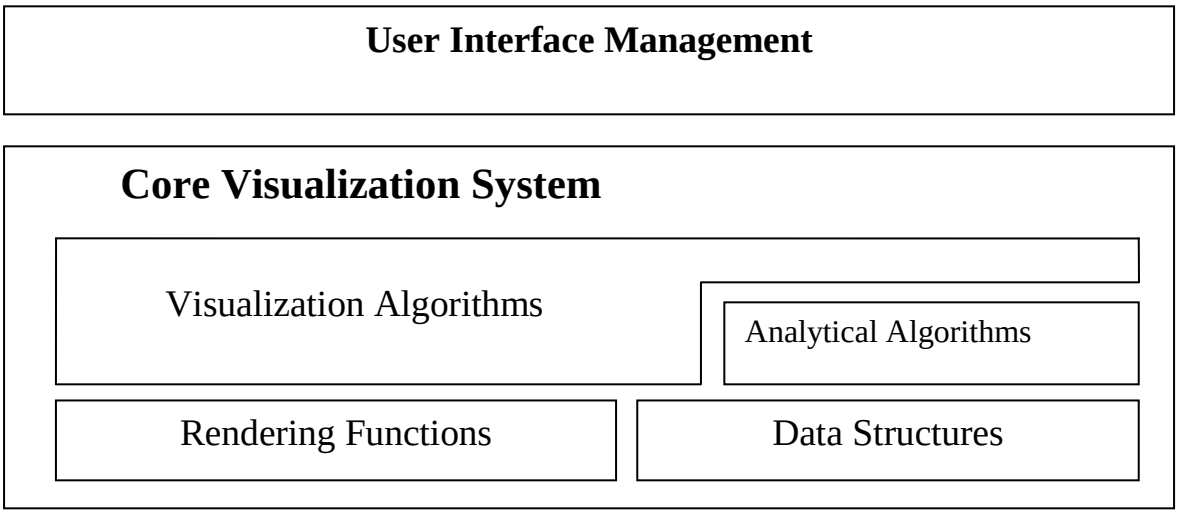

Figure 7-1: Design of the interactive atomistic visualization system

The atomistic visualization system is designed in a layered fashion (Figure 7-1). At the bottom, there are rendering algorithms to render spheres, cylinders and ellipsoids and data structures to handle datasets. The layers above this provide functionalities for analysis and visualization. The visualization algorithms have direct access to the underlying rendering algorithms, low level data structures and also to the analytical algorithms on the same layer.

This system can also be viewed as an example of hybridization of scientific visualization and information visualization. The atomic position-time series dataset represents a real material system and when the dataset is rendered, the physical relevance (3D atomic arrangement) is to be preserved. Subsequent analyses generate additional data that represent information that does not have any direct physical relevance. Such information is useful in understanding system from different perspectives. The information can be superimposed with the physically relevant data and rendered in a more flexible way. For instance, coordination environment represents multivariate information and can be rendered using shape, size and color. The system is also designed with data exploration in mind [92]. A periodic table containing atomic name, atomic symbol, atomic weight, covalent radius and Van der Waals radius for 108 elements is also integrated into the system. It is used to make a consistent representation of atoms across different

systems. This is also useful in representing the system in a physically meaningful way and is helpful in understanding the system under study.

7.2.1 Rendering Functions

Rendering functions are responsible for setting up material properties of the graphical objects, setting up the lighting parameters and drawing primitives. These functions are also responsible for setting up the shading context to execute the vertex and fragment shaders. The visualization algorithms frequently make use of these functions to convert information into various graphical formats.

7.2.2 Data Structures

The individual modules maintain their own data structures to manage the module specific information. As has already been discussed, the common neighbor cluster computation module maintains an extensive datastructure. The modules that require manipulation of graph maintain a separate graph datastrcuture apart from the module specific datastructure. The ring structure computation algorithm also maintains a shortest distance map apart from the graph datastructure to store the shortest distance between selected point and all the other points and as the rings are computed, it maintains the list of currently computed rings. The list of rings is incrementally updated when new rings are computed. The objective of the module specific data structures is to provide higher level interfaces to the underlying data that other algorithms can use to provide higher level functionalities and to reduce the unnecessary recomputation.

7.2.3 Analytical Algorithms

The analytical algorithms operate on the data maintained by the data structures to provide a higher level picture of the underlying dataset. For example, information maintained by the coordination information data structure is used to enable computation of coordination stability, coordination cluster, and polyhedral surface computation among other things. As shown in the Figure 7-1, analytical algorithms have access to the data structures only. They do not have access to the rendering functionalities. Such separation allows for easy addition of other analytical functionalities in future.

7.2.4 Visualization Algorithms

The visualization algorithms utilize all the other modules to provide an interface to the underlying dataset. They utilize rendering functions to present the information generated by analytical algorithms. The visualization algorithms also have access to the underlying raw dataset so that the data points can be rendered in their raw form without any extra information attached to them.

7.2.5 User Interface Management

The multiple visualization algorithms are tied together by a user interface. The details about the user interface are provided in a separate section below. The user interface is used to provide a consistent face to the multiple visualization approaches.

7.3 Information Representation

Extracted information is rendered into various forms using color and intensity, length, thickness, size and geometrical objects.

7.3.1 Color and Intensity

Multiple colors are used to encode atomic species, coordination number and distance between two atoms. For discrete data like atomic species or coordination number, we can use an indexed color map. But for a continuous quantity like time, we design a transfer function that can create color according to the quantity. Let's say $t(.)$ is such a transfer function that takes a value and returns a color representing the value:

Equation 7-1

$$t(v) = \vec{c}$$

Such a function can be modified to provide different features. We use following transfer to encode the elapsed time in the pathline for individual atom:

Equation 7-2

$$t(v) = [\frac{v}{v_{max}}, 1 - \frac{v}{v_{max}}, 0.0, 0.0]$$

Here, v_{max} is the maximum value that v can take and the output is in RGBA format. So, the R and G component always remain between 0.0 and 1.0 where as B and A components are always 0.0.

Theoretically, any number of colors can be used but there are some physical limitations imposed by human visual system. Human eye can simultaneously reliably distinguish between seven and eleven colors [93] and use of more colors to represent more information results visual clutter. The visual clutter does not increase the information content, rather degrades the quality of the existing information.

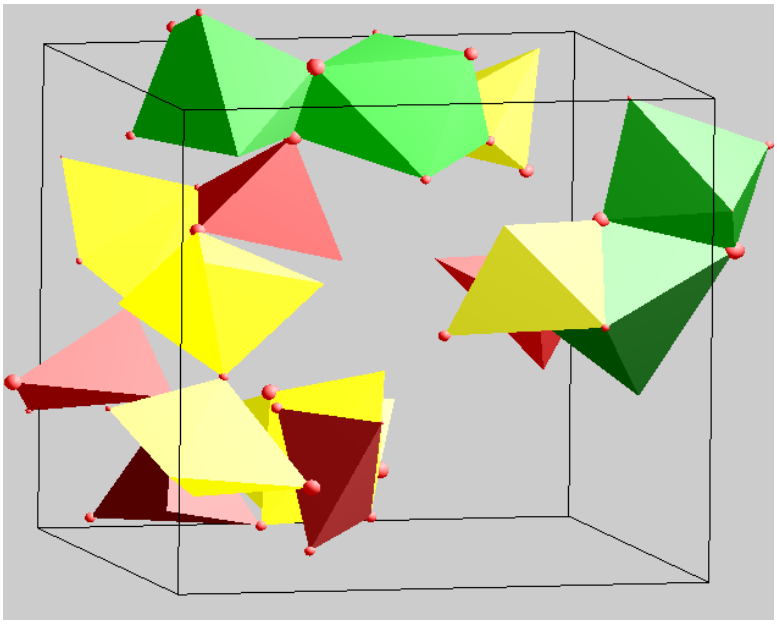

Figure 7-2: Stability encoded polyhedra and the coordinated atoms

Variation in the bond length, *r*, of a given bond between $r_{\min}$ and r_{max} is quantified using:

Equation 7-3

$$\lambda_B = \frac{(r - r_{min})}{(r_{max} - r_{min})}$$

λ_B varies from 0, when two atoms are separated by the minimum distance, to 1, when the atoms are separated by the maximum distance. Let's suppose we have I_{max} number of colors in the index map, then λ_B is converted to an index to the color map as:

Equation 7-4

$$I = \lfloor \lambda_B I_{max} \rfloor$$

The index map is assumed to a 0-indexed map.

The intensity of the color is also varied to represent stability of the coordination polyhedron. This is effective in conveying relative stabilities among multiple snapshots. Figure 7-2 shows stability of polyhedra encoded using the intensity variation. There are distinct variations in the green polyhedra. Two of the four green polyhedra are distinctly faded where as on is very bright. The faded green color means those polyhedra are unstable in that configuration and bright green colored polyhedra mean those polyhedra are stable. We can deduce similar conclusions about other polyhedra as well.

7.3.2 Thickness and Size

The thickness of a bond is used to visualize the stability of the atomic bond. A minimum bond size is always assigned to a bond so that the bond is visible all the time. The size is used to encode stability of a coordination number and the amount of diffusion of an atom. A user-defined minimum size is also used to make the shape visible. A fixed sizes proportional their atomic weights are given to atoms of different species.

7.3.3 Geometric Objects

Polyhedron, sphere, ellipsoid, line and cylinders visually represent different set of information. A polyhedron encodes coordination environment of an atom. A sphere and an ellipsoid encode the amount of atomic diffusion. The lines and cylinders represent path(s) of a single atom or a group of atoms. The cylinders are also used to represent bond between two atoms and the sphere is also used to model an atom.

7.4 Information Presentation

A representational scheme for some information is selected and then visualized on the computer screen. The screen is a two-dimensional plane of a limited size whereas the data are three-dimensional and multivariate. A single piece of information does not usually result insight so different types of information are combined together. We try to follow the visual information seeking mantra: overview first, zoom-and-filter, then details-on-demand [94] for information

presentation. The data points, in our case the atoms, have many attributes. Some of the attributes are specified in the dataset and others are computed during the visualization. [94] proposes seven types of tasks that a user must be able to perform on the data points: overview, zoom, filter, details-on-demand, relate, history and extract. Even though the current interface does not provide all seven tasks, the techniques developed here flow along the similar lines.

7.4.1 Overview

Animation, pathline and RDFs provide complete overview of the dataset. Initially we do not know which temporal region is of interest and animation gives some idea about the possible regions of interest. The average diffusion sphere and the average diffusion ellipsoids also provide aggregate information about the system dynamics. Once selection temporal region is selected, one particular atom or set of atoms can be selected for further exploration. The RDF provides complete spatial map of the data points. Domain specific knowledge of the user comes into picture here to interpret the various properties of RDFs that can be useful for further analysis.

7.4.2 Zoom

Once the temporal region for further exploration is selected, spatial region around an atom or a group of atoms can be zoomed and explored further using the selective display feature. For instance, RDF plot shows there are structures farther away from r_{min} also. These structures are explored by selecting different set of cutoffs from the user interface.

7.4.3 Details on demand

The coordination environment, coordination cluster and coordination distribution, three types of cluster structures, bond-lifetimes and various speciation can be explored on demand to understand the system in more detail. The computed quantitative information can also be saved to files for future analysis by other external tools. These information can be computed for each particle and also can be aggregated for the complete system.

7.5 User Interface

The user interface enables active interaction with the visualization system. The system provides features to rotate the scene, to select a group of atoms and analyze relevant attributes of the set of atoms. The user can use the two markers provided in the RDF plot to select minimum

cutoff distance and maximum cutoff distance for any pair of species. Any change in a parameter results immediate visual feedback facilitating visual exploration. Push buttons, selection boxes and textboxes are used to design a prototype user interface. The user interface provides access to underlying analyses and rendering parameters. These parameters control information extraction from the dataset and final graphical output generation. The complete set of the parameters for the currently selected module is readily available from the currently shown interface. There is no need to navigate multiple levels of menu hierarchy to change one parameter. A spreadsheet-like interface is also implemented to have a comparative analysis among different outputs. The spreadsheet interface is very primitive in the current implementation. A complete spreadsheet interface provides the typical features such as a tabular layout, operator and dependency between the cells [95].

Both parallel and perspective projections are supported and a user can navigate through the atomic system in different directions by using basic operations such as rotation, translation and scaling. As such, it is efficient in tracking and studying particular local structure such as defects in the configuration. We provide solution to occlusion problem by supporting selective dynamic manipulation scheme, which combines various options such as selective rendering, multiple rendering and transparency. Browsing and highlighting options are added to relate various objects on the scene. The user interface is changed according to which visualization module is selected. The parameter set for the selected module can be changed immediately.

7.5.1 Multiple Species and Multiple Viewports

If there are $|\Lambda|$ species, the screen will have $|\Lambda|^2$ viewports showing coordination between all possible pairs of species. User can select any particular viewport using mouse to further explore the coordination environments in more detail. Currently selected viewport is indicated by a red-rectangle around the viewport. It is also possible to select subset of pairs of species from the set of all pairs of species in the system. Creating layout for such selected subset of pairs of species in slightly more complicated compared to creating for all the pairs of species. As mentioned above, there are $|\Lambda|^2$ pairs of species where $|\Lambda|$ is the number of species in the system. Let's suppose user selected λ pairs of species from $|\Lambda|^2$ pairs of species. Now, the screen is split into multiple viewports using following algorithm.

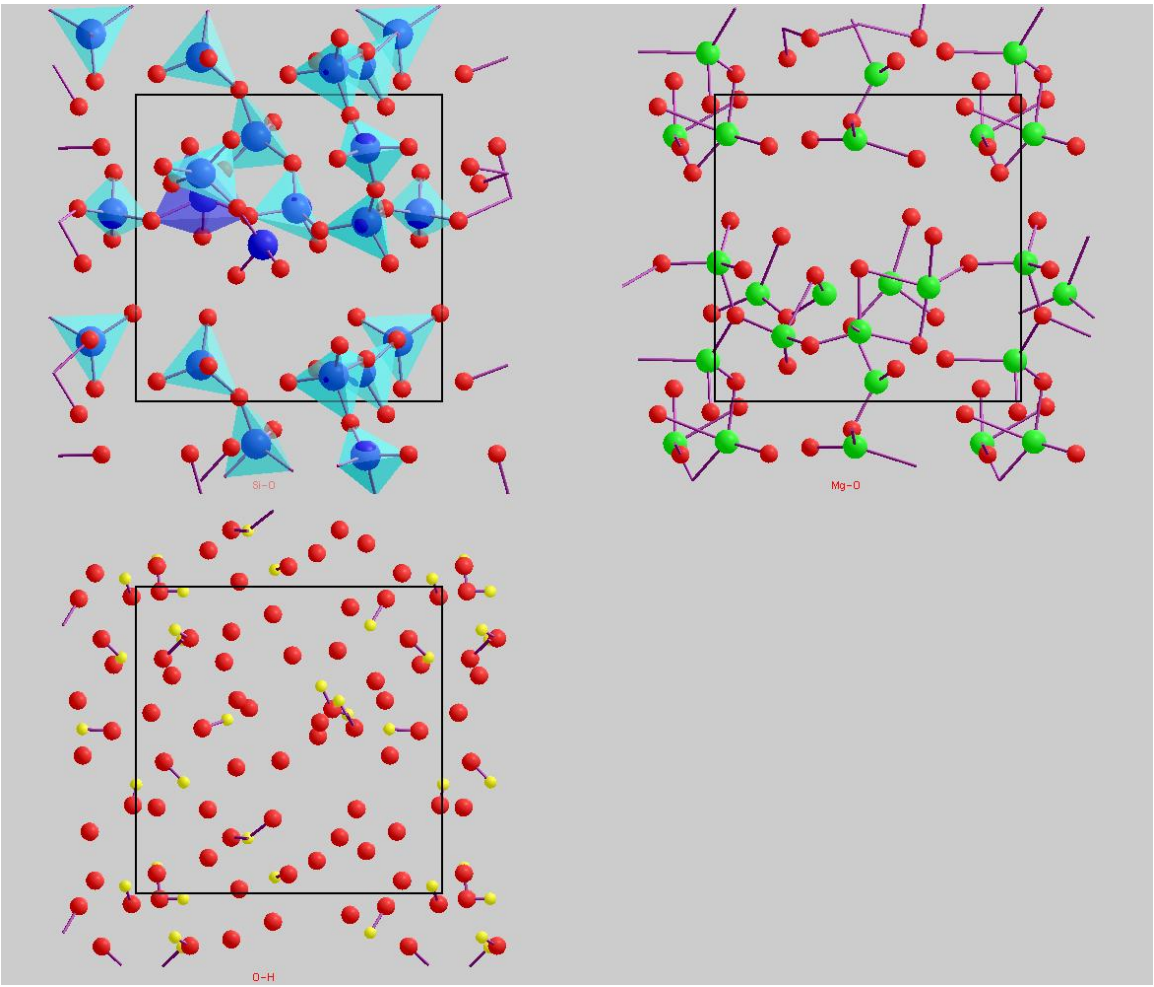

Figure 7-3: Selected coordination environments

Let *w* and *h* are width and height of the window screen on which we want to map λ pairs of coordination environments. If λ is a perfect square, such as 1 or 4, then we calculate a factor, *f*, by taking square root of λ as $\sqrt{\lambda}$. If λ is not a perfect square, then we calculate the factor, *f*, as $\lfloor\sqrt{\lambda} + 0.5\rfloor$. Then, we divide the screen along both width and height into f parts of $\frac{w}{f}$ and $\frac{h}{f}$ width and height. Then λ selected pairs of species are mapped into individual viewports as created above. In the case when λ is not a perfect square, some of the created viewports remain empty because we create more viewports than necessary.

7.6 Performance Optimizations

In cinematographic and game animation, there is a need for real time frame rates because the purpose is creation of an illusion of the real world. Bump mapping [96] can be taken as one example of the way illusion of reality is created in computer graphics. Surfaces that look smooth

on first glance are rougher when examined closely. Representing such roughness using finer geometrical modeling is impractical for interactive rendering. Bump mapping creates an illusion of surface roughness by perturbing the surface normal as specified in a texture, commonly called normal map. Silhouettes of shapes created using bum mapping are smooth, contrary to the real rough surfaces. On the other hand, the purpose of scientific visualization is not necessarily to create an illusion. Rather it is to pass more information as faithfully and efficiently as possible to the user and the visualization is more effective when user is allowed to explore the visual outputs. So, how effective is higher frame rate in fulfilling the true purpose of scientific visualization? The visualization system enables user to explore dataset quickly by providing higher frame rates. This further enables better understanding of the dataset and helps fulfilling the objective.

We took various optimization measures to ensure that the visualization system is interactive and responsive. For example, both the analytical and the visualization modules cache their previously computed results until some parameter is changed avoiding the unnecessary computation. As the performance measurements show some algorithms are computationally expensive and such computations are done in separate threads to make the system more interactive. There is addition overhead in running multiple threads, but the benefit of increased responsiveness outweighs the extra overhead. Along with the increase in responsiveness, use of extra thread allows the visualization outputs to be updated as new data from the computation becomes available. For example, in case of RDF computation, the RDFs converge after a certain point and visualization performed using the incomplete RDF after this point is practically no different than from the visualization done using the complete RDF available only after the whole computation is completed.

7.6.1 RDF Computation

Following two optimization schemes are utilized to accelerate RDF computation and increase interactivity.

7.6.1.1 Step Skipping

The atomic positions are highly time-correlated and from one time step to another time step they remain practically constant. This is also influenced by the very small Δt that is used during the numerical integration in molecular dynamics to stabilize the simulation process and to

make the outputs consistent. Studies have shown that this value has to be less than or equal to 1 femtosecond ($10^{-15}s$) to achieve the numerical stability. Such high time-correlation and small time step allows for step skipping during RDF computation while keeping the finally computed RDF essentially same (Figure 7-4). The computation time is improved proportional to the number of snapshots skipped.

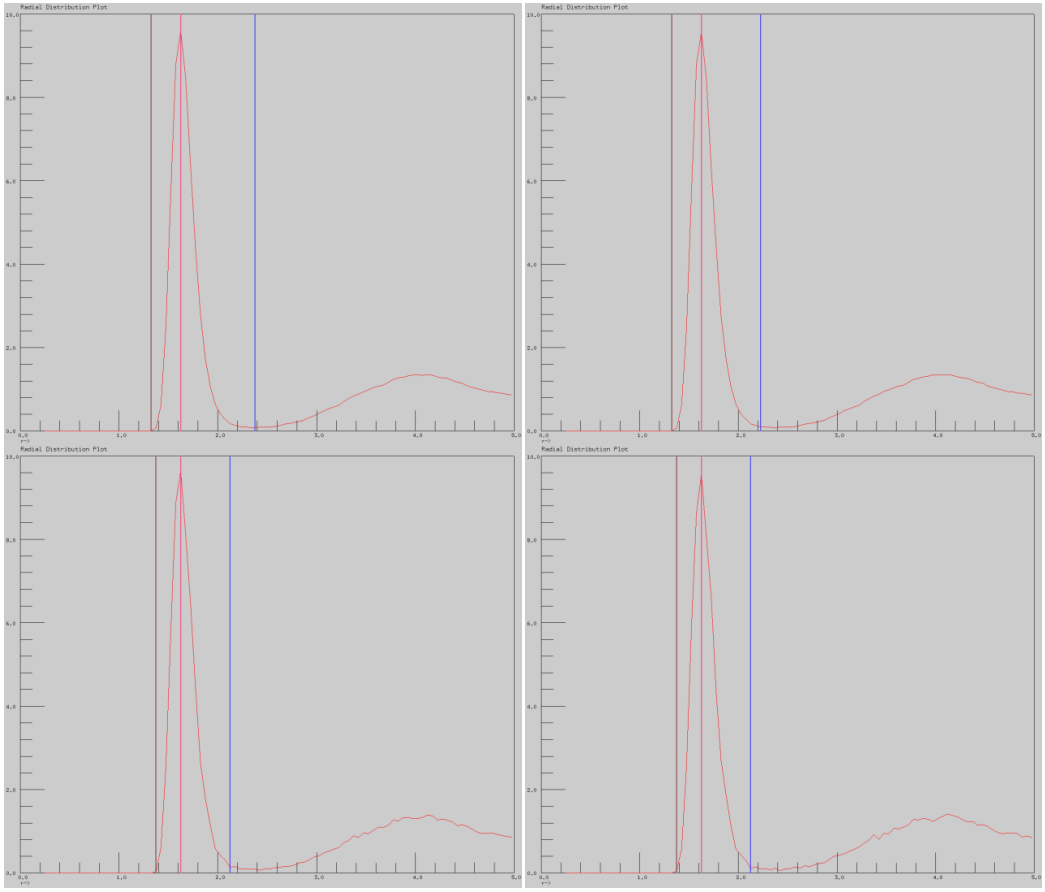

Figure 7-4: RDF Plots (clockwise from top left) no steps skipped, 10 steps skipped, 50 steps skipped, 100 steps skipped

7.6.1.2 Selective Partial RDF Computation

All partial RDF are not equal from information content perspective and small subset of partial RDFs more important for meaningful analysis and visualization than other partial RDFs. Because of this, computation of the complete set of partial RDFs is really unnecessary. We provide options to either compute the complete set of partial RDFs or selectively compute a certain subset of partial RDFs. Computation of selected set of PRDFs reduces the amount of unnecessary computation.

7.6.2 Coordination Environment

Visualization of the coordination environments and their stability is important, but also computationally intensive as shown by the performance timings. So, a special data structure is designed to reduce the unnecessary recomputation. Such a data structure enables exploration of various properties related to the coordination environment without extensive overhead of recomputation.

7.6.2.1 Complete Coordination Environment

The coordination environments for each pair of species, α and β , are stored in separate data structures, $ce^{i}_{\alpha\beta}(t_1, t_2, r_1, r_2)$, where $i \in N_\alpha$. Content of the datastructure depends on the range of time steps, $t_1$and t_2, and range of cutoff distances, $r_1$and r_2. The data structure contains complete information about coordination environment between a pair of species including stability of the coordination, coordination cluster and coordination distribution. The content of the data structure remains constant as long as the range of selected time steps and the range of cutoffs used for the coordination environment computation remain unchanged. If either the range of selected time steps or the range of cutoffs is changed, the content of the data structure also changes and has to be updated. As long as they remain same from one visualization session to another session, coordination distribution and stabilities are directly computed from the data structure. Now, let us formally define the data structure for an atom i of α species with respect to β species.

Equation 7-5

$$ce^{i}_{\alpha\beta}(t_1, t_2, r_1, r_2) = (J, D)$$

where,

$$J = \left\{(j \in N_\beta, T)\middle|\exists t\ s.t. r_1 \leq d\left(p_i(t), p_j(t) \leq r_2\right\}\right.,$$

$$T = \left\{t_1 \leq t \leq t_2\middle|r_1 \leq d\left(p_i(t), p_j(t)\right) \leq r_2\right\},$$

and

$$D = \{(m, t_m)\}.$$

J is the set of pair of atom of β, j, species that is coordinated with the atom i of α species at least once within the selected time range and the set of times, T, where the atoms was coordinated with atom i. This set contains the coordination cluster of the atom i. D is set of pairs of numbers where first element, m, is the coordination number and the second element, t_m, is the total time the atom i had m coordination number. So, D effectively contains the coordination stability for the atom i.

Now, let us say we have to calculate bond stability between the atom i and another atom j. Let us say the total selected time range be τ (=$t_2 - t_1$), and $|T|$ is the total time the atom j was bonded with atom i given by the corresponding T set in J, then the bond stability is between atom i and atom j is given as:

Equation 7-6

$$\frac{|T|}{\tau}$$

We can also calculate the coordination number of atom i for current time, t, as follows:

Equation 7-7

$$cn^i(t) = \left|\left\{j \mid t \in T \; where \; T \in (ce^i.J)_j\right\}\right|$$

Here, $(ce^i.J)_j$ is the set J for the atom j.

7.6.3 Ring Structure

The computation of the primitive rings set for all the atoms over all the selected time steps is time consuming as shown by the performance timing data for the ring structure algorithm. The primitive ring structure set is computed for a selected atom only. To further reduce the computation time, only the rings of selected sizes are computed. When only the shorter rings are of interest, computation time decreases substantially.

Chapter 8 Application Studies

Results of application of the visualization system are presented here. Structural and dynamical behaviors of geophysically relevant silicate and oxide liquids are visually analyzed to understand the effects of pressure and temperature. We focus on atomic radial distribution functions, coordination environments, cluster structures and diffusion patterns.

The datasets were generated by a parallel FPMD application, VASP. Simulations were run on Supermike Linux Cluster at Louisiana State University using 32 to 128 processors (http://www.cct.lsu.edu). The datasets are time dependent three-dimensional atomic position sets. The visualization system was improved after the detail visual analyses of Magnesium Oxide and Magnesium Silicate system were done (Section 8.1 and Section 8.2) and reported in [97]. Visual analyses in later sections were done using the improved system. The new system offers more flexibility, provides additional analytical methods and features mechanisms to save the states of the system as described in the previous sections. We are preparing the visual analyses of the hydrous magnesium silicate system for publication.

8.1 Liquid Magnesium Oxide

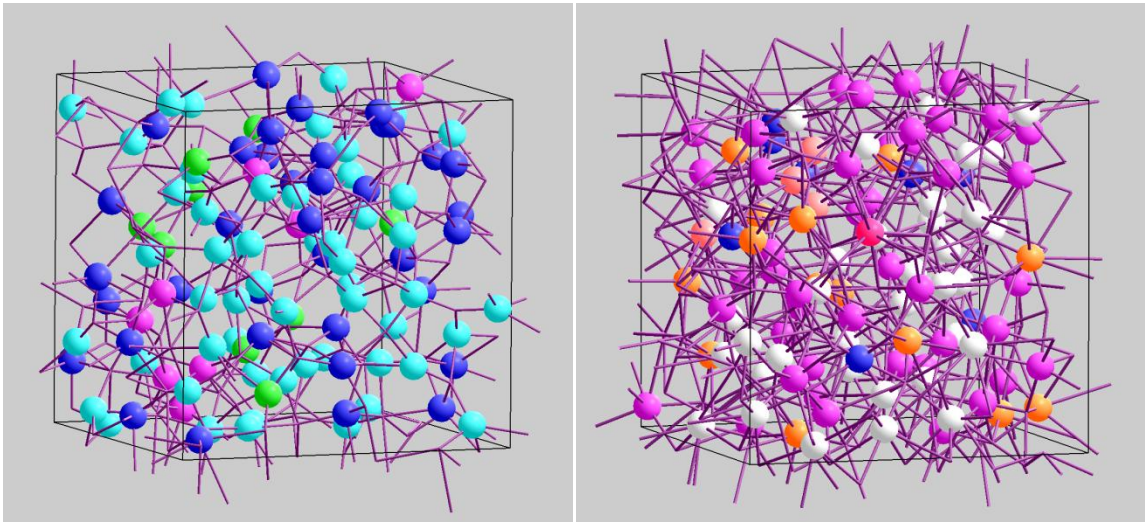

Figure 8-1: Spatial distribution of Mg-O coordination number. (Left) low compression (Right) high compression

Liquid Magnesium Oxide data from simulation using a supercell of 216 atoms for 3 picoseconds contains 3000 time steps. Two sets of Magnesium Oxide liquid data corresponding to two conditions of V1 = 26.96 $Å^3$, T1 = 3000 K (condition V1T1) and V2 = 14.33 $Å^3$, T2 = 9000 K (condition V2T2) are visualized.

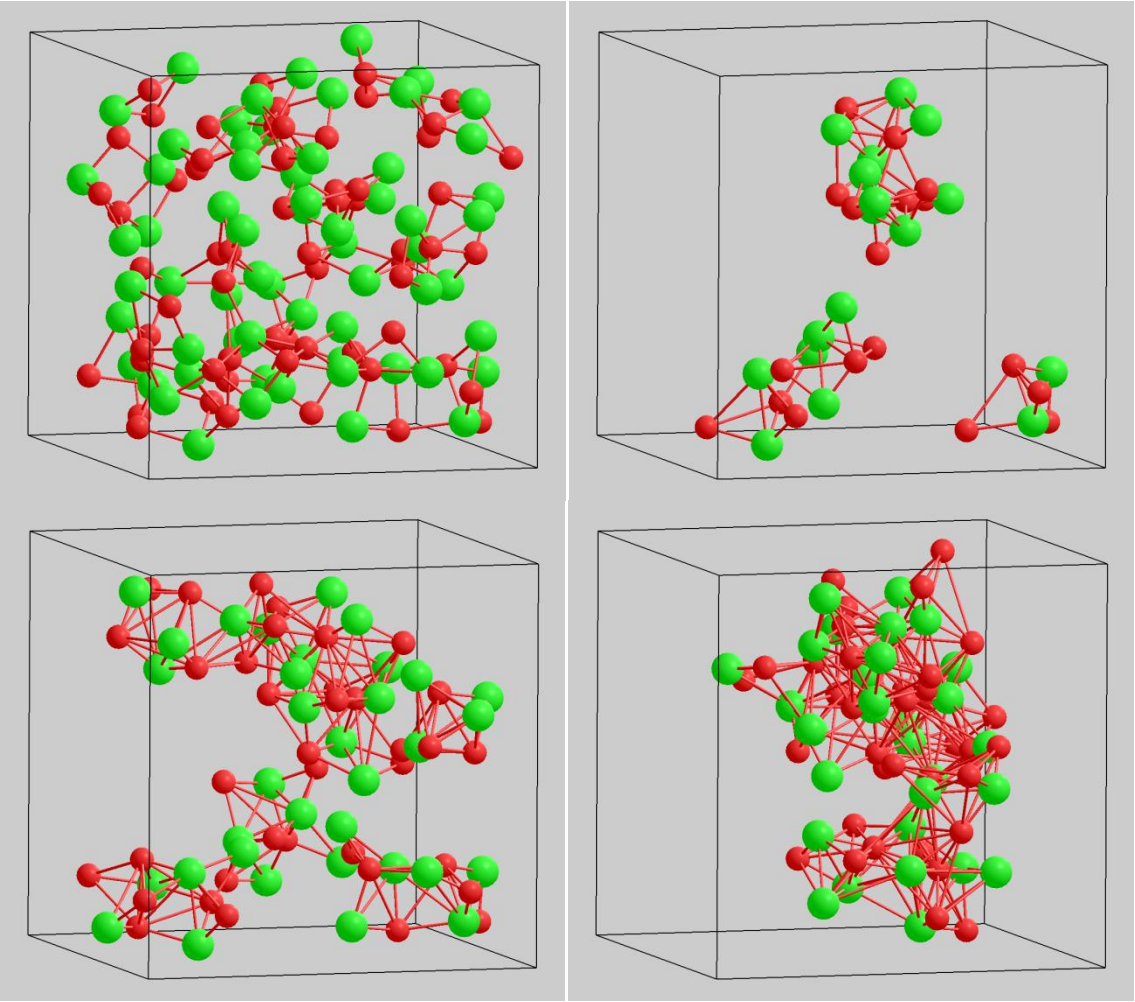

Figure 8-2: Distance dependence of common neighbor clusters: 0.175L (top left), 0.200L (top right) 0.225L (bottom left) and 0.250L (bottom right) at condition V1T1

We visualize the atomic coordination environment of liquid Magnesium Oxide using the data for 216-atom supercell at two conditions (V1T1) and (V2T2) using the coordination-encoded spheres (Figure 8-1). Visualization implies that the coordination environment is not uniform throughout the system and relative contributions of different types (which range from 3-fold to 9-fold) of coordination vary to some extent over the time. Out of 108 Magnesium atoms, there are 9 three- (green), 52 four- (cyan), 39 five- (blue) and 8 six- (pink) fold coordinated

Magnesium atoms at low compression (condition V1T1). At high compression (condition V2T2), there are 8 five- (blue), 45 six- (pink), 35 seven- (white), 14 eight- (orange), 4 nine- (coral) and 1 ten- (deep pink) fold coordinated Magnesium atoms. Compression is shown to suppress the low Mg-O coordination contributions (3-, 4- and 5-fold coordination) and enhance high Mg-O coordination contributions (6-, 7- and 8-fold coordination).

To understand cluster structure, we extract/visualize the common neighbor (CN) clusters as a function of the cutoff distance (r_C). In a 216-atom crystalline Magnesium Oxide cubic supercell of dimension L, the minimum Mg-O distance is $L/6$, and the minimum O-O and Mg-Mg distances both are $(\sqrt{2}/6)L$. For $r_C \geq L/6$ and $< (\sqrt{2}/6)L$, there exist pairs only with two common neighbors. When $r_C \geq (\sqrt{2}/6)L$, higher-order clusters with 2, 3, 4, 6, and 8 common neighbors exist. However in the liquid state, the largest number of common neighbors an atomic pair under consideration can have varies gradually from two to eight or higher value as the cutoff distance increases from $0.15L$ to $0.25L$, as shown in Figure 8-2. A similar trend is observed at condition V2T2 except that the largest number of the common neighbors is less by one than that at condition V1T1 on average. The position of the first minimum of the total RDF is $0.177L$ at low compression so the relevant clusters contain mostly 2 and 3 common neighbors. On the other hand, the first minimum is $0.264L$ at high compression so the relevant clusters contain as many as 9 common neighbors.

8.2 Liquid Magnesium Silicate

Liquid Magnesium Silicate data are for an 80-atom supercell and the simulation was run for 3 ps. Three sets of the silicate data corresponding to three conditions of V1 = 64.60 $Å^3$, T1 = 3000 K (condition V1T1); V2 = 45.22 $Å^3$, T2 = 4000 K (condition V2T2) and V3 = 32.30 $Å^3$, T3 = 6000 K (condition V3T3) are visualized.

Spatial variation of Si–O and Mg–O coordination in the silicate liquid is visualized. At low compression (condition V1T1), the Magnesium Silicate liquid shows mostly fourfold Si–O coordination, which is characteristic to the corresponding crystalline pyroxene phase. Visualization output reveals that there are 15 red (four-fold) and 1 yellow (five-fold) polygons (Figure 8-3, left). Note that there is a total of 16 Silicon polyhedra. However, the liquid structure lacks the elements of longer ranged order seen in the crystal and isolated polyhedra exist. The number of fourfold coordination decreases whereas the number of five- and six-fold coordination

increases with increasing compression. At medium compression (condition V2T2), the average value is about 5, compared to the value of 4.5 for majorite - a high-pressure crystalline polymorph of the silicate. Three types of Si–O coordination (four-, five- and six-fold) exist in almost equal amount; the snapshot showing 6 red, 6 yellow and 4 green polyhedra (Figure 8-3, center). Contribution of the five-fold coordination reaches its maximum at this compression. At high compression (condition V3T3), the Si–O coordination is mostly six-fold, which is characteristic of perovskite and post-perovskite phases [53, 98]. The Silicate liquid shows a few six-fold including both five- and sevenfold coordinated silicon atoms. Visualization snapshot shows 12 green (six-fold), 1 yellow (fivefold) and 3 cyan (sevenfold) polyhedra (Figure 8-3, right). It also reveals that no polyhedron exists in isolation. However, there is no complete 3D framework of corner sharing octahedra as in the crystalline phase in which every octahedron shares all of its corners with six other octahedra. Despite the absence of a complete corner-sharing, some polyhedra in the liquid share edges and some even share faces with each other. In this sense, the liquid structure seems to be more densely packed.

Figure 8-4 visualizes polyhedral distortion using the quadratic elongation parameter for the silicate liquid at three conditions (V1T1, V2T2 and V3T3). The individual polyhedra show different degrees of distortion as shown by their colors. At low and medium compression (V1T1 and V2T2), most polyhedra (black and red colored) show weak distortions whereas at high compression (V3T3), there are relatively few black and red colored polyhedra.

Mg–O coordination environment in silicate liquid also shows a similar trend (Figure 8-5). At low compression, the coordination contains contributions from four-fold, five-fold and six-fold coordination. At medium compression, the contributions are from seven-, eight- and nine-fold coordination, with the number of sevenfold coordinated Mg atoms being the maximum. Finally, at high compression, the number of cyan spheres (seven-fold) decreases whereas that of blue (eight-fold) and magenta (nine-fold) spheres increases. We can even see some white spheres (ten-fold).

The NN clusters for the silicate liquid at condition V2T2 are extracted with three cutoff distances of r_{peak}, r_{avg} and r_{min} obtained from the partial Si–O RDF (Figure 8-6). With r_{peak} and r_{avg}, individual clusters are small consisting of only one Si atom and some O atoms. For the cutoff distance lying between r_{avg}, and r_{min}, larger clusters consisting of more than one Si atoms

are found. Finally, there is a single NN cluster for r_{min}, which contains not only Si–O bonds but also contains some Mg–O, Mg–Si and O–O pairs.

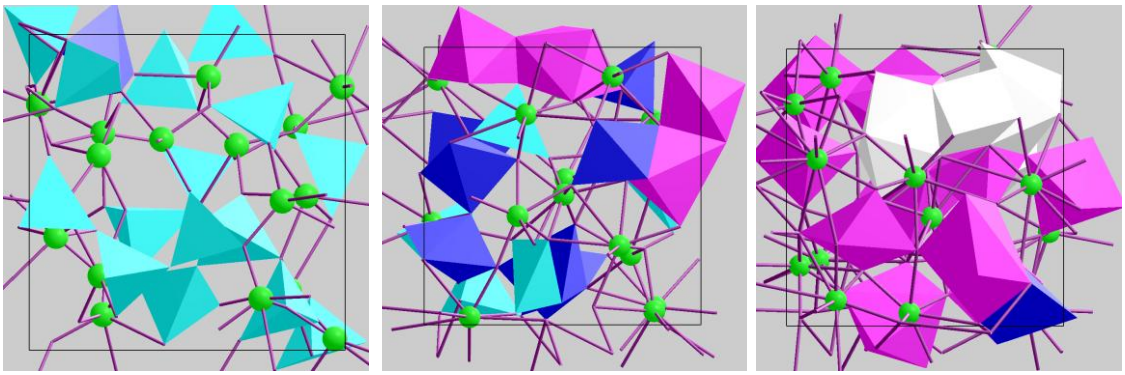

Figure 8-3: Polyhedral representation of Si-O coordination environment in the silicate liquid at low (V1T1), medium (V2T2) and high (V3T3) compression from left to right.

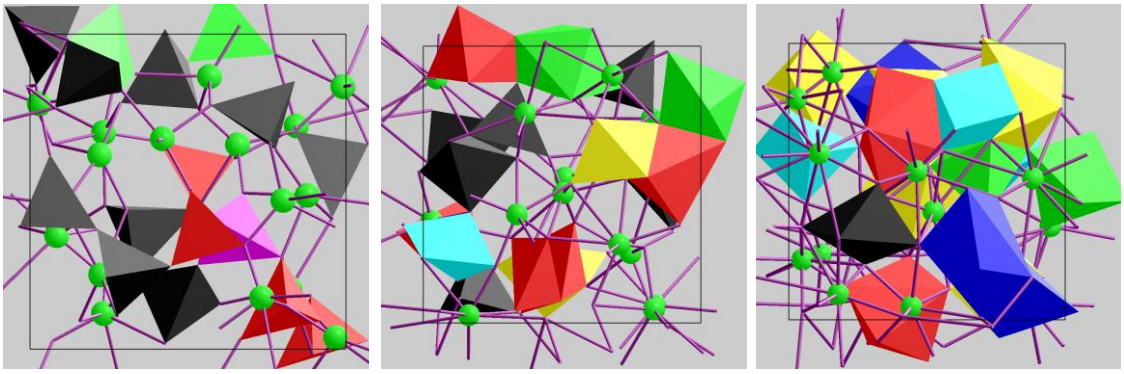

Figure 8-4: Visualization of the quadratic elongation (λ_p) for liquid silicate at low (V1T1), medium (V2T2) and high (V3T3) compression from left to right. Note that min = 1.0 and max = 1.04 for λ_p

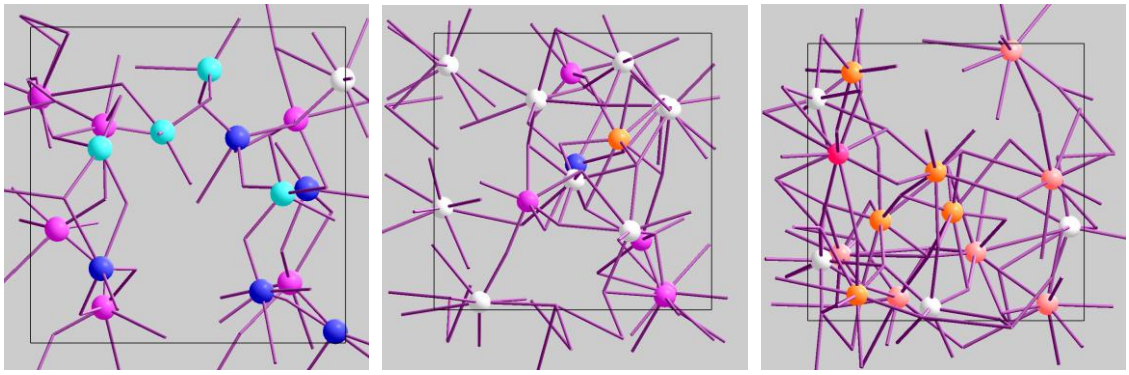

Figure 8-5: Sphere representation of Mg-O coordination environment in the silicate liquid at low (V1T1), medium (V2T2) and high (V3T3) compression from left to right.

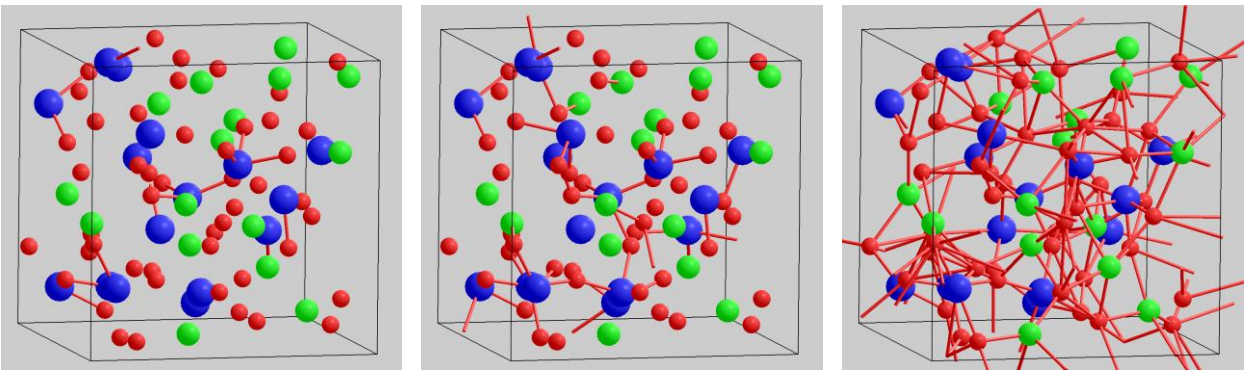

Figure 8-6: Nearest-neighbor (NN) clusters in silicate liquid using r_{peak} (left), r_{avg} (center) and r_{min} (right) at condition V2T2

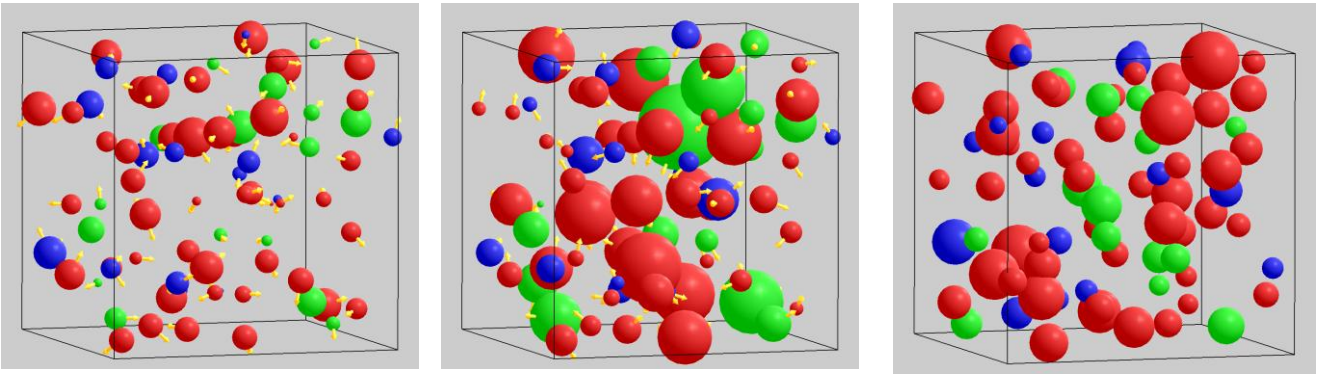

Figure 8-7: Diffusion spheres for liquid silicate at medium compression (condition V2T2). Left: instantaneous displacement pattern (scaling factor = 15), Center: displacement pattern over 500 time steps (scaling factor = 0.4), Right: centroid spheres over 2000 time steps (scaling factor = 0.2).

Diffusion is also an important dynamical phenomenon for liquid phase. Simulations have shown that for liquids temperature systematically enhances diffusion whereas pressure systematically suppresses it. Atomic diffusion pattern is explored by visualizing three types of displacement data (Figure 8-7). They are instantaneous (positional differences between the present and previous steps), finite-time (positional differences between 2,500 and 3,000 steps) and maximum (differences between farthest and mean positions) displacement data. Visualization reveals that diffusion is not uniform throughout the system (i.e., among the constituent atoms of the same or different types). The spheres have different sizes and the arrows showing the direction of atomic displacements are oriented along different directions. The red spheres (Oxygen atoms) are on average larger than the green (Magnesium atoms) and blue (Silicon atoms) spheres; and the blue spheres are the smallest ones. This means that the Oxygen

atoms diffuse most among the three elements. Variation in the size of the centroid spheres is systematically much smaller than that of the instantaneous and finite-time spheres.

8.3 Hydrous Magnesium Silicate system

The simulation box contains 12 $MgSiO_3$ units and 8 water molecules (the total number of 84 atoms) for hydrous phase and 16 $MgSiO_3$ units (total of 80 atoms) for anhydrous phase. The low water content system simulated only at the ambient pressure has 12 $MgSiO_3$ units and 4 water molecules (total of 72 atoms). Time steps of 0.5 and 1 femtosecond are used for hydrous and anhydrous phases, respectively. A series of simulations of high water content system have been performed covering six volumes: V/V_X = 1.0, 0.9, 0.8, 0.7, 0.6 and 0.5, V_X = 1,033 $Å^3$ is the reference volume of the simulation cell used for both hydrous and anhydrous melts. The initial structure at each volume is first melted and equilibrated at 6,000 K for a period of 3 picoseconds, and then cooled isochorically to first 4,000 K and then to 3,000 K. In this study, the system at V_X is further cooled down to 2500 K and then finally to 2000 K. The total durations at V_X are 12, 24, 36 and 90 picoseconds for 4000, 3000, 2500 and 2000 K, respectively. A relatively longer runs are performed as the liquid is compressed. For instance, the total run durations at $0.5V_X$ are 12, 36 and 150 picoseconds at 6000, 4000 and 3000 K, respectively. Similarly, the simulations of anhydrous melt are performed for much longer durations at the selected conditions (three volumes of V_X, $0.7V_X$ (4000 K and 3000 K) and $0.5V_X$ (6000 and 4000 K)

8.3.1 Hydrogen and Oxygen Coordination Species

Our understanding of speciation of the dissolved water component of silicate melt requires a detailed knowledge of bonding environments involving hydrogen and oxygen, which can be characterized by coordination parameters ${}_ZC_{HO}$ and ${}_ZC_{OH}$. The effects of compression on relative abundances of different species for both coordination types are weakly dependent on temperature (Figure 8-8). In the case of H-O coordination, as the liquid is compressed from V_X to $0.5V_X$ at 3000 K, ${}_1C_{HO}$ (one-fold coordination) decreases rapidly from 80.9 to 17.4 % whereas ${}_2C_{HO}$ (two-fold coordination) increases rapidly from 19.0 to 72.3 %. The crossover occurs around $0.7V_X$ such that ${}_1C_{HO}$ dominates at larger volumes whereas ${}_2C_{HO}$ dominates at smaller volumes. The proportion of ${}_3C_{HO}$ (three-fold coordination), which is negligible (0.03 %) at V_X, becomes significant at compressed volumes reaching as high as 10.1 % at $0.5V_X$. Unlike higher order coordination species, the value of ${}_0C_{HO}$ is negligible at all compression (0.04 and 0.00 %,

respectively, at V_X and $0.5V_X$ for 3000 K); the maximum contribution being 1.33 %, which is at 4000 K and V_X. This means that there are essentially no free hydrogen atoms (i.e., almost all of 16 hydrogen atoms are bonded to oxygen). On the other hand, in the case of the O-H coordination, the value of ${}_0C_{OH}$ (zero-fold coordination) is large and it actually far dominates other species at all conditions. This means that not all oxygen atoms are bonded with hydrogen atoms although the opposite is not true. Compression forces more oxygen to participate in hydrogen bonding as shown by a decreasing value of ${}_0C_{OH}$ from 63.0 % at V_X to 46.9 % at $0.5V_X$ for 3000 K (Figure 8-8, Right). About 16.28 oxygens at V_X and 23.36 Oxygen at $0.5V_X$ are bonded with hydrogen. Since the amount of oxygen involved is equal or higher than the total number of hydrogen (16 atoms) present in the system, water speciation cannot be a homogeneous distribution of only hydroxyls and water molecules; other forms of speciation must be present. The enhanced participation of oxygen results in gradual increase of the proportions of the ${}_1C_{OH}$ and ${}_2C_{OH}$ species on compression. Their values are, respectively, 31.0 and 5.65 % at V_X, and 38.3 and 12.5 % at $0.5V_X$ for 3000 K. Even three-fold coordination (${}_3C_{OH}$) species almost absent (0.3 %) at V_X becomes significant at high compression (2.1 % at $0.5V_X$).

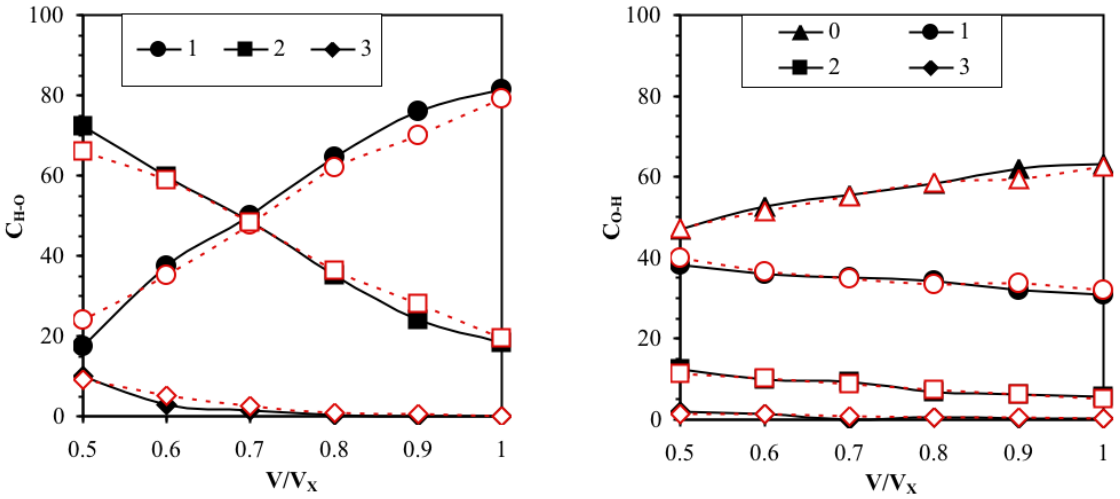

Figure 8-8: Abundances of different species of H-O and O-H coordination as a function of compression and temperature.

Figure 8-9 shows the snapshots for the coordination species of the H-O and O-H coordination at V_X and $0.5V_X$ for 3000 K. The hydrogen atoms (small spheres) are mostly red (one-fold coordinated) at V_X whereas they are mostly yellow (two-fold coordinated) at $0.5V_X$. Also note one green sphere for hydrogen in three-fold coordination state for the compressed

volume. Many oxygen atoms (large black spheres) are free at both conditions. More yellow (large) spheres and even green (large) sphere imply the dominance of higher coordination O-H species at high compression.

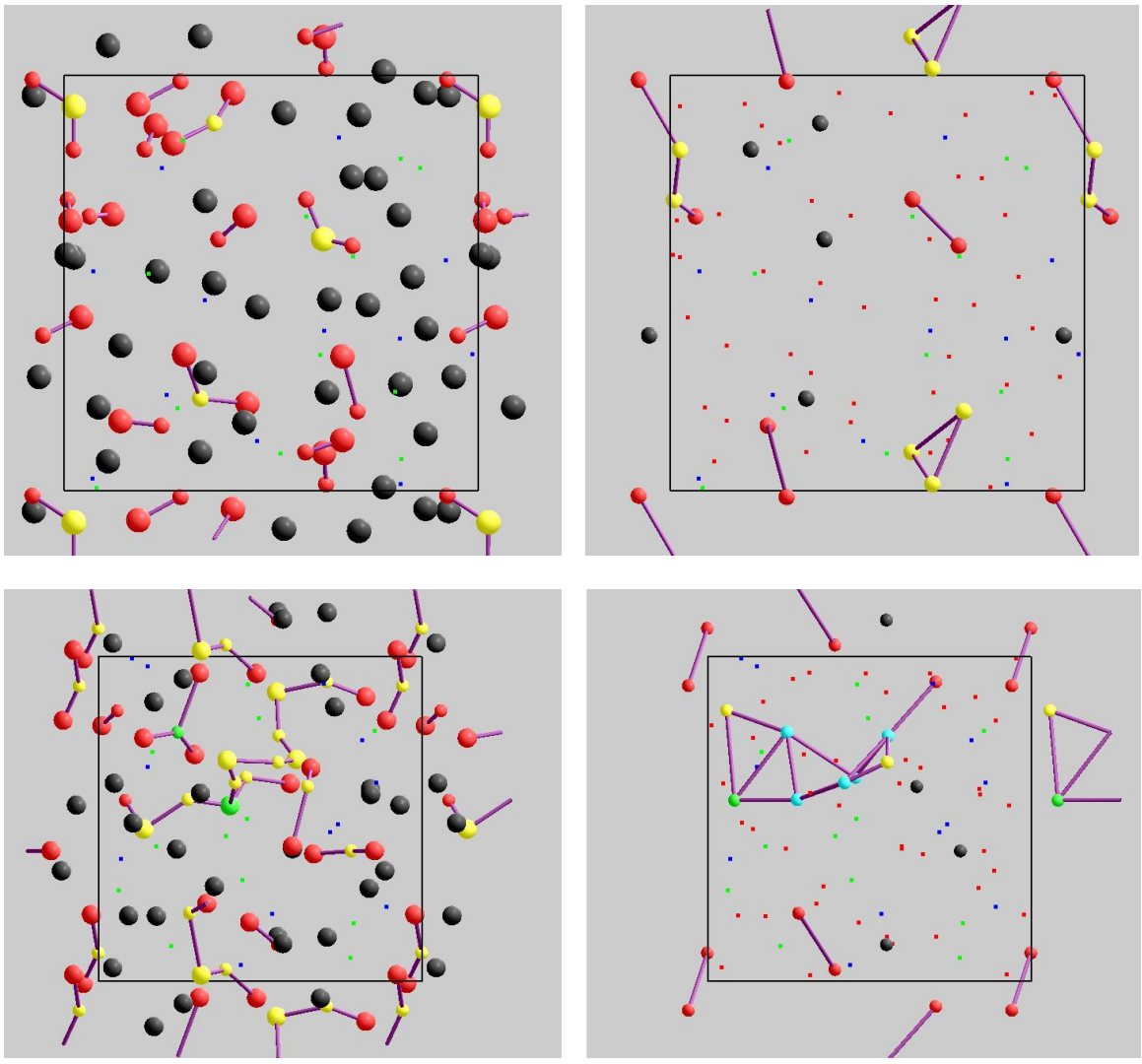

Figure 8-9: Snapshots of mixed H-O and O-H coordination (left) and H-H coordination (right) at V_X (upper) and $0.5V_X$ (lower) for 3000 K.

Also relevant in water speciation is H-H coordination. Since the H-H radial distribution function does not show a clear minimum, we choose a cutoff of 2.2 Å which lies the near the right base of the peak (at V_X and 3000 K). For this cutoff, more than half (54 %) of hydrogen atoms are not coordinated with any other hydrogen. The values of ${}_1C_{HH}$ and ${}_2C_{HH}$ are 36 and 9 %, respectively. There is a small proportion (1.2 %) three-fold coordination species. Dominance of low coordination species means that H-H correlation is significant only at the level of individual

structural units. In Figure 8-9, the bonds are shown between two H-atoms, which are within the cutoff distance, and the color of H atom indicates its coordination state. At high compression ($0.5V_X$ and 3000 K), a larger cutoff of 2.4 Å which is the near the right base of the peak is used. A relatively few H atoms (13 %, compared to 54 % at V_X) are not coordinated with other H atoms. About 29 % H atoms are one-fold coordinated. Both the two- and three-fold coordination is dramatically enhanced (with 26 % ${}_2C_{HH}$ and 18 % ${}_3C_{HH}$). Even four- and five-fold coordination species exist in considerable amounts (9 and 3 %, respectively). The existence of higher coordination species implies that H atoms tend to form large clusters at high compression, compared to isolated small structures at the equilibrium volume.

8.3.2 Water Speciation

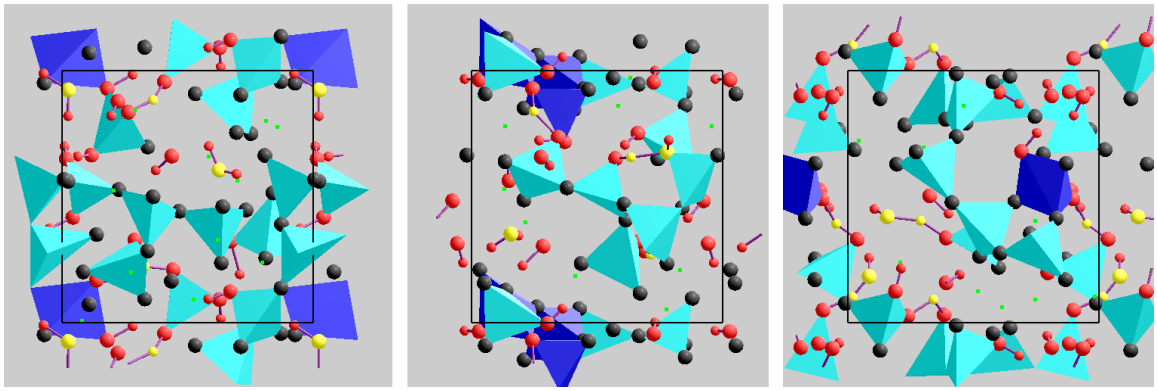

Figure 8-10: Snapshots of water speciation in relation with H and O coordination at VX and 3000 K. Hydroxyls, water molecules, polyhedral bridging, edge decoration, and 4-atom sequences.

We can identify a rich variety of water speciation in terms of H-O bonds through the interpretation of the calculated oxygen and hydrogen coordination environments (Figure 8-10). We first discuss the implications of the H-O coordination (${}_ZC_{HO}$). Since the value of coordination ${}_0C_{HO}$ is almost zero, our analysis assumes that every hydrogen atom participates in H-O bonding of one or more of species, which can vary from isolated units to extended structures but the speciation in the form of free protons does not exist. We express a given speciation in terms of the total number of oxygen and hydrogen atoms present in it. Thus, molecular water or O-H-O bridging may also be referred to as a three-atom group. Similarly, a four-atom group may be H-O-H-O, O-H-O-H, H-O=H (a hydronium group) or O-H=O (a three-fold coordinated hydrogen). In these structures, oxygen may be a part of polyhedron (i.e., polyhedral oxygen) or only bound to magnesium (i.e., non-polyhedral or free oxygen).

8.3.2.1 V_0 and 3000 K

The most abundant (80.9 %) ${}_1C_{HO}$ coordination contributes to two types of the water speciation. The first type is hydroxyls, which can be classified into two groups. The first group involves bonding between hydrogen and polyhedral oxygen, in other words, the hydroxyl is bound to a silicon atom (Si-OH). A polyhedral oxygen atom may also be bonded to Mg atoms (i.e., it may lie within the Mg-O cutoff distance). We call this type of speciation as polyhedral hydroxyl. Our results show that the non-bridging oxygen sites are heavily preferred over the bridging oxygen sites. The second group consists of isolated hydroxyls called non-polyhedral (free) hydroxyls, which are bonded to only magnesium atoms (Mg-OH) without any polyhedral involvement. The amount of polyhedral hydroxyls is somewhat higher (by about 10 %) than that of non-polyhedral hydroxyls. The one-fold coordination also contributes to the formation of water molecules since two hydrogen atoms, each coordinated with the same oxygen atom represent a water molecule. Like hydroxyl, a water molecule can be polyhedral water or non-polyhedral (free) water. Our results show that almost all (~95 %) water molecules are non-polyhedral (i.e., bound to Mg atoms). Finally, singly coordinated hydrogen also exists at the ends (or dead-ends) of extended structures (e.g., H-O-H-O or H-O-H-O-H, where oxygen may be a polyhedral or non-polyhedral type).

The proportion of two-fold coordination species (${}_2C_{HO}$) is 19.0 %, much smaller than that of ${}_1C_{HO}$. A given hydrogen atom bridges two neighboring polyhedra by forming bonds with oxygen atoms, one from each polyhedron, that is, forming a Si-O-H-O-Si sequence. This can be referred to as polyhedral bridging (Figure 8-10). Alternative to this is an intra-polyhedral linkage in which a hydrogen atom forms bonds with two oxygen atoms from the same polyhedron – referred to as polyhedral edge decoration (Figure 8-10, Center), which was previously suggested by the experimental study of hydrous glass [99]. However, our results show that edge decoration is much less prevalent than bridging. The reason is that any two oxygen atoms on a polyhedral (triangular) face are too far apart to share the same hydrogen atom since the most probable O-O distance (2.685 Å) is more than twice the most probable H-O distance (1.005 Å). Even if the O-O distance happens to be smaller than the twice H-O distance in some cases, hydrogen atom cannot be arbitrarily close to the O-O edge because of silicon and hydrogen cationic repulsions. Note that the most probable Si-H distance (2.345 Å) is much larger than the most probable Si-O distance (1.625 Å). However, the O-O distance involved in the polyhedral bridging is more

flexible (to be small enough) to share the same hydrogen atom. Thus, the bridging speciation involves the H-O distances on the right side of the peak and O-O distances on the left side of the peak. These also apply in the other forms of linkage, which comprise of either one or both non-polyhedral oxygen (X-O-H-O-X with X = Mg or Si) (Figure 8-10, Right). The polyhedral bridging (Si-O-H-O-Si) accounts for more than one half (~53 %) whereas the mixed bridging (involving unlike cation pairs) accounts for ~40 % and the non-polyhedral bridging (Mg-O-H-O-Mg) accounts for the remaining small amounts. Finally, the two-fold coordinated hydrogen can appear in the four or more atom (linear) sequences, which are much less abundant in an uncompressed liquid.

Now we consider the implications of the O-H coordination ($_ZC_{OH}$) for water speciation. Almost two-third (63.0 %) oxygen atoms are not bonded to hydrogen, and they all are polyhedral oxygen. The contributing oxygen atoms (37.9 % or 16.28 oxygen at V_X) are either part of polyhedra (rarely bridging oxygen) or bound to Mg atoms. Their contribution to $_1C_{OH}$ coordination is 31.0 %, which represents primarily the polyhedral and non-polyhedral hydroxyl groups. It also accounts for the presence of any bridging and dead-ends of extended structures. A significant number of the O atoms are two-fold coordinated ($_2C_{OH}$ = 5.7 %), which represent isolated water molecules. The Mg-bound (non-polyhedral) water molecules are much more prevalent than polyhedral ones. There are also the structures like Si-O-H-O-H, which may be considered as water molecule hanged to polyhedral oxygen. We find that oxygen atoms with two nearest neighbor H atoms are stable over long durations (of the order of 1 ps).

We can use the information on coordination species in the representation of water speciation in two ways. First, we can color code the coordination values to aid the visual identification of the speciation. According to the color- and size-coding schemes used, sequences of small (hydrogen) and big (oxygen) spheres in red and yellow (and possibly green for three-fold state) colors visually represent different kinds of water speciation (Figure 8-10). A hydroxyl comprises of one O and one H, both atoms being in one-fold coordination state so it appears as a big red sphere-small red sphere pair (simply a red-red pair) in the visualization snapshot. Similarly, a water molecule appears as a small red sphere-big yellow sphere-small red sphere triplet whereas a bridging structure appears as a big red sphere-small yellow sphere-large red sphere triplet. A red-yellow-red triplet represents a three-atom group. Similarly, other structures are represented; for example, a four-atom sequence H-O-H-O appears as small red-big yellow-

small yellow-big red. In any polyhedral association, O appears as a sphere at a polyhedral corner (attached to an Si atom) otherwise it appears as a free sphere (attached to an Mg atom). Second, we can use the coordination numbers to quantify the speciation. Let *x*, *y* and *z* be the amounts of hydroxyls, water molecules and bridging H atoms, respectively, and also consider them as three fundamental forms of speciation so that any extended structure can be viewed as a combination of two or more of them. The proportion of ${}_2C_{HO}$ can be taken as the amount of bridging H atoms, *z* = 19.0 % of the total hydrogen (16 atoms) = 3.04 hydrogen. Similarly, the proportion of ${}_2C_{OH}$ can be taken as the amount of water molecules, *y* = 5.7 % of the total oxygen (44 atoms) = 2.51 oxygen = 5.02 hydrogen since two hydrogen atoms are associated with each water molecule. The value of ${}_1C_{HO}$ represents the sum of hydroxyls and water molecules so *x* + *y* = 80.9 % = 12.94 hydrogen thereby giving the amount of hydroxyls, *x* = 7.92 hydrogen. Thus, 49.5, 31.14 and 19.0 % of the total hydrogen participate in the formation of hydroxyls, water molecules and bridging, respectively. At 2000 K, 46.9, 35.8 and 17.4 % of the total hydrogen participate in the formation of hydroxyls, water molecules and bridging, respectively.

Finally, the preponderance of isolated structures such as hydroxyls, water molecules and bridging is also reflected in the H-H coordination. The very high proportion (90 %) of ${}_0C_{HH}$ and ${}_1C_{HH}$ together suggests that hydrogen is well dispersed in the melt in the form of more or less isolated hydroxyls, water molecules and bridging (Figure 8-9, Top-right). In other words, the water speciation involves a relatively homogeneous distribution of these structures.

8.3.2.2 $0.5V_0$ and 3000 K

All hydrogen atoms participate in bonding at the highest compression as well (${}_0C_{HO}$ = 0) resulting in a wide range of speciation with extended structures dominating (Figure 8-11). The proportion of the one-fold coordination (${}_1C_{HO}$) is significantly reduced (17.4 % or 2.78 hydrogen), compared to the value at low compression. The ${}_1C_{HO}$ coordination contributes to isolated hydroxyls (1.49 hydrogen) with isolated water molecules being extremely rare (0.02 hydrogen), but more importantly, it can be associated with the H-ends of other extended structures.

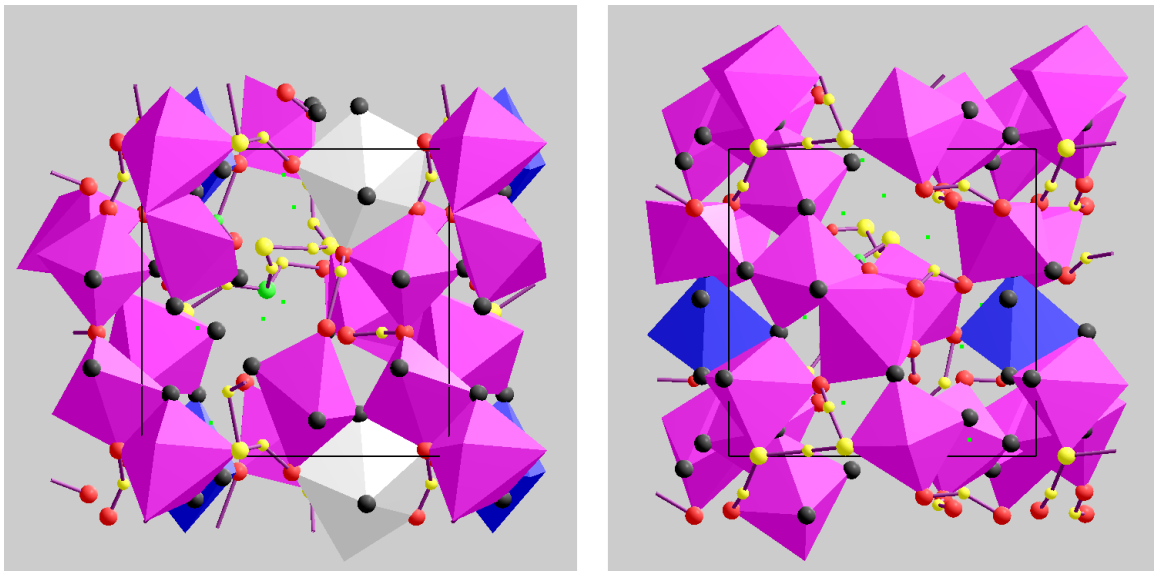

Figure 8-11: Snapshots of water speciation in relation with H and O coordination at compressed volume (0.5 VX and 3000 K). Hydroxyls, polyhedral bridging, edge decoration, and long sequences containing three-fold coordinated O and H atoms are present.

The very large proportion (72.3 % or 11.57 hydrogen) of two-fold coordination species ($_2C_{HO}$) reflects a dramatic enhancement of bridging (X-O-H-O-X, with X = Mg, Si), which occur in isolation (3.29 hydrogen) but, more commonly, in combination (8.28 hydrogen). The isolated polyhedral bridging and edge decoration account for 1.83 hydrogen, with bridging being more common than edge decoration. The extended structures consuming the most hydrogen comprise of two or more of PO-H-PO (pure polyhedral), NPO-H-PO, PO-H-NPO, and NPO-H-NPO bridging, and edge decoration. The extended structures require all H atoms except the ones at the dead ends be two-fold (at least) coordinated. The portion of three-fold coordination is also significant (10.1 % or 1.62 hydrogen) in which the same H atom forms a mixture of bridging and edge decoration. Possible structures include 1) two polyhedral bridging and one edge decoration, 2) two consecutive edge decorations, 3) pure three-way polyhedral bridging, 4) two PO-H-NPO bridging and one edge decoration, and so on. A face decoration in which H atom is coordinated with three O atoms defining a polyhedral face (i.e., a pure three-way edge decoration) is not seen. A three-fold H atom rarely exists in isolation (only 0.21 hydrogen) and its presence causes branching of the extended structures, which comprise of five or more H and O atoms together.

The O-H coordination number shows that more than half (53.1 %) of the O atoms are involved in the water speciation at the highest compression. They include all non-polyhedral O atoms (bound to Mg atoms) and several polyhedral O atoms. Since a significant number of O

atoms still do not participate in H-bonding, more water can be dissolved in the silicate melt. The dominating ${}_1C_{\mathrm{OH}}$ coordination (38.3 %) represents the isolated hydroxyls (1.49 hydrogen). Major contributions also come from the isolated polyhedral bridging and edge decoration and from the O dead ends of extended structures. The dramatically enhanced ${}_2C_{\mathrm{OH}}$ coordination (with 12.5 % abundance) represents rare isolated water molecules (0.02 hydrogen) and, more commonly a part in extended structures. The proportion of ${}_3C_{\mathrm{OH}}$ representing hydronium group is not negligible (2.1 %). This three-fold state mostly involves non-polyhedral oxygen and always occurs in an extended structure causing it's branching (Figure 8-11, Left).

The extended structures may be long enough to run across the entire supercell. (Figure 8-11, Left) shows a fourteen-atom structure, which comprises of seven H atoms and seven O atoms running from the near mid left boundary towards the upper right corner. It starts with the first (one-fold coordinated) H atom (near mid left boundary) attached to a polyhedral O (two-fold coordinated), which is bridged with a NPO by the second (two-fold coordinated) H atom. Branching occurs at this NPO (three-fold coordinated) site with the shorter branch consisting of NPO-H-PO bridging. The other longer branch extends with the NPO-H-NPO-H-PO-H-PO-H-PO bridge sequence. The O atom at the end of the structure (near the upper right corner) is in the one-fold coordination state. In right of Figure 8-11, a seven-atom structure near the bottom left of the super cell is a linear sequence of three consecutive polyhedral bridges. Note that the red large sphere representing the one-fold coordinated oxygen at the beginning and end of the chain.

8.3.2.3 Variation with Compression

As we have discussed above, the water component in the hydrous silicate melt adopts a wide variety of species, which differ substantially between low (V_X) and high ($0.5V_X$) compression. Now, we present the calculated abundances of different forms of water speciation as a function of compression (Figure 8-12). Note that the proportions of three basic forms of speciation (hydroxyl, water molecule and bridging) estimated earlier differ from those directly calculated since the direct calculations explicitly consider bigger structures containing more than three O and H atoms together as different species. At V_X and 3000 K, 58.1, 12.0 and 8.0 % hydrogen form hydroxyls, water and bridging, respectively. About 11.8 and 4.3 % hydrogen contribute to the formation of four-atom structures (H-O-H-O or O-H-O-H) and five-atom structures (O-H-O-H-O and H-O-H-O-H), respectively, whereas the remaining 3.7 % H is

involved in other structures. Also three-fold coordination states are rare (less than 2 % hydrogen). By using the calculated concentrations of hydroxyls, water molecules and free oxygen, we can also estimate the equilibrium constant of the reaction $O + H_2O = 2OH$ as $9.30^2 / (27.83 \times 0.96) = 3.23$. Its value at 2000 K is $8.58^2/(28.39\times 1.49) = 1.74$. The values extrapolated from experimental measurements on rhyolite compositions [100] at much lower temperatures and water contents are 2.3.

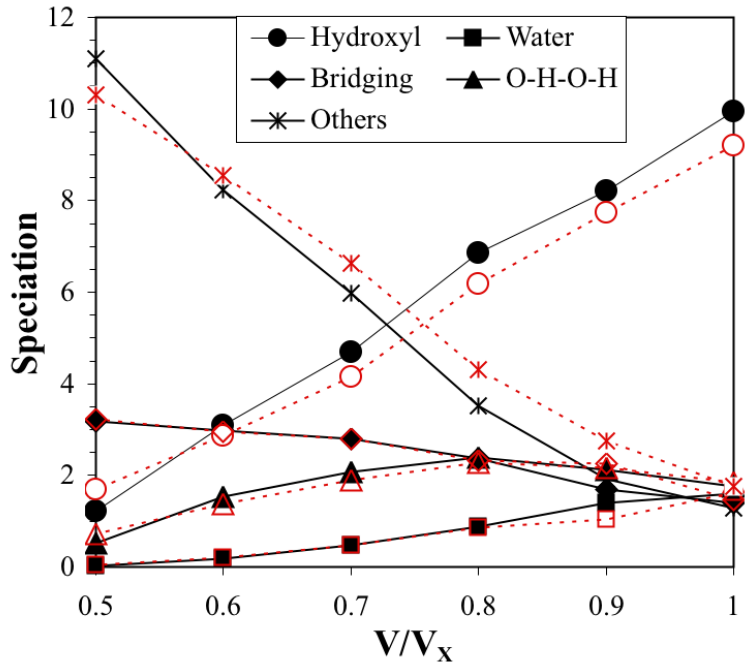

Figure 8-12: Abundances (expressed in terms of the number of hydrogen atoms) of different types of water speciation as a function of compression at 3000 K (solid lines and symbols) and 4000 K (dashed lines and open symbols). Note that the total number of hydrogen atoms in the melt is 16.

With increasing compression, the numbers of isolated hydroxyls and water molecules decrease strongly on compression (7.5 and 0.1 % H, respectively, at $0.5V_X$) whereas the amount of bridging (X-O-H-O-X) increases rapidly on compression (19.8 % at $0.5V_X$), see Figure 15. The fewer water molecules and more bridging imply an increased participation of O atoms in bonding as the liquid is compressed. The amount of linear four-atom sequences (O-H-O-H and H-O-H-O) increases and then decreases on compression remaining significant at all compression. The total abundance of other structures (those containing three-fold coordinated atoms and/or five or more atoms) increases rapidly on compression from 10.1 % at V_X to 69.4 % at $0.5V_X$. The most contribution comes from the structures consisting of more than five atoms (57.8 % at

$0.5V_X$). The four-atom structures comprising of three-fold coordinated O atom or three-fold coordinated H atom are relatively rare so these atoms must contribute to other bigger structures. The isolated three-fold coordinated H atoms account for only 1.5 % hydrogen at $0.5V_X$. Finally, among the five-atom structures, those consisting of three O and two H atoms become more abundant on compression (1.4 to 9.9 % from V_X to $0.5V_X$).

8.3.3 Hydrogen Diffusion Mechanisms

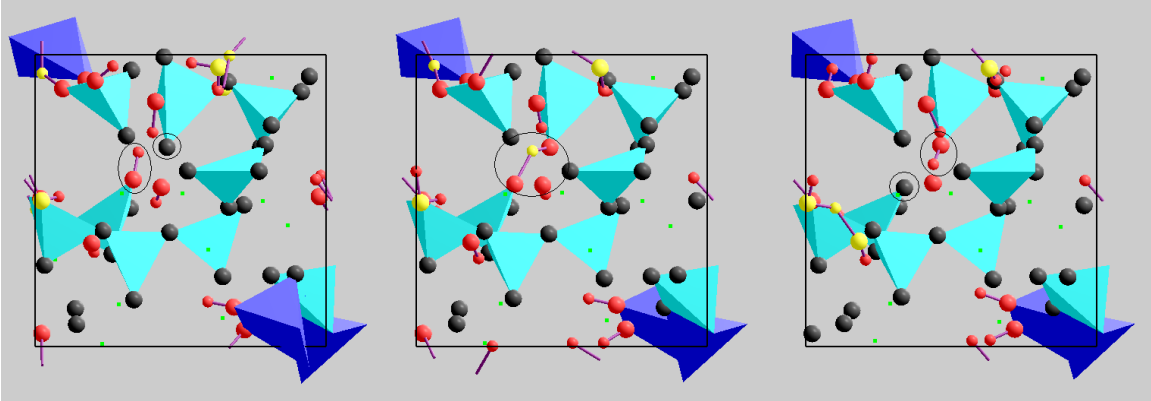

Figure 8-13: Three stages, initial (left), intermediate (center) and final (right), of the H transfer between two non-bridging oxygen atoms marked by circles.

We visualize the position-time series data to gain insight into the possible mechanisms for hydrogen diffusion in hydrous silicate melt. In particular, we want to determine whether various structural units identified for water speciation can serve as transition states for hydrogen diffusion. The existence of structural units including stable hydroxyls and water molecules, bridging (including polyhedral) and edge decoration, and a variety of other extended structures suggests that the possible diffusion mechanisms can be broadly defined into two categories: The first category involves the movement of H atoms through the rupture and formation of O-H bonds. The second category involves the movement of hydroxyls or even water molecules as stable units thereby requiring the breaking and formation of Si-O and Mg-O bonds.

Since the numbers of free H atoms (non-bonded to any O) and that free O atoms (not bonded to any cation) are zero (or nearly zero), all hydrogen must be attached to the Si-Mg matrix in the form of Si-O-H groups or Mg-O-H groups or in the mixed forms. During diffusion, an H atom is transferred from one group to another group. Several combinations of the source

(start) and destination (end) groups for the H-atom transfer are possible but only a few cases appear to dominate the diffusion process.

We first consider the cases where both the source and destination groups involve polyhedral oxygen (PO), which can be non-bridging oxygen (NBO) or bridging oxygen (BO). In the simplest but perhaps the most common case, the H atom moves from one NBO to another NBO thereby breaking the bond with the first NBO and forming a new bond with the second NBO. Note that NBOs are highly abundant in hydrous silicate melt at all volumes. When only one H atom is involved, this reaction proceeds as follows:

$$\text{PO-H}\ldots\text{.PO} \rightarrow \text{PO-H-PO} \rightarrow \text{PO}\ldots\text{.H-PO}$$

with PO as NBO. The reactants are the source polyhedral hydroxyl (NBO-H) and destination polyhedral oxygen (NBO), in other words, the source O has one H atom to lose to the H-free destination O (Figure 8-13, Left). In the intermediate state the H-atom forms polyhedral bridging (Figure 8-13, Center). The products are the source polyhedral oxygen (NBO) and the destination polyhedral hydroxyl (NBO-H), i.e., the source O becomes H free whereas the destination O has now one H atom (Figure 8-13, Right). In effect, an H atom is transferred from one polyhedral O to another polyhedral O via a momentary formation of H bridging. The high abundances of hydroxyls and bridging support this mechanism. The newly formed polyhedral hydroxyl may now act as a source for the next transfer of the same H atom. While polyhedral bridging is strongly favored, if both the source and destination O atoms belong to the same polyhedron, the H atom simply decorates an edge in the intermediate state. Other polyhedral mechanisms (perhaps less favorable ones) involve the motion of H from NBO to BO or from BO to NBO or from BO to BO.

In the general form of the above reactions, the source may contain one or more H atoms (i.e., it may be a polyhedral hydroxyl or a polyhedral water or a part of some extended structure) and the destination may contain zero or more H atoms (i.e., may be a polyhedral oxygen or a polyhedral hydroxyl or even a polyhedral water or a part of some extended structure). We do not consider a hydronium group as a source or a destination and rather treat it as an intermediate state because of its relative short lifetime. Depending on the reactants, the intermediate states and final products can vary. Some relevant cases are:

$$\text{PO-H}\ldots\text{.PO-H} \rightarrow \text{PO-H-PO-H} \rightarrow \text{PO}\ldots\text{.H-PO-H}$$

H-PO-H….PO → H-PO-H-PO → H-PO…..H-PO

H-PO-H….PO-H → H-PO-H-PO-H → H-PO…..H-PO-H

PO-H….H-PO-H → PO-H=PO-H → PO ….. H=PO-H

H-PO-H….H-PO-H → H-PO-H=PO-H → H-PO ….. H=PO-H

The last two reactions require O atom be coordinated with three H atoms thereby forming a hydronium group, which is extremely rare at low compression.

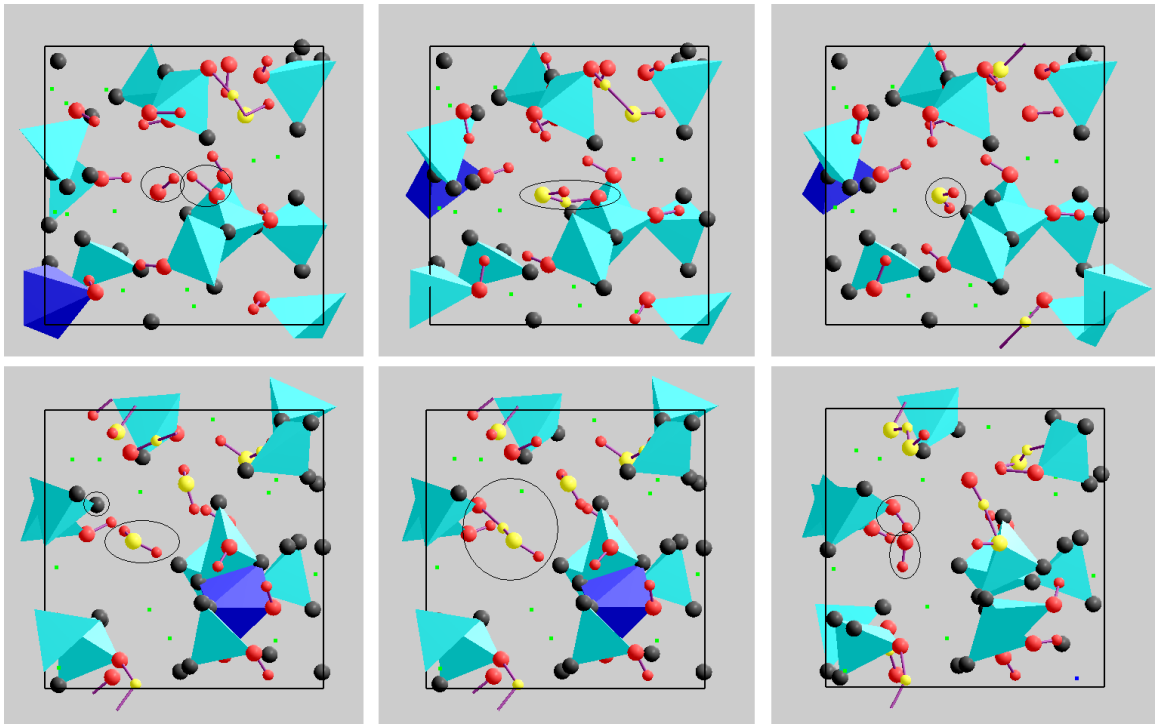

Figure 8-14: Three stages of the H transfers from NPO to PO (top) and from PO to NPO (bottom) marked by circles. The intermediate stage in both the cases involves a four-atom species.

An H atom bound to PO can be released to a non-polyhedral oxygen (NPO), which is an O atom bonded to only Mg atom. Since NPO is always involved in H-bonding, it must be already bonded to, at least, one H atom even though it is a receiver. In the simplest (but the most common) case, the reactants are the source polyhedral hydroxyl (preferably, NBO-H) and the destination non-polyhedral hydroxyl (Figure 8-14, Top-left).

PO-H….NPO-H → PO-H-NPO-H → PO…..H-NPO-H

The intermediate stage must involve water-like group (about Mg atom), with one H shared between PO and NPO, and other H attached only to NPO (Figure 8-14, Top-center). This

four-atom structure is quite abundant at low compression. The PO-H bond eventually breaks thereby forming an Mg-bound water molecule (H-NPO-H) as shown in Figure 8-14 (Top-right). This reaction may be subsequently followed by another reaction in which the newly formed water molecule loses one of its H atoms to a different PO (Figure 8-14, Bottom).

H-NPO-H….PO → H-NPO-H-PO → H-NPO…..H-PO

Note that the source NPO must be a water group for it to be able to lose one H atom (Figure 8-14, Bottom-left). The intermediate stage is the same as in the first reaction, and the products are the source non-polyhedral hydroxyl and destination polyhedral hydroxyl (Figure 8-14, Bottom-right). The former may now serve as the destination group with a new source PO-H and the latter may now serve as the source group with a new destination NPO-H for possible two occurrences of the PO-to-NPO transfer. If the source in the PO-to-NPO reaction contains water, then

H-PO-H….NPO-H → H-PO-H-NPO-H → H-PO…..H-NPO-H

One H is transferred from the source water to the destination hydroxyl. Similarly, if the destination contains water, the H transfer reaction proceeds as

PO-H….NPO=H → PO-H-NPO=H → PO…..H-NPO=H

The product hydronium eventually disintegrates into water and hydroxyl or two water groups in association with other oxygen or hydroxyl.

Finally, in the case of NPO to NPO transfer, the source O atom must have already, at least, two H atoms attached (i.e., water like group) whereas the destination O must have, at least, one H atom so a minimum of three H atoms are involved. More importantly, two H-bearing non-polyhedral groups should be close to each other in order to be able to transfer H. The number of NPO is relatively small (about 10 %) and decreases with compression so it's less likely to find two NPOs sufficiently close to each other. This condition makes the NPO-NPO transfer less favorable than the other two mechanisms. We consider only the simplest and perhaps most likely version:

H-NPO-H….NPO-H → H-NPO-H-NPO-H → H-NPO…..H-NPO-H

The five-atom structure involved in the intermediate stage is relatively rare at all compression.

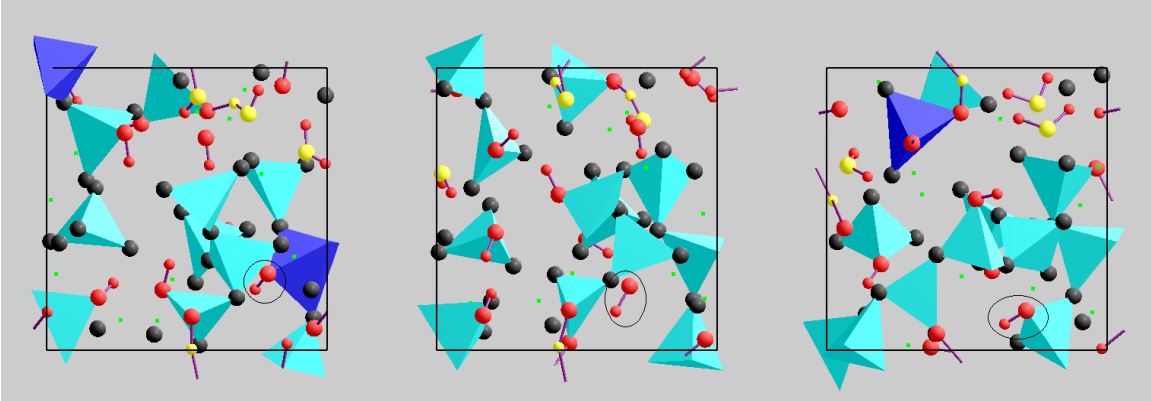

Figure 8-15: Transfer of a hydroxyl group between polyhedral and non-polyhedral units.

We now explore the mechanisms in which stable hydroxyls and water molecules serve as hydrogen carriers. Our structural analysis suggests that non-polyhedral oxygen atoms are always bonded to one or more H atoms, and some O-H bonds are stable for extended durations. This means that the motion of such stable units can proceed through the rupture and formation of Mg-O and/or Si-O bonds without breaking H-O bond. Diffusion in the form of hydroxyl can occur through its transfer between like (Mg-Mg or Si-Si) or unlike (Mg-Si or Si-Mg) cationic pairs:

$$\text{X-OH} \ldots\ldots \text{Y} \rightarrow \text{X-OH-Y} \rightarrow \text{X} \ldots\ldots \text{HO-Y}$$

where X, Y = Si or Mg. In these reactions, we assume that the H-O bond remains stable during its transfer from one cation (X) to another (Y) (Figure 8-15). The transfer mechanism is more common during time periods when the O atom is not a part of any polyhedron for two reasons. First, a nonpolyhedral hydroxyl does not disintegrate until it is turned into a nonpolyhedral water molecule or water-like group or a polyhedral hydroxyl. Second, Mg-O bonds have much shorter lifetimes than Si-O bonds. The above reactions also hold for the case of water molecule considered as a moving species but water molecules (free) are much less abundant than hydroxyls and are almost absent at high compression. Also unlike non-polyhedral hydroxyl, non-polyhedral water can lose one of its H atoms at any time.

The direct H transfer mechanism involving PO and NPO can be considered occurring together with hydroxyl/water transfer mechanism. The mixed mechanisms allow the possibility of transition between hydroxyl and water molecule as well as between polyhedral and non-polyhedral association during the diffusion process. To illustrate the nature of hydrogen diffusion, we follow the trajectory of a randomly selected hydrogen atom (labeled as H*) over a

period of about 10 ps starting at the 22190th step. At the start, this hydrogen atom forms polyhedral hydroxyl (Si-O*-H* or simply PO*-H*). After 90 steps, the polyhedral hydroxyl forms 4-atom structure with another polyhedral hydroxyl and steals its hydrogen thereby forming polyhedral water at the 22340th step:

PO-H …… PO*-H* → PO-H-PO*-H* → PO …. H-PO*-H*

Within the next 10 steps, the Si-O bond breaks and PO thus becomes NPO to form nonpolyhedral water (H-NPO*-H*). The original O*-H* bond remains unbroken up to the 25530th step existing mostly as a non-polyhedral hydroxyl although four-atom structures with other hydrogen attached to different PO or NPO or molecular water appear occasionally. Long structure involving as many as six atoms (NPO-H-NPO*-H-NPO-H) is formed at the 25000th step, which soon (within 160 steps) reduces to NPO*-H*. Later, this hydroxyl turns into molecular water which forms bond with another nonpolyhedral hydroxyl before the original bond (NPO*-H*) breaks (between 25380 and 25530 steps):

H-NPO*-H* ……. NPO-H → H-NPO*-H*-NPO-H → H-NPO* .... H*-NPO-H

In effect, the original hydrogen is now transferred from one NPO to another NPO. The non-polyhedral water soon reduces to a non-polyhedral hydroxyl, which lasts for another 1650 steps. At the 25560th step, it turns into a polyhedral hydroxyl (H*-PO) by forming Si-O bond. This is the second time H* is bonded with PO (of course, a different oxygen atom than original O). The polyhedral hydroxyl is stable for another 400 steps before it loses H* to another PO via edge-decoration. The newly formed polyhedral hydroxyl remains stable until the 28750th step and then it loses H* to non-polyhderal hydroxyl thereby forming molecular water:

PO-H* …… NPO-H → PO-H*-NPO-H → PO ……. H*-NPO-H

The water molecule loses/exchanges the other hydrogen a few times before H* is transferred to PO at the 31020th step:

PO …… H*-NPO-H → PO-H*-NPO-H → PO-H* …… NPO-H

During the next 3700 steps, H* moves from PO to PO three times (twice via polyhedral bridging and once via edge decoration) before it turns into a nonpolyhedral hydroxyl (at the 34720th step). This hydroxyl turns into a polyhedral hydroxyl at the 35640th step and H* maintains its polyhedral association over next 8000 steps involving 11 PO-to-PO transfers. Then H* is transferred to a NPO and so on.

8.4 Silica

We visually analyze four different silica systems at various temperatures and pressures using the ring structure statistics. The systems contain 72 atoms (24 Silicon atoms and 48 Oxygen atoms). Let's say the four systems are V1T1, V2T2, V3T3 and V4T4. V1T1 represents a system at 3000K temperature and volume 1099.1 A^3, V2T2 represents a system at 4000K temperature and volume 882.03 A^3, V3T3 represents a system at 5000K temperature and volume 659.46 A^3 and V4T4 represents a system at 6000K temperature and volume 439.635 A^3. The volume at 3000K temperature is the zero pressure volume and with the temperature the pressure is also increased.

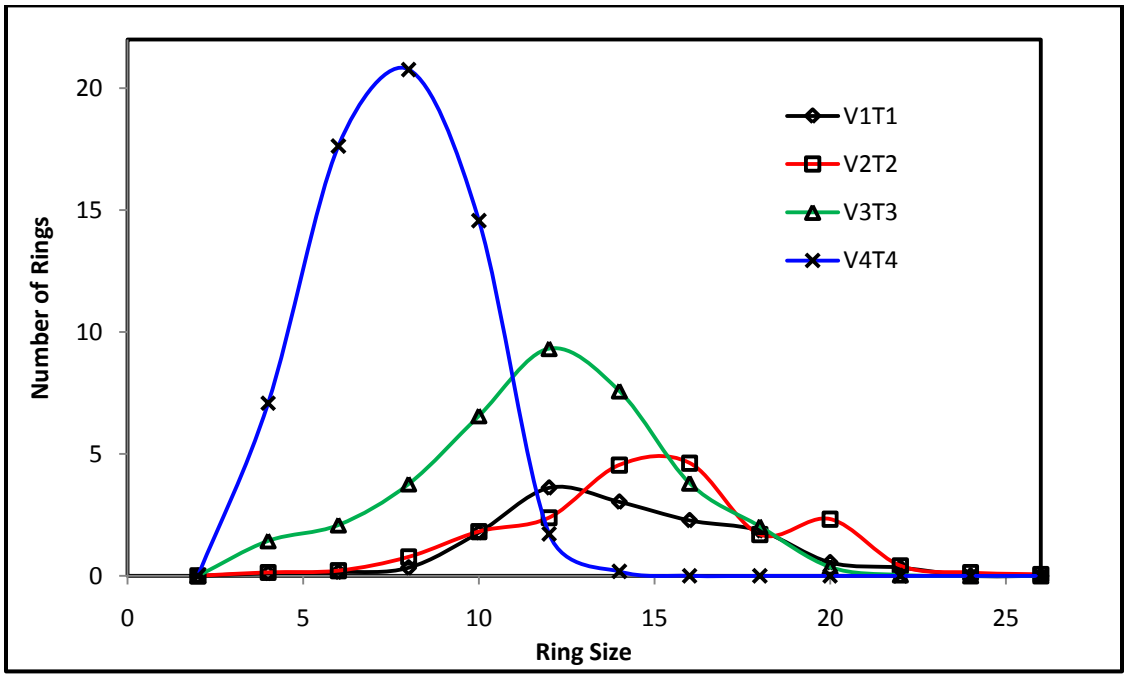

Figure 8-16: Distribution of primitive rings in silica at four different temperature and pressure conditions

The chart (Figure 8-16) shows distribution of primitive rings per step per atom for silicon atoms. The ring statistic distribution was calculated over 2500 time steps and for every silicon atom at every step and averaged out. For the ring calculation, in the graph construction the Silicons had edges only with the Oxygens and the Oxygens had edges only with the Silicons within the r_{min}^{SiO} of the Si-O RDF. The r_{min}^{SiO} for 3000K, 4000K, 5000K and 6000K were 2.275 A, 2.275 A and 2.375 A respectively. There were no edges between similar species and the constructed graph was a bipartite graph. All the cycles in a bipartite graph are even cycles.

We found that with the increasing pressure and temperature, the total number of ring increases. At 6000K, the rings with 12 atoms are largest in proportions, whereas at 4000K, there are rings with 14 and 16 atoms almost in equal amount, 4.553 and 4.632 rings respectively. The number of rings per time step per atom at 3000K is 14.1, at 4000K, it is 19.2, at 5000K, it is 37.0, and at 6000K, it is 62.0. The compressed systems, V3T3 and V4T4 contain considerably less number of large sized rings compared to less compressed system, V1T1 and V2T2.

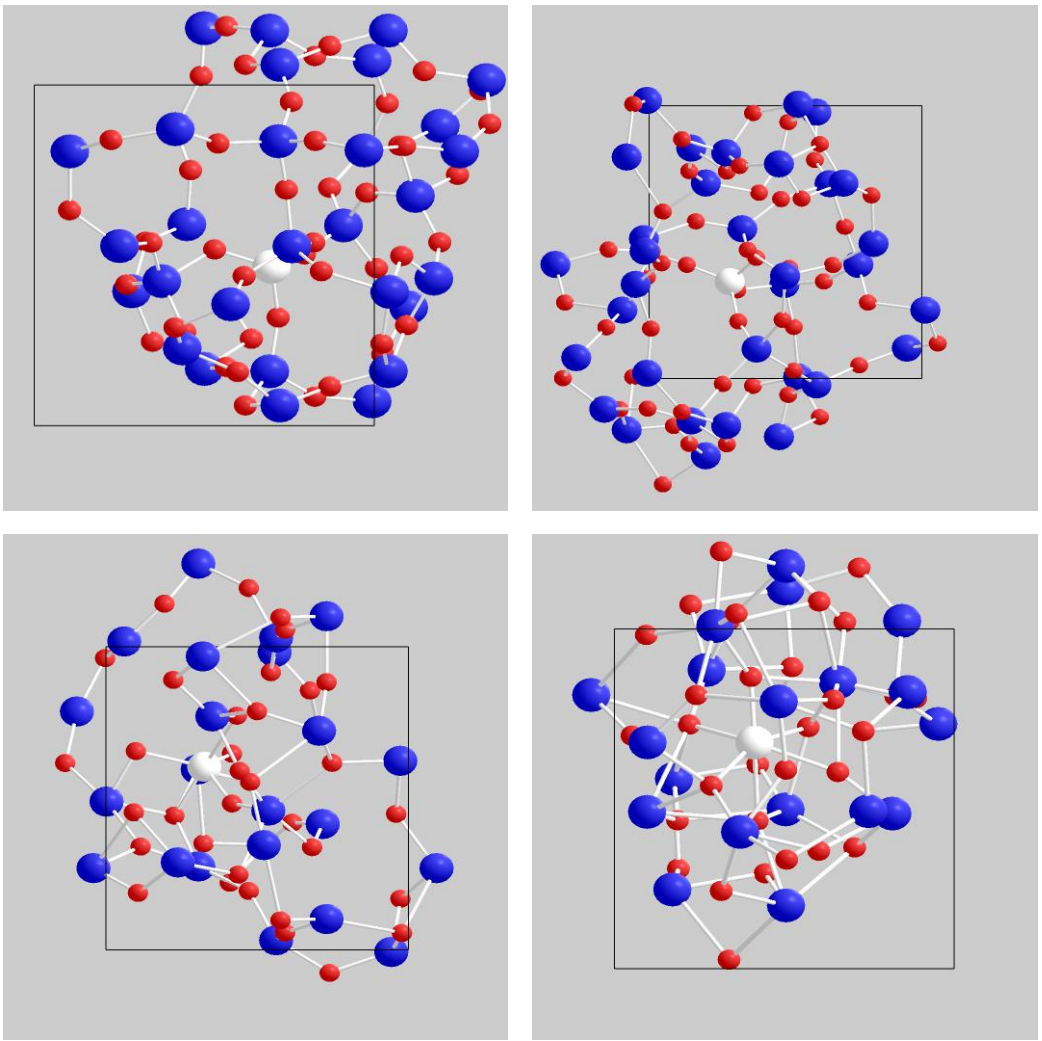

Figure 8-17: Silicon-Oxygen ring at four different temperatures and pressures, (left top) V1T1, (right top) V2T2, (bottom left) V3T3, and (bottom right) V4T4

Figure 8-17 shows the instantaneous ring structures for all the Silica systems. The white sphere in the figures is the atom for which the ring structure is calculated. The V1T1 snapshot contains 16 rings (4 10-ring, 2 12-ring, 2 14-ring, 3 16-ring, 4 18-rings and 1 20-ring), the V2T2 snapshot contains 23 rings (1 8-ring, 1 10-ring, 2 12-ring, 4 14-ring, 8 16-ring, 6 20-ring and 1

22-ring), the V3T3 snapshot contains 26 rings (2 4-ring, 1 6-ring, 11 10-ring, 5 14-ring, 6 16-ring and 1 18-ring), and the V4T4 snapshot contains 47 rings (7 4-ring, 14 6-ring, 16 8-ring and 10 10-ring). We notice that with the increasing pressure the number of small rings become more dominant and the number of large rings becomes insignificant.

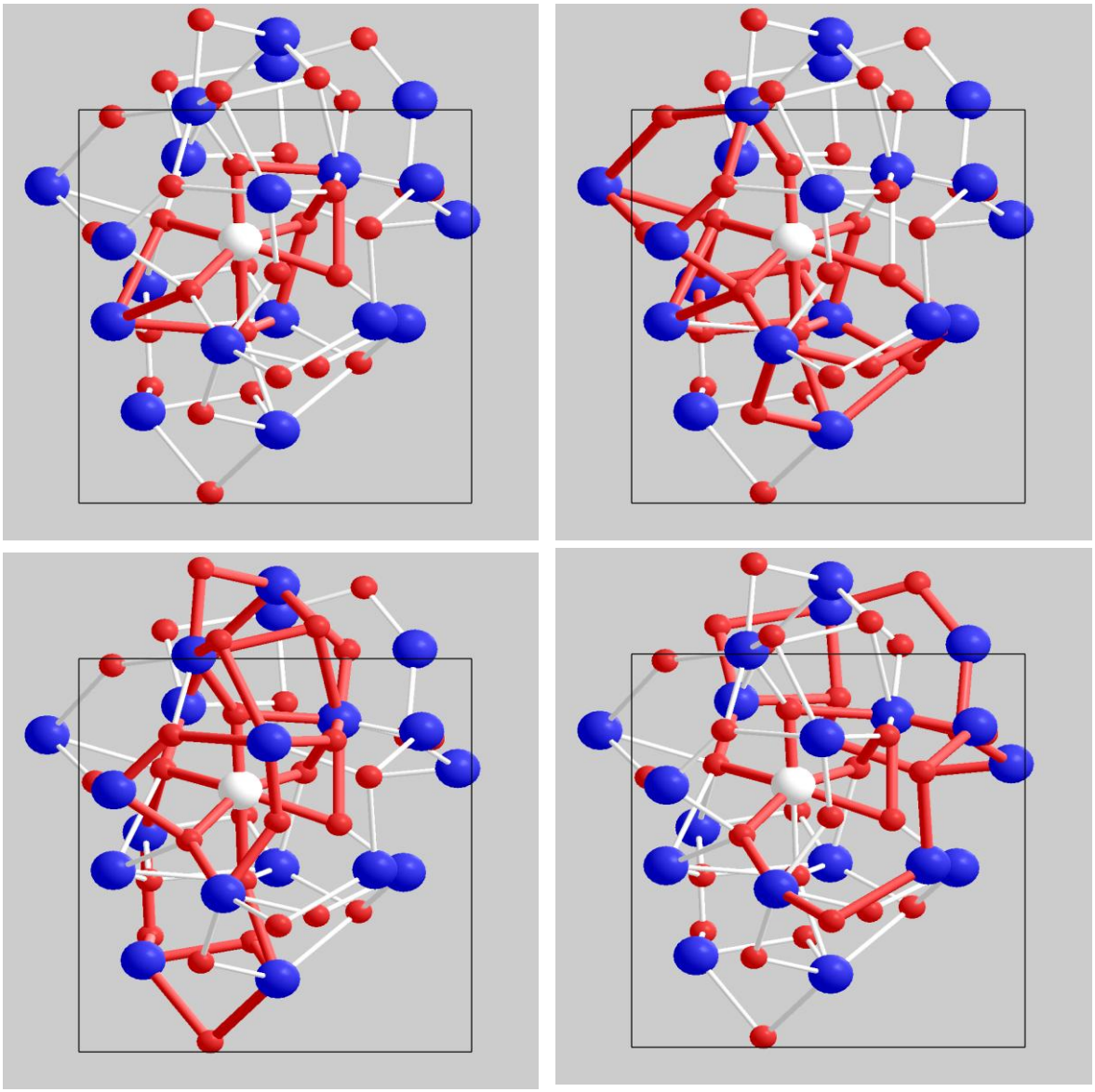

Figure 8-18: The rings of sizes 4, 6, 8 and 10 (clockwise from left top) respectively for the V4T4 dataset.

Chapter 9 Discussion, Conclusion and Future Works

In this work, we modeled the time-varying atomistic positional dataset as a realization of some unknown point process. Such modeling enabled us to abstract away the quantum interactions that went in determining the positions and focus on the spatial and temporal behaviors of the dataset. We assumed that the given dataset was stationary and isotropic. Such assumptions allowed us to compute and utilize the radial distribution function for detail structural analyses and also to explain the results with rigor. The correlations between pair of types of atoms vary according to the selected pair of types. The radial distribution functions allowed us to quantify and visualize such correlations. We were able to deduce important structural differences in the various mineral systems at different temperature and pressure conditions as explained in the application studies.

Also, we were able to utilize concepts from principal components analyses to visualize the anisotropy in the atomic movement by treating the set of positions an atom occupied during the simulation simply as a set of point. Once the time property of the positions was removed, it was an easy task to compute the covariance matrix for the positions set. The principal components of the covariance matrix for an atom, the eigenvalues and the eigenvectors, were the visualized using an ellipsoid. The ellipsoid was able to better impart the information about the correlation between movements along three canonical axes than the diffusion sphere method. The diffusion sphere visualization assumed the atomic movement to be isotropic which was not true in reality.

The ultimate objective of this work was to provide a complete software workbench to flexibly analyze and visualize the atomistic datasets. I provided a solid foundation upon which the future extensions can be made to achieve the ultimate goal. Many features expected of such a workbench was worked out and were modeled using techniques from theory of point process. The assumptions implicit in the other atomistic visualization works were also made explicit in this work. For example, the time-varying atomistic datasets were previously treated as deterministic datasets. Here, I showed that they can be modeled as one realization of the marked point processes. This realization opened up a whole new range of exploratory techniques from

the theory of point process that can be utilized. Only a few of such techniques such as the radial distribution functions and K-function have been utilized in this work. This work also showed that the formal integration of analytical algorithms with the visualization algorithms provided a new set of tools to the scientific community.

The process of information extraction is an iterative process and requires exploration of the dataset from multiple perspectives. There are some theoretical works to make a general visualization system useful for data exploration [92, 101]. This work provided multiple exploratory techniques to analyze and to visualize the atomistic datasets. Large scale atomistic simulations are becoming possible. Such simulations generate huge amounts of datasets and exploration and understanding of the generated datasets pose a tremendous challenge. The integration of the complete set of exploratory techniques into a flexible interface is a separate problem and remains as a future work to be done. The approaches presented here can also be generalized for more general time-varying spatial datasets. The ideas in the field of point process combined with the techniques from the current work will provide a very powerful paradigm for future atomistic visualization systems. The exploratory techniques for larger systems may also have to be refined and be modeled differently than what is proposed in this work.

A more formal approach can result better estimation of the statistical parameters with less computation thus resulting better performing analytical algorithms. The large systems will also present difficulties in rendering. With more graphics rendering power increasingly becoming easily available, graphical rendering part of the visualization system will still have more leeway in terms of performance than the analytical algorithms. The performance analyses show that the computational techniques are time consuming and hinder an interactive analysis. With multi-core processors becoming easily accessible, such computations can be parallelized to execute interactively. The interface developed in this work can be taken as a first step in that direction.

The user interface can provide features to merge and integrate multiple exploratory techniques together so that multiple features of the dataset can be explored simultaneously. The visualization acts as an interface to the dataset to enable the understanding and without understanding how human perceive the graphical outputs generated by the visualization system, it will be impossible to find the optimal way of interfacing.

References

1. Cressie, N.A.C., *Statistics For Spatial Data*. Wiley Series in Probability And Mathematical Statistics. 1993: John Wiley & Sons, Inc.

2. Spence, R., *Information Visualization*. 2001: ACM Press.

3. Bertin, J., *Semiology of Graphics*. 1st ed. 1983: The University of Wisconsin Press.

4. Zipf, G.K., *The Psycho-Biology of Language*. 1935, Boston: Houghton Mifflin Company.

5. Wainer, H., *Graphical Visions from William Playfair to John Tukey.* Stastical Science, 1990. **5**(3): p. 340-346.

6. Yeary, R., *Making the Unseen Visible*, in *IEEE Computer Graphics and Applications*. 1997. p. 2.

7. Swift, R.M. and D. White, *Low Temperature Heat Capacities of Magnesium Diboride (MgB_2) and Magnesium Tetraboride (MgB_4).* Journal of American Chemical Society, 1957(14): p. 3641-3644.

8. Pickett, W.E. *Probing Room Temperature Superconductivity*. 2005 [cited; Available from: http://yclept.ucdavis.edu/Projects/NotreDameBanquet.pdf.

9. Tufte, E.R., *Envisionning Information*. 1990, Cheshire, Connecticut: Graphics Press.

10. Tufte, E.R., *Visual Explanationa : Images and Quantities, Evidence and Narrative*. 1997, Cheshire, Connecticut: Graphics Press.

11. Tufte, E.R., *The Visual Display of Quantitative Information*. 2001, Cheshire, Connecticut: Graphics Press.

12. Tufte, E.R., *Beautiful Evidence*. 2006: Graphics Press.

13. Tukey, J.W., *Exploratory Data Analysis*. Behavioral Science: Quantitative Methods. 1977: Addison-Wesley Publishing Company.

14. Chernoff, H., *The Use of Faces to Represent Points in K-Dimensional Space Graphically.* Journal of the American Statistical Association, 1973. **68**(342): p. 361-368.

15. Kandogan, E. *Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions.* in *Proceedings of IEEE Information Visualization, Hot Topics*. 2000.

16. Inselberg, A., *The plane with parallel coordinates.* The Visual Computer, 1985. **1**(4): p. 69-91.

17. McCormack, B.H., *Scientific and Engineering Research Opportunities.* SIGGRAPH-Computer Graphics, 1987. **21**(6): p. 15-21.

18. Connolly, M.L., *Analytical Molecular Surface Calculation.* Journal of Applied Crystallography, 1983. **16**: p. 548-558.

19. Max, N.L., *Computer representation of molecular surfaces.* Journal of Molecular Graphics, 1984. **2**(1).

20. Moon, J.B. and W.J. Howe, *A fast algorithm for generating smooth molecular dot surface representations.* Journal of Molecular Graphics, 1989. **7**: p. 109-112.

21. Pearl, L.H. and A. Honegger, *Generation of molecular surfaces for graphics display.* Journal of Molecular Graphics, 1983. **1**(1): p. 9-12.

22. Carson, M. and C.E. Bugg, *Algorithm for ribbon models of proteins.* Journal of Molecular Graphics, 1986. **4**(2): p. 121-122.

23. Carson, M., *Ribbon models of macromolecules.* Journal of Molecular Graphics, 1987. **5**(2): p. 103-106.

24. Staudhammer, J., *On Display of Space Filling Atomic Models in Real-time.* SIGGRAPH, 1978.

25. Max, N.L., *ATOMLL: - ATOMS with Shading and Highlights.* SIGGRAPH, 1979.

26. Porter, T.K., *The Shaded Surface Display of Large Molecules.* SIGGRAPH, 1979.

27. Humphrey, W., A. Dalke, and K. Schulten, *VMD - Visual Molecular Dynamics.* Journal of Molecular Graphics, 1996. **14**: p. 33-38.

28. Kokalj, A., *XCrySDen-a new program for displaying crystalline structures and electron densities.* Journal of Molecular Graphics and Modelling, 1999. **17**(3-4): p. 176-179.

29. Li, J., *AtomEye: an efficient atomistic configuration viewer.* Modelling and Simulation in Materials Science and Engineering, 2003. **11**: p. 173-177.

30. Kraulis, P.J., *MOLSCRIPT: A Program to Produce Both Detailed and Schematic Plots of Protein Structures.* Journal of Applied Crystallography, 1991. **24**: p. 946-950.

31. ProteinExplorer, *Protein Explorer.*

32. Sayle, R.A. *http://www.umass.edu/microbio/rasmol/perhistory.txt*. 1995 [cited.

33. Pettersen, E.F., et al., *UCSF Chimera - A Visualization System for Exploratory Research and Analysis.* Journal of Computational Chemistry, 2004. **25**(13): p. 1605-1612.

34. Momma, K. and F. Izumi, *VESTA: a three-dimensional visualization system for electronic and structural analysis.* Journal of Applied Crystallography, 2008. **41**: p. 653-658.

35. CrystalMaker. *http://www.crystalmaker.com*. [cited.

36. amiraMol. *http://www.amiravis.com/mol*. [cited.

37. Sharma, A., et al., *Immersive and interactive exploration of billion-atom systems.* Presence, 2003. **12**: p. 85-95.

38. Sharma, A., et al., *Scalable and portable visualization of large atomistic datasets.* Computer Physics Communications, 2004. **163**: p. 53-64.

39. Kadau, K., T.C. Germann, and P.S. Lomdahl, *Large-Scale Molecular-Dynamics Simulation of 19 Billion Particles.* International Journal of Modern Physics C, 2004. **15**(1): p. 193-201.

40. MolScript. *http://www.avatar.se/molscript/*. [cited.

41. Ferrin, T.E., et al., *The MIDAS display system.* Journal of Molecular Graphics, 1988. **6**.

42. Voorhies, D., D. Kirk, and O. Lathrop, *Virtual Graphics.* SIGGRAPH, 1988. **22**(4): p. 247-253.

43. OpenGL-ARB. *http://www.opengl.org*. 2007 [cited.

44. DirectX. *DirectX*. 2008 [cited; Available from: http://msdn.microsoft.com/en-us/directx/default.aspx.

45. Foley, J.D., et al., *Computer Graphics: Principles and Practices*. Second ed. The Systems Programming Series. 1997: Addison Wesley.

46. Owens, J.D., *Computer Graphics on a Stream Architecture*, in *Department of Electrical Engineering*. 2002, Stanford University. p. 178.

47. Segal, M. and M. Peercy. *A Performance-Oriented Data Parallel Virtual Machine for GPUs*. in *SIGGRAPH*. 2006: ACM SIGGRAPH.

48. Whitted, T. and J. Kajiya. *Fully Procedural Graphics*. in *Graphics Hardware 2005*. 2005. Los Angeles, CA: ACM.

49. Hall, R., *Illumination and Color in Computer Generated Imagery*. Monographs in Visual Communication, ed. D.F. Rogers. 1989, New York: Springer-Verlag.

50. Goedecker, S. and K. Maschke, *Transferability of pseudopotentials.* Physical Review A, 1992. **45**(1): p. 88-93.

51. Kizuka, T., *Atomistic visualization of deformation in gold.* Physical Review B, 1998. **57**(18): p. 158-163.

52. Merkel, S., et al., *Deformation of (Mg, Fe)SiO_3 Post-Perovskite and D'' Anisotropy.* Science, 2007. **316**.

53. Oganov, A.R. and S. Ono, *Theoretical and experimental evidence for a post-perovskite phase of $MgSiO_3$ in Earth's D'' layer.* Nature, 2004. **430**: p. 445-448.

54. Murakami, M., et al., *Post-Perovskite Phase Transition in $MgSiO_3$.* Science, 2004. **304**: p. 855-858.

55. Nakano, A., et al., *A divide-and-conquer/cellular-decomposition framework for million-to-billion atom simulations of chemical reactions.* Computational Materials Science, 2007. **38**: p. 11.

56. Wong, P.C., *Visual Data Mining*, in *IEEE Computer Graphics and Applications*. 1999. p. 2.

57. Lorensen, B. *On the Death of Visualization*. in *Fall 2004 Workshop on Visualization Research Challenges*. 2004.

58. Wijk, J.J.v., *Views on Visualization.* IEEE Transactions on Visualization and Computer Graphics, 2006. **12**(4): p. 12.

59. Patrascioiu, A., *The Ergodic-Hypothesis: A complicated Problem in Mathematics and Physics.* Los Alamos Science (Special Issue), 1987: p. 263-279.

60. Walker, H.M., *Degrees of Freedom.* Journal of Educational Psychology, 1940. **31**(4): p. 253-269.

61. Diggle, P.J., *Spatio-Temporal Point Processes: Methods and Applications*, in *Statistical Methods For Spatio-Temporal Systems*, B. Finkenstadt, L. Held, and V. Isham, Editors. 2007, Champam & Hall/CRC. p. 1-46.

62. Diggle, P.J., *Statistical Analysis of Spatial Point Patterns*. Mathematics in Biology, ed. R. Sibson and J.E. Cohen. 1983: Academic Press.

63. Ripley, B.D., *Spatial Statistics*. Wiley series in probability and mathematical statistics: applied probability & math section). 1981: John Wiley & Sons, Inc.

64. Lotwick, H.W. and B.W. Silverman, *Methods for Analysing Spatial Processes of Several Types of Points.* Journal of the Royal Statistical Society. Series B (Methodological), 1982. **44**(3): p. 406-413.

65. Smith, F.W., *Euclidean Geometry and the Flow of Generalized Liquids.* Discussions of the Faraday Society, 1967. **43**: p. 231-234.

66. Rapaport, D.C., *The Art of Molecular Dynamics Simulation*. 1995: Cambridge University Press.

67. Bernal, J.D., *The structures of liquids.* Proceedings of Royal Society of London, 1962. **280**: p. 299-322.

68. Yuan, X. and A.N. Cormack, *Efficient algorithm for primitive ring statistics in topological networks.* Computational Materials Science, 2002. **24**: p. 18.

69. Frazblau, D.S., *Computation of ring statistics for network models of solids.* Physical Review B, 1991. **44**(10): p. 4925.

70. Balducci, R. and R.S. Pearlman, *Efficient Exact Solution of the Ring Perception Problem.* Journal of Chemical Information and Computer Sciences, 1994. **34**(4): p. 822-831.

71. Stixrude, L. and M.S.T. Bukowinski, *Rings, topology, and the density of tectosilicates.* American Mineralogist, 1990. **75**: p. 1159-1169.

72. Rino, J.P., et al., *Structure of rings in vitreous SiO_2.* Physical Review B, 1993. **47**(6): p. 3053-3062.

73. Jain, A., V. Kumar, and Y. Kawazoe, *Ring structures of small ZnO clusters.* Computational Materials Science, 2006. **36**: p. 5.

74. Jin, Y.-j., B.-h. Yang, and Y.-h. Huang, *Molecular orbital studies on the rings composed of D_{2d} C_{36} cages.* Computational Materials Science, 2006. **36**: p. 6.

75. Sahar, M.R., A.W.M.A. Hussein, and R. Hussin, *Structural characteristic of Na_2O-P_2O_5-GeO_2 glass systems.* Journal of Non-Crystalline Solids, 2007.

76. Sastre, G. and J.D. Gale, *A software tool for the topological and geometrical characterization of three-dimensional frameworks.* Computational Materials Science, 2003. **28**: p. 8.

77. Rybicki, J., G. Bergmanski, and G. Mancini, *A new program package for investigation of medium-range order in computer-simulated solids.* Journal of Non-Crystalline Solids, 2001. **293-295**: p. 758-763.

78. Zachariasen, W.H., *The atomic arrangement in glass.* Journal of American Chemical Society, 1932. **54**(10): p. 11.

79. Alon, N., R. Yuster, and U. Zwick, *Finding and Counting Given Length Cycles.* Algorithmica, 1997. **17**: p. 209-223.

80. Allen, M.P. and D.J. Tildesley, *Computer Simulation of Liquids*. 1987: Oxford University Press.

81. Jolliffe, I.T., *Principal Component Analysis*. 1986: Springer-Verlag.

82. Brunk, H.D., *An Introduction to Mathematical Statistics*. 1965: Blaisdell Publishing Company.

83. Rahman, A., *Correlations in the Motion of Atoms in Liquid Argon.* Physical Review, 1964. **136**(2A): p. 405-411.

84. Barber, C.B., D.P. Dobkin, and H. Huhdanpaa, *The Quickhull Algorithm for Convex Hulls.* ACM Transactions on Mathematical Software, 1996. **22**(4): p. 15.

85. Grunbaum, B. *Measure of symmetry for convex sets.* in *Proceedings of the 7th Symposium in Pure Mathematics of the American Mathematical Society, Symposium on Convexity*. 1961. Providence, R.I.: AMS.

86. Hazen, R.M., *Temperature, Pressure and Composition: Structurally Analogous Variables.* Physics and Chemistry of Minerals, 1977. **1**: p. 83-94.

87. Robinson, K., G.V. Gibbs, and P.H. Ribbe, *Quadratic elongation - quantitative measure of distribution in coordination polyhera.* Science, 1971. **171**: p. 567-570.

88. Thomas, N.W., *Crystal structure-physical property relationships in perovskites.* Acta Crystallography, 1989. **B45**: p. 337-344.

89. Everitt, C., *Interactive Order-Independent Transparency*. 2001.

90. Moller, T. and J.F. Hughes, *Efficiently Building a Matrix to Rotate One Vector to Another*. Journal of Graphics Tools, 1999. **4**(4): p. 1-4.

91. MATLAB. *http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/f2-9676.html*. [cited.

92. Jankun-Kelly, T.J., *Visualizing visualization: A model and framework for visualization exploration*, in *Computer Science*. 2003, University of California at Davis: Davis. p. 134.

93. Jackson, R., L. MacDonald, and K. Freeman, *Computer Generated Color: A Practical Guide to Presentation and Display*. 1994: John Wiley & Sons.

94. Shneiderman, B. *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. in *IEEE Visual Languages*. 1996. Los Alamos, California.

95. Chi, E.H.-h., et al., *Principles for Information Visualization Spreadsheets.* IEEE Computer Graphics and Applications, 1998. **18**(4): p. 30-38.

96. Blinn, J.F., *Simulation of wrinkled surfaces.* ACM SIGGRAPH Computer Graphics, 1978. **12**(3): p. 286-292.

97. Bhattarai, D., B.B. Karki, and L. Stixrude, *Space-time multiresolution atomistic visualization of MgO and $MgSiO_3$ liquid data.* Visual Geoscience, 2006. **11**(1): p. 11.

98. Horiuchi, H., E. Ito, and D. Weidner, *Perovskite type $MgSiO_3$ single crystal X-ray diffraction study.* American Mineralogist, 1987. **72**: p. 357-360.

99. Closmann, C. and Q. Williams, *In-situ spectroscopic investigation of high-pressure hydrated (Mg, Fe)SiO_3 glasses - OH vibrations as a probe of glass structure.* American Mineralogist, 1995. **80**: p. 201-212.

100. Zhang, Y.X., *H_2O in rhyolitic glasses and melts: Measurement, speciation, solubility and diffusion.* Reviews of Geophysics, 1999. **37**: p. 493-516.

101. Jankun-Kelly, T.J., K.-L. Ma, and M. Gertz, *A model and framework for visualization exploration.* IEEE Transactions on Visualization and Computer Graphics, 2007. **13**(2): p. 13.

Vita

Dipesh Bhattarai was born in Chitwan, Nepal, in September, 1975, to Tara Prasad Bhattarai and Madhu Maya Bhattarai. He graduated from Institute of Engineering, Tribhuvan University, Nepal, with Bachelor in Electronics Engineering. He worked as a software developer for some time before joining Alabama A&M University at Normal, Alabama, for master's degree in computer science on August, 2001. After graduating with a master's degree from Alabama A&M University on December, 2003, he came to Louisiana State University on August, 2004, to pursue the doctoral degree in computer science. He worked as a research assistant to Dr. Bijaya B. Karki while working toward the doctoral degree. His doctoral work involved scientific visualization, computer graphics and molecular dynamics simulation. He likes to extend his doctoral work in the future so that large datasets that are routinely generated during first principles molecular dynamics simulations can be visualized more efficiently.

His scientific publications are as follows:

1. Karki BB, **Bhattarai D** and Stixrude L, Visualization and analysis of structural and dynamical properties of hydrous silicate melt. *In preparation.*
2. Karki BB, **Bhattarai D** and Stixrude L, First principles simulations of liquid silica: structural and dynamical behavior at high pressure. *Physical Review B*, 2007, 76: 104205.
3. **Bhattarai,D** and Karki, BB, *Atomistic visualization: on-the-fly data extraction and rendering.* ACMSE 45: Proceedings of the 45th annual southeast regional conference, 2007. ISBN 978-1-59593-629-5. pp437-442.
4. **Bhattarai D**, Karki BB and Stixrude L. *Space-time multiresolution atomistic visualization of MgO and MgSiO3 liquid data.* Visual Geosciences, 2006.
5. Karki BB, **Bhattarai D** and Stixrude L. *A first-principles computational framework for liquid mineral systems.* CMC: Computers, Materials and Continua, 2006, 3: 107-118.
6. Karki BB, **Bhattarai D** and Stixrude, L. *First principles calculations of the structural, dynamical and electronic properties of liquid MgO.* Phys. Rev. B., 2006, 73: 174208.
7. **Bhattarai D** and Karki BB, *Visualization of atomistic simulation data for spatio-temporal information.* The 14th Int'l. Conf. on Central Europe in Computer Graphics, Visualization and Computer Vision, ISBN 80-86943-03-8BB, 2006, pp. 17-24.