Time Critical Computing and Rendering of Molecular Surfaces Using a Zonal Map

Henk Huitema, Robert van Liere
Center for Mathematics and Computer Science CWI,
Kruislaan 413, 1090 GB Amsterdam, the Netherlands.
e-mail: {huitema,robertl}@cwi.nl

Abstract

We describe the zonal map, a data structure used for the visualization of large, time dependent molecular configurations in virtual environments. The governing idea of the zonal map is to use the user's line of sight to define a region of interest onto which time critical algorithms can be applied. Two examples of time critical algorithms are given: for computing and rendering of solvent-accessible surfaces of protein molecules. We show that substantial performance gains can be obtained by using the zonal map.

1 Introduction

Virtual environments have shown great promise as a research tool for the exploration of molecular data. The additional depth information and intuitive interaction inherent to virtual environments can aid in obtaining insight to the 3-D structure and properties of a molecular configuration. Various implementations of virtual environments for molecular modeling and molecular dynamics are underway.

There are a vast number of visualization methods to represent molecules; e.g. stick, ball-and-stick, CPK space-filling methods, surfaces, etc. Each method highlights certain molecular properties. A problem with many of these methods is that of performance. Rendering large molecules – for example combining a CPK and a surface representation of a 5000 atom molecule – at a performance of minimal 10 frames a second is currently not possible. Recent developments have addressed the problem of performance by developing techniques that enhance the rendering speed at the cost of a less detailed representation of the molecule. Higher detail representations are used when a part of the molecule is perceptually more important and lower detail representations are used when the part of the molecule is perceptually less significant.

However, when studying molecular data in virtual environments, the goal is to visualize highly detailed molecule representations at close range. It is likely that the molecule being visualized will not move adequately far away from the viewer to allow the rendering algorithm to switch to a lower level of detail. The approach taken in this paper is not to compromise the level of detail of the representation. Instead, time critical algorithms are used to compute and render parts of the molecule representation at a high level of detail at predictable frame rates. This is realized by the *zonal map*: a data structure containing geometrical information of the molecule. A view dependent region of interest is defined in which the representation is rendered at a high level of detail. The size of the region of interest depends on the time critical algorithms that operate on the atoms in the region.

The motivation for a view dependent region of interest is that this is the area of the molecule where the user is focusing her attention on. Since the region of interest is close to the user, we render the molecule representation with a high level of detail. Molecule representations outside the region of interest need not be computed and rendered at all.

The zonal map has many applications: it can be used for the computation and rendering of many molecular representations, data culling, visibility, and adaptive molecular algorithms. This paper will focus only on using the zonal map for time critical computing and rendering of the solvent accessible surfaces of the molecule. The rest of the paper is organized as follows: First, we briefly review related work on molecular visualization and time critical rendering. In section 3 we discuss the zonal map and give two examples of time critical algorithms. In section 4 we show how the time critical algorithms are managed, in which we take advantage of pipelining in a multi-threaded environment. Finally, in section 5 we show some timing results for various molecules.

2 Related work

Numerous academic and commercial efforts have used virtual environments for the study of molecules. Many have placed emphasis on interactive protein modeling and molecular dynamics, [1, 2, 3], while others have concentrated on the interactive display of molecule representation, [4, 5]. Haase et. al., for example, developed techniques for rapid interaction, fast rendering of large molecules (including techniques as level of detail, fast sphere rendering, multi-pipe rendering, sorting for transparent surfaces), and on perceptual issues (such as shadows for additional size and position cues). Our approach for computing and rendering molecular surfaces differs from Haase's techniques in that we require that our techniques are applicable to large, time dependent, molecular configurations. Haase's techniques for surface rendering are less suitable when the molecular structure changes over time, since they rely on a number of pre-processing steps on the molecule.

Time critical computing for virtual environments has been an extensive topic of study. Funkhouser and Séquin presented an adaptive display algorithm for interactive frame rates during visualization of complex virtual environments, [6]. Their algorithm uses hierarchical model representations in which objects are described at multiple levels of detail. The predictive algorithm adjust image quality adaptively to maintain a uniform target frame rate. This is achieved by maximizing a benefit/cost ratio for rendering graphical objects in a scene. Transitions from one level of detail to the next are based on image-space metrics; e.g. the ratio of the image-space area of the representation to the distance of the representation from the viewer. Tests indicate that the algorithm performs at near uniform frame rates with large walk through models. The zonal map technique differs from the techniques based on hierarchical model representations in that we require the highest level of detail at all times. We use the notion of a view dependent region of interest in which the highest level of detail is used for the molecular surface. Outside the region of interest we may or may not display the surface. Our experience is that providing maximal detail in the region of interest outweighs the approach in which less detail is given. Obviously, this observation is application dependent and will not hold for walkthroughs.

Bryson and Johan report on an alternative method of time critical computing in interactive time-varying visualization environments, [7]. Assuming a scene consists of a number of individual visualizations, this method carefully assigns a time budget to each individual visualization. They report on different techniques that may used to allow the visualizations to meet their time budget. An example is that of a *local isosurface*, where there is a fixed cost per polygon so that the only way to control the time required to compute the isosurface is to control the number of polygons. One can start from a point in space and generate isosurface triangles until the time budget is used up. Although rendering the geometry in this way is similar to our approach, we compute the geometry differently. The zonal map provides structural information and properties of the object (in our case a molecule), which may be used by the visualization techniques to control which fragments of the object are to be visualized. In particular, we show that the number of triangles is not an optimal time critical parameter for the computation of the geometry. Instead, we use the number of atoms and their neighborhoods. In the case of molecular surfaces, the number of atoms is

directly related to a target frame rate, whereas the number of triangles is not (this is due to the abundance of cavities in the surface). We show that, by separating the computation from the rendering, we can use different time critical parameters to achieve the desired target frame rates.

The definition of a molecular surface – due to Richards [8] – is the surface which an exterior probe-sphere touches as it is rolled over the spherical atoms of the molecule. There are many algorithms for analytical computation of molecular surfaces, e.g. [9, 10]. These algorithms vary in their time complexity. In this paper we use an adapted version of the algorithm described by Varshney and Brooks. We use this algorithm because the source code is publicly available, but believe that the zonal map technique will also apply to the other computation algorithms of molecular surfaces.

3 The Zonal Map

3.1 Data structure

The zonal map is defined as a regular 3D grid of cells. Each atom is represented as a sphere with a center and a van der Waal's radius; $S_i = < c_i, r_i >$. Each cell contains a list of atoms for which the corresponding sphere intersects the cell. [1] The data structure can be written in pseudo-C code as:

```
struct zonal_map {
   int natoms;              /* number atoms in cell */
   Atom *atoms;             /* list of atoms */
}[NX][NY][NZ];
```

The resolution of the zonal map depends on various geometric properties of the molecule, such as its bounding box. Usually, each cell is configured to span approximately 0.2 nm.

We use the Manhattan distance compute the distance between two cells. The Manhattan distance is the distance between two cells measured along axes at right angles. For example, two cells at c_1 and c_2, have a Manhattan distance of $|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$. Using the Manhattan distance between two cells, instead of the Euclidean distance between two cells, simplifies the definition of a region in the zonal map.

Associated with the zonal map are the view dependent notions of a *focus cell* and a *region of interest*:

- focus cell

  A focus cell is defined as the cell which is at the user's center of attention. The line of sight obtained from the user's head position and orientation is used for this purpose. The focus cell is computed by first finding all spheres which intersect the line of sight and selecting the sphere with the smallest distance to the user. The focus cell is the cell in the zonal map which contains the center of the selected sphere.

  We denote the focus cell as C_f.

- region of interest

  A concentric ring of distance n is defined as the set of cells with a Manhattan distance of n to C_f. The notation:

  $$C(n, C_f) = \{C_i \mid Manhattan\_dist(C_i, C_f) = n\}$$

  is used to denote the concentric ring of distance n.

[1] Note that a sphere can reside in more than one cell.

A region of interest of size N is the set of cells whose Manhattan distance to C_f is less than or equal to N;

$$ROI(N, C_f) \;=\; \bigcup_{i=0}^{N} C(i, C_f)$$

As an example, consider the 2D zonal map with a resolution of 7x7 (see figure 1). A region of interest of size 3 is shaded. Twelve atoms and their van der Waal's radius are shown; three atoms are outside the region of interest. 1 atom resides in $C(0)$. 3 atoms reside in $C(1)$. 2 atoms reside in $C(2)$. 3 atoms reside in $C(3)$.

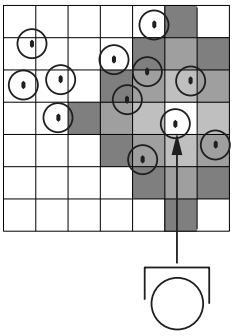

Figure 1: A 2D zonal map. The user's line of sight determines the focus-cell. The Manhattan distance to the focus-cell is used to determine concentric rings around the focus-cell (shaded cells).

The goal of a time critical algorithm operating using the zonal map can be stated as:
Maximize :

$$ROI(n, C_f)\;, \qquad n = 0, 1, ..., N \tag{1}$$

Subject to:

$$T_{actual} \;\leq\; T_{target} \tag{2}$$

in which T_{target} is the target time and T_{actual} is the actual time used by the time critical algorithm operating in the region of interest.

In the next two sections we give examples of time critical compute and render algorithms of solvent surfaces.

3.2 Time critical computation of solvent surfaces

An algorithm for the computation of smooth molecular surfaces is described Varshney and Brooks in 1993, [10]. The algorithm uses the notion of a *feasible cell* to determine those atoms that contribute to the surface of the molecule. A property of this algorithm is that it can compute the surface by considering only the neighbors of an atom that are close enough to effect the probe placement. The complexity of the surface generation algorithm is $O(k\ log\ k)$, in which k is the average number of neighbors per atom.

Although the algorithm can be parallelized, the computation of the surface of a large molecular configuration may still take a few seconds. The goal of our time critical approach is to restrict the computation of the surface to user's region of interest.

This leads to the following time critical algorithm for computing solvent surfaces:

ComputeSurface (**in** C_f, **in** A_{target})
$n \leftarrow 0$

$A_{actual} \leftarrow 0$
while $A_{actual} \leq A_{target}$ **do**
foreach $A_i \in C(n, C_f)$
GenSurface ($A_i \bigcup$ Neighbors (A_i))
$A_{actual} \leftarrow 1 + A_{actual}$
$n \leftarrow n + 1$
return A_{actual}

$ComputeSurface$ is parameterized with C_f and a time critical target parameter A_{target}, the target number atoms to be processed (determining A_{target} is discussed in the next section). After initialization the main loop will repeatedly call the procedure $GenSurface$ with each atom and its neighbors in the concentric ring. The test in the loop will fail when $A_{actual} > A_{target}$ is reached. The actual number of processed atoms is returned.

For each atom, the procedure $GenSurface$ returns a list of triangles that contribute to the molecular surface.

The algorithm can be refined by decreasing the granularity of the for-loop for testing if $A_{actual} > A_{target}$ after calling $GenSurface$. This refinement will lead to a more accurate A_{actual}.

3.3 Time critical rendering of solvent surfaces

The time budget for time critical rendering of solvent surfaces is controlled by the number of triangles that can be rendered per frame. Each atom may or may not contribute triangles to the solvent surfaces. We denote $\bigtriangledown(A_j)$ as the number of triangles that atom A_j contributes to the solvent surfaces.

This leads to the following time critical algorithm to render a solvent surface:

RenderSurface (**in** C_f, **in** $\bigtriangledown_{target}$)
$n \leftarrow 0$
$\bigtriangledown_{actual} \leftarrow 0$
while $\bigtriangledown_{actual} < \bigtriangledown_{target}$ **do**
foreach $A_i \in C(n, C_f)$
render ($\bigtriangledown(A_i)$)
$\bigtriangledown_{actual} \leftarrow \bigtriangledown_{actual} + \bigtriangledown(A_i)$
$n \leftarrow n + 1$
return $\bigtriangledown_{actual}$

$RenderSurface$ is parameterized with C_f and the time critical target parameter $\bigtriangledown_{target}$, the target number triangles (determining $\bigtriangledown_{target}$ is discussed in the next section). The main loop will repeatedly renders the triangles in concentric rings around the focus cell. The test of loop will fail when the $\bigtriangledown_{actual} > \bigtriangledown_{target}$. The actual number of rendered triangles is returned.

4 Determining time critical parameters

The zonal map is implemented in PVR, a bus-based architecture for portable VR applications, [11]. A PVR configuration consists of one or more processes attached to the bus. Processes synchronize by publishing events to the bus which dispatches these events to subscribing processes. Shared memory is used to share data structures.

A sample PVR configuration for the zonal map is shown in figure 2. It consists of 4 processes: one file, one compute, one render and head tracking process. In this example, the file process reads in the molecule PDB data sets from disk and constructs the zonal map data structure. After the data set is read, the file process will publish the `TIME_STEP` event to the bus. The head tracker process manages the head positions, publishing `HTRACK` events

upon each change. The compute process (subscribed to the TIME_STEP and HTRACK event) recomputes the solvent surface and publishes a NEW_SURFACE event when completed. The render process redraws the surface (triggered by the HTRACK or NEW_SURFACE event).

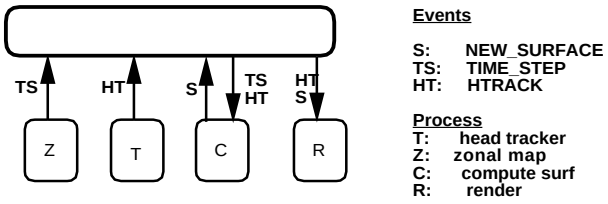

Figure 2: PVR zonal map configuration.

In addition to event dispatching, the bus manages the actual running times of a process. The entire time budget, T_{target} for all the processes is distributed among the time critical processes; i.e. in the simple case that processes are executed sequentially this would be $T_{target} = \sum_i T^i_{target}$. [2] The bus uses the following heuristics in order to realize the constraint in equation 2: When

$$T^i_{target}/T^i_{actual} > 1.1$$

then T^i_{target} is increased by $T^i_{actual} * 1.1$. When

$$T^i_{target}/T^i_{actual} < 0.9$$

then T^i_{target} is decreased by $T^i_{actual} * 0.9$.

Given each time budget T^i_{target}, the bus also determines time critical parameters. In the case of the compute surface algorithm the new target number of atoms, A_{target}, will be adjusted as $T^i_{target} * A_{actual}$. In the case of the render surface algorithm the new target number of triangles, $\bigtriangledown_{target}$, will be adjusted as $T^i_{target} * \bigtriangledown_{actual}$.

5 Results

The timings have been done on a 4 CPU SGI Onyx2 with InfiniteReality graphics using PVR.

Table 1 shows timings for the computation and rendering of the molecular surface for various molecules. A probe-radius of 0.14 nm was used for the computation of the surface. The molecules for which we have made these studies are crambin (1crn), photoactive yellow protein (2phy), chloroperoxidase(1vnc), FABD44.1/Lysozyme complex (1mlc). [3] The third column gives the number of triangles that have been generated for the complete surface. Column four and five show the render and compute times for the rendering and generating complete surface. Column six gives the measured number of triangles that are rendered given a target time of 0.035 seconds.

Table 1 experimentally provides the motivation of using the zonal map and time critical computing for large, time dependent molecular configurations:

[2] More complicated cases with two or more parallel processes are possible by changing the entire budget. $T_{target} = max(T_0, T_1) + T_2$ is an example of distributing the entire time budget among two parallel processes T_0, T_1 sequentially followed by T_2.

[3] The data of these configurations can be obtained from the Protein Data Bank at http://www.rcsb.org/pdb.

PDB	atoms	triangles	render (s)	compute (s)	TC render (0.035 s)
1crn	327	16624	0.01	0.4	16608
2phy	976	44306	0.04	0.9	44012
1vnc	4485	177037	0.14	5.4	35373
1mlc	8582	356086	0.33	11.1	35366

Table 1: The number of surface triangles for various molecules. Column 3 shows the number of triangles for the complete surface. Column 4 and 5 show the render and compute times for the complete surface. Column 6 shows the number of triangles that can be rendered with a target time of 0.035 seconds.

- The compute time for the complete surface of a moderately large molecular configuration is orders of magnitude too high to be able to generate in a virtual environment. This might be somewhat amortized by using a parallel version of the surface generation algorithm.
- Generating the surface takes substantially longer than the rendering of the surface (measured on a SGI Onyx2). Applying level of detail or simplification algorithms to the generated surface will only decrease the render time of the surface.

Figure 3 and 4 experimentally illustrates the dynamic behavior of the zonal map and time critical algorithms. In this experiment we rotate the molecule photoactive yellow protein (2phy) with respect to a fixed point. This simulates a situation in which the user's line of sight is fixed while the molecule is rotated 360 degrees. A new focus cell is computed for each new angle. The X-axis in each plot denotes the rotation angle. In figure 3, the Y-axis for the plots denote (a) the frames per second, (b) the number of triangles rendered (dashed line) versus the number of visible triangles (solid line) for the complete surface. (c) the number of triangles rendered (dashed line) versus the number of visible triangles (solid line) using the zonal map.

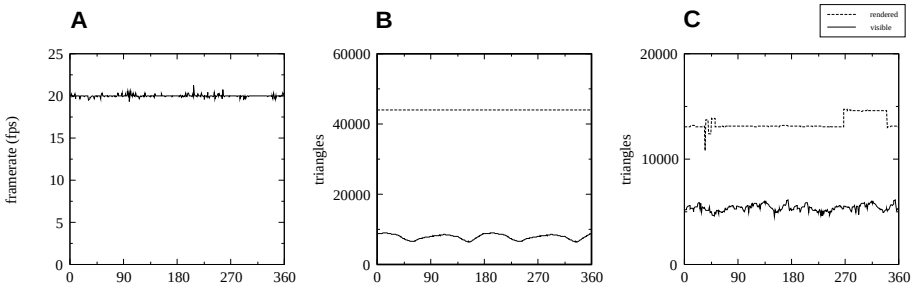

Figure 3: Time critical surface rendering with a varying viewing angle, subject to the time critical parameter $\bigtriangledown_{target}$

The goal of the experiment was to maintain a constant frame rate of 20 frames per second by adjusting $\bigtriangledown_{target}$ when necessary:

1. Plot A shows that the actual frame rate for rendering can be kept constant.
2. Plot B shows that the number of visible triangles (solid line) varies between 7K and 10K triangles. [4] The difference between rendered and visible triangles is due to the

[4] The number of visible triangles has been computed by coloring triangles with identifier information and reading this information from pixels in the frame buffer.

way algorithm generates and tessellates the convex spherical, concave spherical, and toroidal surface patches.

3. Plot C shows that actual number of rendered triangles, $\bigtriangledown_{actual}$, varies between 11K and 15K per frame (dashed line). The number of visible triangles (solid line) varies between 4K and 6K triangles.

   Comparing plot B and C shows that the number of triangles rendered with the zonal map is significantly less that the number of triangles rendered for the complete surface.

   In addition, the percentage of visible triangles differs substantially in plot B and C. In the case of the complete surface, approximately 22% of the rendered triangles are actually visible, while in the case of the zonal map, approximately 40%. This suggests that the effective amount of work done while using the zonal map is higher than without using the map.

In figure 4, the Y-axis for the plots denote (a) the frames per second, (b) the number of atoms used (excluding neighbors). (c) the number of triangles generated per second. The target time was set to 5 frames per second.

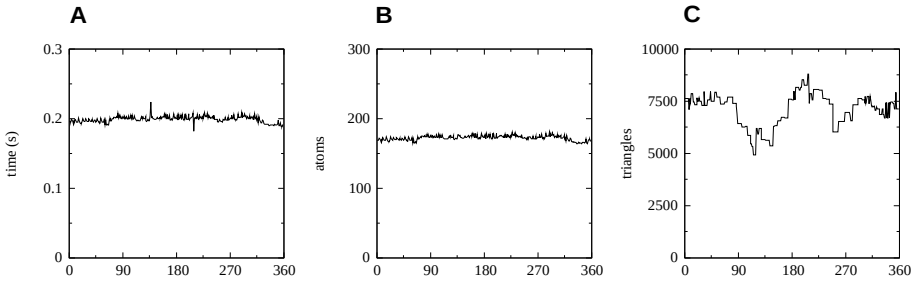

Figure 4: Time critical surface generation with a varying viewing angle, subject to the time critical parameter A_{target}

The goal of the experiment was to maintain a constant frame rate of 5 frames per second by adjusting A_{target} when necessary:

1. Plot A shows that the actual surface generation frame rate can be kept constant.

2. The actual number of atoms, A_{actual}, contributing to the surface can be kept constant.

3. The generated number of triangles varies substantially. This is due to areas in the molecule in which the positions of the atoms are dense, resulting in excessive small cavities and many overlapping van der Waals radii.

   Plots B and C shows that the number of triangles is not an accurate measure to use for a time critical parameter and that the number of atoms is preferred.

These results show that the user can control the target number of frames for generating and rendering the surface. Users can independently set the surface generation and the rendering frame rates. In particular, the update rate of a molecular dynamics simulation is very different than the update rate of the renderer.

Figure 5 shows the result of the zonal map when applied to rendering the solvent surface of chloroperoxidase(1vnc). The image on the left shows the complete solvent surface consisting of 177K triangles. The image on the right use the zonal map. The surface consists of 41K triangles, and can be rendered at 10 frames per second. The focus cell is

located at the small spot in the middle of the right image. A wire frame representation is used to show the parts of the molecule for which no surface is rendered. Although it is less apparent from the projections in figure 5, the distance from the eye point to the wire frame is large compared to the distance to the focus cell.

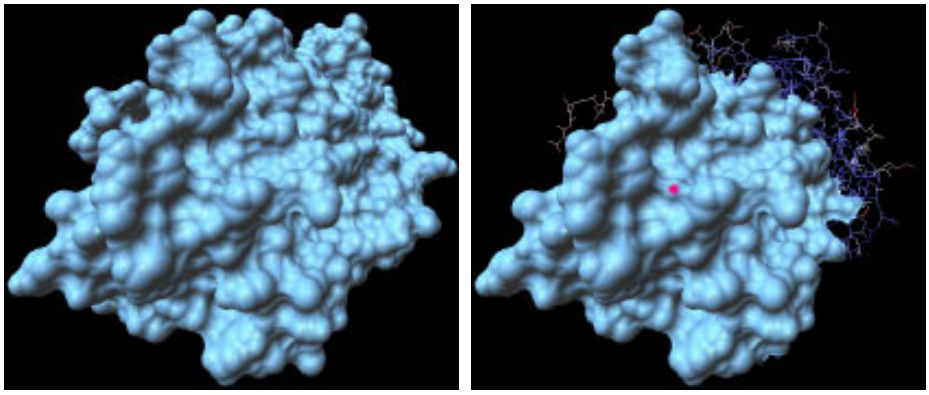

Figure 5: Solvent surface rendering without zonal map (left) and with the zonal map (right) for the 1vnc protein. The shown surfaces have 177K and 41K triangles. The focus cell is draw as a small spot in the right image. A wire frame shows the parts of the protein for which no surface is rendered.

6 Conclusions

There are a vast number of visualization methods to represent molecules in a virtual environment. However, the time needed to generate surfaces of large, time dependent, molecular configuration is orders of magnitude too high. Applying level of detail or simplification algorithms to the generated surface will only decrease the render time of the surface. In this paper we have presented the zonal map, a data structure used for time critical computing of molecule representations. The governing idea is to use the user's line of sight to define a region of interest on which time critical algorithms can be applied. We have given examples of time critical algorithms for generating and rendering a molecular surface. By comparing the number of rendered triangles we have shown how the time critical versions of the algorithms compare to the non-time critical versions.

There are three lessons we have learned from this work. First, uniform frame rates can be realized for the time critical computation and rendering of solvent-accessible surfaces of protein molecules. Second, by allowing an external controller to determine time critical parameters, many combinations of time critical algorithms can be coordinated. Third, as a side effect of view dependent data culling, substantial savings on the amount of work are obtained when computing and rendering surfaces. In addition, the number of visible triangles in the region of interest can approach the number of visible triangles of the complete surface.

In the future, we will investigate how the zonal map can be used in interactive simulation. For example, we envision that the viewing direction can be used for steering adaptive grid refinement techniques, in which fine grain grids are used in the region of interest and coarse grids are used elsewhere.

References

[1] M.C. Surles. Techniques for interactive manipulation of graphical protein models. Technical Report TR92-016, Univ. of North Carolina, 1992.

[2] F.P. Brooks. Project GROPE – Haptic displays for scientific visualization. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 177–185, 1990.

[3] Z. Ai and T. Fröhlich. Molecular dynamics simulation in virtual environments. *Eurographics Forum (Proceedings Eurographics '98)*, 17(3):207–217, 1998.

[4] H. Haase, J. Strasser, and F. Dai. VR techniques for the investigation of molecule data. *Computer & Graphics*, 20(2):207–217, 1996.

[5] C. Cruz-Neira et. al. Scientists in wonderland: a report on visualization applications in the CAVE virtual reality environment. In *Proceedings of the IEEE Symposium on Research Frontiers on VR*, pages 59–66, 1993.

[6] T.A. Funkhouser and C.H. Séquin. Adaptive display algorithms for interactive frame rates during visualization of complex virtual environments. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, 1993.

[7] S. Bryson and S. Johan. Time management, simultaneity and time-critical computation in interactive unsteady visualization environments. In R. Yagel and G. Nielson, editors, *Proceedings IEEE Visualization'96*, pages 255–261, 1996.

[8] F.M. Richards. Areas, volumes packing and protein structure. *Ann. Rev. Biophysics. Bioengineering*, 6:151–176, 1977.

[9] M.L. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16:548–558, 1983.

[10] A. Varshney and F.P. Brooks Jr. Fast analytical computation of Richard's smooth molecular surface. In G.M. Nielson and D. Bergeron, editors, *Visualization '93 (Proceedings of the 1993 Visualization Conference)*, pages 300–307, 1993.

[11] R. van Liere and J.D. Mulder. PVR - an architecture for portable VR applications. In M. Gervautz, A. Hildebrand, and D. Schmalstieg, editors, *Virtual Environments '99, Proceedings of the Virtual Environments Conference & 5th Eurographics Workshop*, pages 125–135. Springer Verlag, 1999.