Interactive Volume Rendering of Functional Representations in Quantum Chemistry

Yun Jang, *Member, IEEE*, and Ugo Varetto

Abstract— Simulation and computation in chemistry studies have been improved as computational power has increased over decades. Many types of chemistry simulation results are available, from atomic level bonding to volumetric representations of electron density. However, tools for the visualization of the results from quantum chemistry computations are still limited to showing atomic bonds and isosurfaces or isocontours corresponding to certain isovalues. In this work, we study the volumetric representations of the results from quantum chemistry computations, and evaluate and visualize the representations directly on the GPU without resampling the result in grid structures. Our visualization tool handles the direct evaluation of the approximated wavefunctions described as a combination of Gaussian-like primitive basis functions. For visualizations, we use a slice based volume rendering technique with a 2D transfer function, volume clipping, and illustrative rendering in order to reveal and enhance the quantum chemistry structure. Since there is no need of resampling the volume from the functional representations, two issues, data transfer and resampling resolution, can be ignored, therefore, it is possible to interactively explore large amount of different information in the computation results.

Index Terms—Quantum Chemistry, GTO, Volume Rendering, GPU

1 Introduction

Quantum chemistry applies quantum mechanics and field theory concepts to give a complete and detailed description of the electronic structure of a chemical system. Detailed electronic structure information is key in understanding chemical reactions and, in general, the physical properties of materials. Applications of quantum chemistry include predicting or confirming radiation spectra, studying chemical reaction to understand how two molecules (e.g., possible drug with enzymes) can interact, molecular design, and computational materials science. Quantum chemistry is based on the Schrödinger equation in which electrons are considered as wave-like particles whose behavior is mathematically represented by a set of wavefunctions obtained by solving the Schrödinger equation that defines the state of a physical system at atomic scale. The time-independent Schrödinger equation for a one particle system is defined as

$$\left(\frac{-h^2}{8\pi^2 m}\nabla^2 + \mathbf{V}\right)\psi(\vec{\mathbf{r}}) = \mathbf{E}\psi(\vec{\mathbf{r}}) \quad (1)$$

where h is Planck's constant, m is the mass of the particle, $\mathbf{V}$ is the potential energy, $\mathbf{E}$ is the total energy and $\vec{\mathbf{r}}$ represents a position in 3-D space. The quantity $|\Psi(\vec{\mathbf{r}})|^2$, where $\Psi(\vec{\mathbf{r}})$ is a solution to Equation 1, gives the probability of finding the particle at position $\vec{\mathbf{r}}$. Analytical solutions to Equation 1 are only available for very simple systems. Quantum chemistry computations focus on finding approximated solutions to the Schrödinger equation for multi-atom systems. Solutions to Equation 1 are used to define properties, represented as scalar fields, such as electron density, electrostatic potential and molecular orbitals.

To properly analyze the output of quantum chemistry computations, 3D visualizations of atomic orbitals (AOs), molecular orbitals (MOs), electron and spin density, and electrostatic potential are required. Visualization tools currently available in programs like *Molden* [26] or *Molekel* [20] display scalar fields by first sampling the solutions, which are functional representations, on 3D regular grids and then building isosurfaces using triangulations for specific isovalues. The resampling process prior to the volume rendering of molecular data raises the following challenges: data storage, data transfer, and evaluation speed of the functional representations. Since there are multiple atomic and molecular orbitals, it is almost impossible to interactively render all combinations of these atomic and molecular orbitals for atomic and molecular structures. For example, the C_2H_5 molecule is one of our simplest Gaussian type orbital (GTO) data and it has 40 different atomic orbitals and 38 different molecular orbitals. Exploring all different combinations is a daunting task because of the $2^{40} \times 38$ resampling processes required.

In this work, we present a novel method to perform interactive evaluation and volumetric rendering of atomic and molecular orbitals directly on graphics hardwares without resampling. Since the approximate solutions to the Schrödinger equation are sums of basis functions, we evaluate the functional representations on the fly in a fragment program by storing all parameters and coefficients in textures. We apply general volume rendering, volumetric isosurface rendering, illustrative rendering, and volume clipping techniques in order to visualize the atomic and molecular structures. Figure 1 shows our system overview. We extend conventional slice-based volume rendering [32]. Our contributions from this work are presented as follows.

- Direct evaluation of the functional representation for molecular data on graphics hardware – computing accurate gradients and avoiding data transfers from CPU to GPU
- Interactive volume rendering of molecular data without resampling on grid structures – discarding an expensive intermediate process for volume rendering
- Illustrative and volume clipping rendering to show nested volumetric atomic and molecular structures – multiple isosurfaces

In this work, we use two types of basis functions. One is pure Gaussian and the other is Gaussian type orbital (GTO). Gaussian type orbital has been used widely because of its easy shape change according to the orbital, whereas, pure Gaussian is newly proposed because of its simplicity in quantum chemistry computation. However, the proposed technique can further be extended to perform interactive evaluation and rendering of any dataset functionally represented.

We first review previous work in Section 2 and describe data from quantum chemistry computation in Section 3. We then introduce details of our interactive evaluation and visualization of the molecular data and present results produced by our system in Section 4 and 5. Finally conclusion and future directions are described in Section 6.

2 Previous Work

In the molecular research area, there are many visualization studies from drawing simple atoms to rendering volumetric representations.

- *Yun Jang is with ETH Zürich, Switzerland, E-mail: jangy@inf.ethz.ch.*
- *Ugo Varetto is with Swiss National Supercomputing Center (CSCS), E-mail: uvaretto@cscs.ch.*

Manuscript received 31 March 2009; accepted 27 July 2009; posted online 11 October 2009; mailed on 5 October 2009.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org .

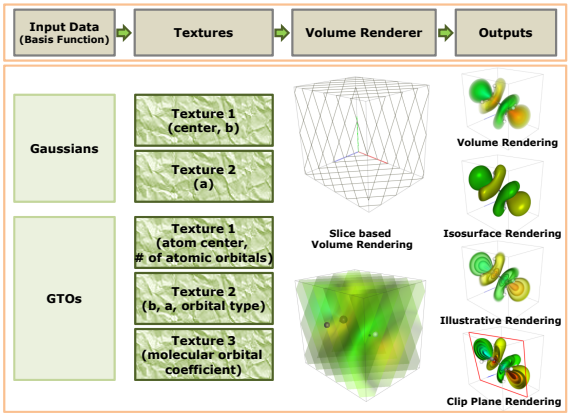


Fig. 1. Overview of our interactive visualization system. Two types of data are stored in textures and evaluated and visualized on our volume rendering. Different rendering results, such as volume rendering, isosurface rendering, illustrative rendering, and volume clipping rendering, are produced using our system.

One approach to show molecular surfaces is to use triangular meshes. Cheng and Shi [4] propose the Restricted Delaunay Triangulation to extract high quality smooth molecular skin surfaces. Another molecular surface representation is studied by Cipriano and Gleicher [5] and they show the abstracted molecular surfaces to provide the boundary of a molecule and the physical and chemical properties at the boundary by extracting the surface abstract from input triangular meshes. Lampe et al. [14] present protein dynamics by a two-level rendering approach. They generate geometry residues on the fly to show interactive protein dynamics with *balls and sticks* for atoms and bonds. When there are many atoms in the visualization, it is difficult to see the overall molecular structure with direct illumination. To enhance visual perception, Tarini et al. [30] present ambient occlusion and edge cueing.

Volume visualization is also applied to molecular data for volumetric structures. Hu et al. [10] present direct volume rendering of protein data and they study an improved transfer function to show various data ranges in the data. In their work, they resample scalar data to 3D regular grids. Lattice-based volume visualization is presented by Qiao et al. [22]. They store all lattice information from quantum dot simulations and visualize electron orbitals in a volume. Their work is based on the sampled lattices which is already provided from the computation, which is a discrete form of data, whereas our approach is to handle the continuous data form. Network based visualization of nanotechnology applications is studied in [23] and electron particles are visualized with volume rendering of electron density volume.

Several researchers present physical and chemical properties of molecular data using visualizations. Lee and Varshney [15] represent thermal vibrations and uncertainty on the molecular surfaces and visualize the fuzzy molecular surfaces to provide more informative display for a better understanding of protein structure and function. Mehta et al. [19] detect anomalous structures in lattice-based molecular simulation data on regular grid and show and verify the detection with visualization. Similar research is presented by Mehta and Jankun-Kelly [18] on unstructured models of nematic liquid crystals. Another study on cluster detection of molecular dynamics is presented by Grottel et al. [8] with visual verification and analysis. Schmidt-Ehrenberg et al. [27] present molecular conformations by visualizing regular grid molecular data.

In order to emphasize the molecular structures, chemists use different primitives, such as *balls and sticks*, helices and ribbons. Liu et al. [16] present interactive molecule construction on GPU. They reconstruct atoms with spheres interactively with GPU acceleration for the educational purpose. Bajaj et al. [2] study the primitives reconstructed on programmable graphics units by 3D image based rendering. Recently, Stone et al. [28] propose a fast way to resample results of quantum chemistry computation using GPUs and multi-core CPUs. This work is motivated by heavy computation for visualizations of results from quantum chemistry computations. They use graphics hardware to resample the results on regular grids and visualize the resampled volumetric data. The large speedup is achieved compared to different number of CPUs cores and different GPUs. Ufimtsev and Martines [31] use a similar approach using GPUs to evaluate the results without rendering capability, whereas our approach combines both the evaluation and the rendering.

Table 1. Polynomial functions (f_t) used in Equation 3.

Basis Function	Orbital Type	f_t
Gaussian	all	1
GTO	s	1
	p_x	x
	p_y	y
	p_z	z
	d_{xx}	$x^2/\sqrt{3}$
	d_{yy}	$y^2/\sqrt{3}$
	d_{zz}	$z^2/\sqrt{3}$
	d_{xy}	xy
	d_{xz}	xz
	d_{yz}	yz

Volume rendering is widely used and the techniques vary according to input grid structures. Jang et al. [11, 12], however, present reconstruction of volumetric functional representations using graphics hardware without resampling on grid structures. Ebert et al. [7] show procedural textures which are continuous and can be evaluated easily in a volume. Since the quantum chemistry computations generate functional representations of atomic and molecular orbitals, their work motivates our volumetric visualizations of molecular data. Volume illustration is another technique to enhance the visual understanding of volumetric data and it shows enhanced understanding of medical and flow data [6, 29].

3 Data in Quantum Chemistry

Most quantum chemistry programs find an approximated solution to the Schrödinger equation then generate data containing a description of the electron structure of the system under analysis in terms of coefficients to be applied to a set of basis functions as the following.

$$\phi(x,y,z) = a\, f_t\, e^{-b r^2} \tag{2}$$

where a and b are real values and $f_t(x,y,z)$ is a polynomial function. r is the distance between the center of basis function and (x,y,z). $f_t(x,y,z)$ varies depending on orbital types and computation methods. Based on the basis function $\phi(x,y,z)$, atomic and molecular orbitals are reconstructed as described in the following sections.

3.1 Atomic orbitals

An atomic orbital is a mathematical function that describes the behavior of an electron in an atom. This function can be used to calculate the probability of finding any electron of an atom in a region of space surrounding the atom's nucleus. The term may also refer to the region of 3D space where the electron is *most likely* to be.

Each atomic orbital is approximated with a sum of basis functions of the form described in Equation 2 as the following.

$$\chi(x,y,z) = \sum_{i=1}^{M} \phi_i(x,y,z) = \sum_{i=1}^{M} a_i\, f_t(x,y,z)\, e^{-b_i r_i^2} \tag{3}$$

where M is the number of basis functions used to define an orbital and $f_t(x,y,z)$ is a polynomial function defined according to the orbital types (s, sp, p, d, f). The polynomial functions (f_t) for most common orbitals, s, p, and d are summarized in Table 1.

3.2 Molecular orbitals

Molecular orbitals are defined as mathematical functions which do not represent any physical quantity; they are very useful in the qualitative

description of bonding and in studying chemical reactions. The molecular orbital is represented with a linear combination of atomic orbitals as the following.

$$\psi(x,y,z) = \sum_{j=1}^{N} g_j \chi_j(x,y,z) \quad (4)$$

where g_j is a real value (molecular orbital coefficient) and $\chi_j(x,y,z)$ is an atomic orbital introduced in Equation 3. N is the number of the atomic orbitals.

3.3 Data Structure

As mentioned in Section 1, we use two different basis functions in this work. In the case of Gaussian basis functions, data are stored as sets of center (x,y,z), exponent (b), and coefficient (a). Since the Gaussian basis function has one polynomial function shown in Table 1 it is not possible to represent different types of atomic orbitals with one common center of a basis function for different atomic orbitals. Therefore, each Gaussian has its own center. Note that there is no molecular orbital coefficient in the Gaussian data format. On the other hand, data represented with Gaussian type orbitals (GTOs) has common centers only at the atom centers since the orbital types can be represented by the polynomial functions shown in Table 1. The other parameters (b's and a's) and orbital types are stored afterward followed by molecular orbital coefficients (g's). The number of molecular orbital coefficients corresponds to the number of atomic orbitals. For example, let us assume that there are 2 s-orbitals, 3 p-orbitals, and 2 d-orbitals. The total number of atomic orbitals (the number of molecular orbital coefficients) is $2\times 1_{(s)} + 3\times 3_{(p)} + 2\times 6_{(d)} = 23$ since there are 3 orbitals in the p-orbital and 6 orbitals in the d-orbital.

3.4 Data Preprocessing

The coefficients a_i and b_i of atomic orbitals, together with orbital types and the coefficients g_i of the molecular orbitals are read from the output of popular quantum chemistry packages such as *Gaussian* and *GAMESS*, and used to reconstruct the molecular orbitals functions as described in Equation 4. Coefficients are properly normalized to ensure that the probability of finding each electron in the system in the entire 3D space is always equal to 1.

We also find the bond structures among atoms when the data is read and they are sent to the GPU to evaluate *ball and stick*s for atoms and bonds. Since it is still difficult to find the solution of the Schrödinger equation for molecules with many atoms, we are generally dealing with small number of atoms. Therefore, we can afford to compute distances for all pairs of atoms in a 2D square array on the CPU and compare the distance with the covalent radius [1]. If the distance between two atoms is less than the covalent radius, there is a bond between them.

4 Interactive Visualization of Molecular Data

As described in Section 3, the molecular data in quantum chemistry is modeled as a sum of basis functions, such as Gaussians and Gaussian type orbitals (GTOs). The molecular data file contains sequences of basis function parameters including centers, coefficients, exponents, etc.. Some basis functions, such as GTOs, imply different reconstruction equations depending on the orbital types. The reconstruction (resampling) of data values in a certain volume would require time-consuming computations proportional to the selected grid resolutions and the visualization of the reconstructed volume would incur high computational preprocessing with conventional direct volume rendering techniques. Seeing the details of molecular structures requires many combinations of orbitals, which makes it impossible to precompute and store the resampled volumes.

Avoiding completely this resampling approach used for most of molecular visualization research, we reconstruct the volume directly on GPUs by storing the basis function parameters in textures. With our approach, it is possible to show the volumetric molecular structures without transferring massive volumetric data.

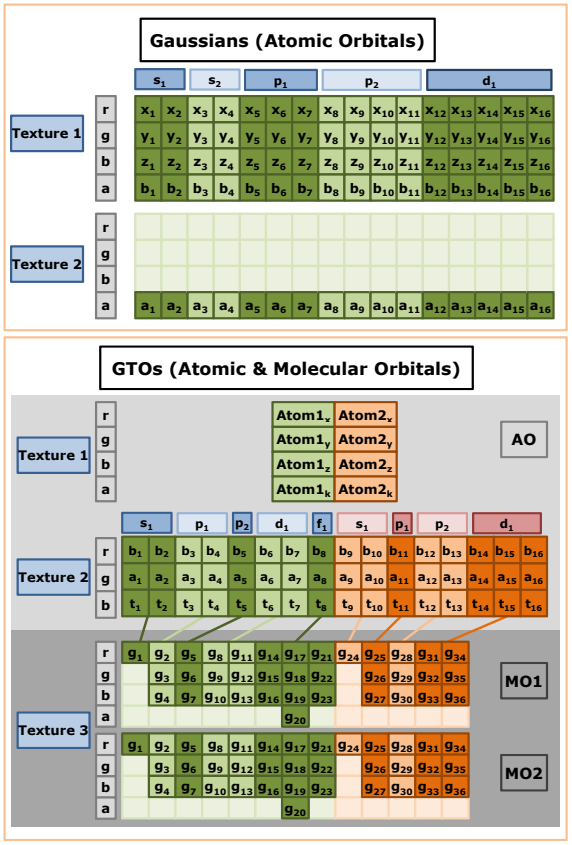


Fig. 2. The parameters of basis functions are packed into 2 or 3 textures. The top image shows an example of texture layouts for 16 Gaussian basis function with 5 atomic orbitals in 2 textures. The bottom image presents our texture packing of GTOs into 3 textures. In the example, there are 2 atoms, 9 atomic orbital types, 36 atomic orbitals, and 2 sets of the molecular orbital coefficients.

4.1 Volume Rendering on GPU

Volume rendering in visualization is a common technique and there are many approaches depending on underlying grid structures. In this molecular visualization, the data does not lie on a specific grid structure. We, therefore, chose the slice-based volume rendering technique, and we evaluate fragments on each slice in a fragment program. The slices are rendered from back to front, so that the color and opacity are properly displayed. The slices are generated by computing intersections between a bounding box and planes in a vertex program proposed in [24]. The number of slices can be adjusted by our user interface. Using this volume rendering base, a 2D transfer function is used to explore the interesting data values in the volume.

4.2 Texture Layouts For Molecular Data

Currently we use two different types of molecular data from quantum chemistry computations. One is formed with all Gaussians and the other is formed with Gaussian type orbitals (GTOs). The Gaussian basis functions have centers, exponents, and coefficients of Gaussians as parameters. On the other hand, GTOs have centers of atoms, exponents and coefficients of atomic orbitals, and coefficients of molecular orbitals. In this work, we focus on two different types of orbitals, namely atomic and molecular orbitals. For the atomic orbitals, only basis function parameters are needed for the reconstruction, whereas the molecular orbitals require one additional parameter, which is the molecular orbital coefficient.

We store all these basis function parameters and corresponding coefficients in 2D textures and fetch the texture values in a fragment program to evaluate data values and gradients of the functional representations. In order to evaluate the functional representation efficiently in a fragment program, we encode all parameters and coefficients as

```
for(i = 0; i < NumOfOrbitalTypes; i++){
 if(ithOrbitalDraw){
  for(j = ithOrbitalStart; j < ithOrbitalEnd; j++){
   // for x,y,z, and exponent
   texValue1 = texRECT(Texture1, texpos(j));
   // for coefficient
   texValue2 = texRECT(Texture2, texpos(j));
   r = inpos.xyz - texValue1.xyz;
   // Data value
   f = texValue2.w *
       exp((-1.0) * texValue1.w * dot(r, r));
   // Gradient
   df = (-2.0) * texValue1.w * f * r;
   val += float4(df, f);
}}}
```

Fig. 3. Main loop of pseudo Cg fragment program for the evaluation of the atomic orbital with Gaussian basis functions.

shown in Figure 2. Since we use two types of basis functions, we introduce two different texture layouts, one for Gaussians and the other for GTOs.

The top layout in Figure 2 shows how we encode the texture for Gaussian basis functions. Since Gaussian basis function does not imply different polynomial functions according to atomic orbital types, we simply store the centers (x_i, y_i, z_i) and exponents b_i in one texture, and coefficients a_i in another texture. Therefore, simple fetching the texture values in a fragment program is possible in order to obtain the sum of all basis functions. Note that there is no molecular orbital coefficient in our molecular data.

In the evaluation of GTOs shown in Equation 3, there is a polynomial term $f_t(x,y,z)$, which is defined according to the atomic orbital type. There is only one common center with different atomic orbital parameters for the different atomic orbital types in one atom. The shape of an orbital is defined by the polynomial term and the polynomial terms can be multiple functions for p, sp, d, and f. For example, the p-orbital has 3 polynomial terms (p_x, p_y, p_z) and the d-orbital has 6 polynomial terms (d_{xx}, d_{yy}, d_{zz}, d_{xy}, d_{xz}, d_{yz}). Each polynomial term has its own molecular orbital coefficient. The bottom in Figure 2 is an example of the texture layouts for the atomic orbitals and molecular orbitals with 2 GTO basis functions. In the example, there are 2 atoms, 9 atomic orbital types with 16 sets of parameters, and 2 sets of molecular orbital coefficients for 36 atomic orbitals. There are multiple basis functions for one orbital type. In this example, the first s-orbital ($s1$) for Atom 1 has 2 sets of parameters and the first d-orbital for Atom 2 ($d1$) has also 3 sets of parameters. The number 36 is calculated as $1_{(s1)}+3_{(p1)}+3_{(p2)}+6_{(d1)}+10_{(f1)}+1_{(s1)}+3_{(p1)}+3_{(p2)}+6_{(d1)}$. As shown in the figure, we put the atom centers and the number of orbital types in the texture 1. Texture 2 is composed of exponents, coefficients, orbital types for atomic orbitals. Then we store the molecular orbital coefficients in the texture 3. Texture 1 and 2 are organized in the order of parameters, whereas, texture 3 is designed according to the orbital type. For example, since the s-orbital has only one polynomial term, there is one molecular orbital coefficient, which is stored in R out of RGBA. the p-orbital has 3 polynomial terms, therefore, we place three molecular orbital coefficients in one RGB in order to reduce the texture fetch. In the same way, for the d-orbital, we use two RGB's for the 6 molecular orbital coefficients.

4.3 Per-Fragment Reconstruction

In order to evaluate fragments, we use a high level language, NVIDIA Cg [21], and the Cg code is compiled on the fly after the molecular data is read. With the support of Cg language, we use `if` statement to choose the appropriate fragment program for the functional evaluations depending on the basis functions. Since we have two different basis functions, we show two different fragment programs according to the basis functions. The functional values as well as gradients are evaluated at the same time.

For Gaussian basis functions two textures, as shown in the top of Figure 2, are fetched for all parameters including the centers of basis functions and the atomic orbitals are evaluated using Equation 3. Figure 3 presents a pseudo Cg code for the reconstruction of Gaussian basis functions. In the Cg code, `ithOrbitalDraw` is connected to our user interface, therefore, we can select/deselect any orbitals on the fly.

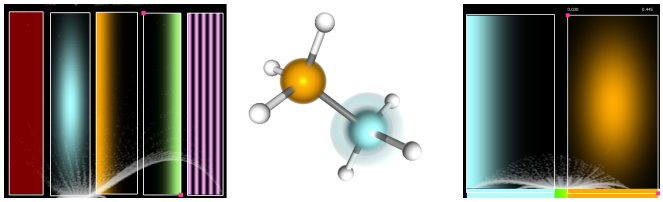
```
n = 0;
for(i = 0; i < NumAtoms; i++){
 // for x,y,z and number of orbital types
 texValue1 = texRECT(Texture1, texpos(i));
 r = inpos.xyz - texValue1.xyz;
 for(j = start(i); j < start(i)+texValue1.w; j++){
  for(k = jthOrbitalStart; k < jthOrbitalEnd; k++){
   // for exponent, coefficient, orbital type
   texValue2 = texRECT(Texture2, texpos(k));
   // for s orbital
    ......
   // for p orbital
   else if(texValue2.z == 3){
    if(k == jthOrbitalStart){
     // for 3 molecular orbital coefficients
     texValue3 = texRECT(Texture3, texpos(n));
     // increase offset for Texture 3
     n += 1;
    }
    tmp1 = dot(r, texValue3.xyz);
    tmp2 =  texValue2.y *
        exp((-1.0) * texValue2.x * dot(r, r));
    // Data value
    f = tmp1 * tmp2;
    // Gradient
    df = texValue3.xyz * tmp2 -
         2.0 * texValue2.x * r * f;
    val += float4(df, f);
   }
   // for d orbital
    ......
}}}}
```

Fig. 4. Main loop of pseudo Cg fragment program for the evaluation of the molecular orbital with GTO basis functions. In this code, only p-type molecular orbital computation is shown because it shows the efficiency of our texture layout for the molecular orbital coefficients.

For the GTO basis functions, each fragment is evaluated by computing either Equation 3 for the atomic orbitals or Equation 4 for the molecular orbitals. The atomic orbitals are computed by two texture lookups (Texture 1 and Texture 2 from GTOs texture layout in Figure 2). We fetch Texture 1 for the atom centers and the number of atomic orbital types. Then we fetch Texture 2 for the exponents, coefficients and atomic orbital types of each atomic orbital. Once we have all parameters, Equation 3 is evaluated with the polynomial functions shown in Table 1 in the `for` loop.

On the other hand, the molecular orbital evaluation needs one more computation based on the atomic orbital evaluation as described in the previous paragraph. We fetch one more texture (Texture 3) for the molecular orbital coefficients and multiply the coefficients right after the atomic orbital evaluation. Figure 4 shows a pseudo Cg fragment program for this molecular orbital evaluation. Specifically we show the functional value and gradient calculation with p-orbital in order to show the efficiency of our texture layout (Texture 3 from GTOs texture layout in Figure 2). As shown in the Cg code, we can evaluate 3 different p-orbitals (p_x, p_y, p_z) at the same time by fetching 3 molecular coefficients by one texture lookup. Similar computation is applied to d and f orbitals.

Once the value and gradient of a fragment are computed, we fetch the transfer function texture for color and opacity. Then we apply illumination, illustrative rendering techniques, and volume clipping technique in the fragment program.



(a) (b) (c)

Fig. 5. Transfer function (TF) comparison. (a) presents our five different transfer functions (uniform, Gaussian, the right half of Gaussian, the left half of Gaussian and sinusoidal from the left). (c) shows our 2D TF setting to generate an image in (b). The right half of Gaussian TF (sky blue) is compared with simple Gaussian TF (orange). It is possible to see the nested orbital structures at the core of atom with the right half of Gaussian TF.

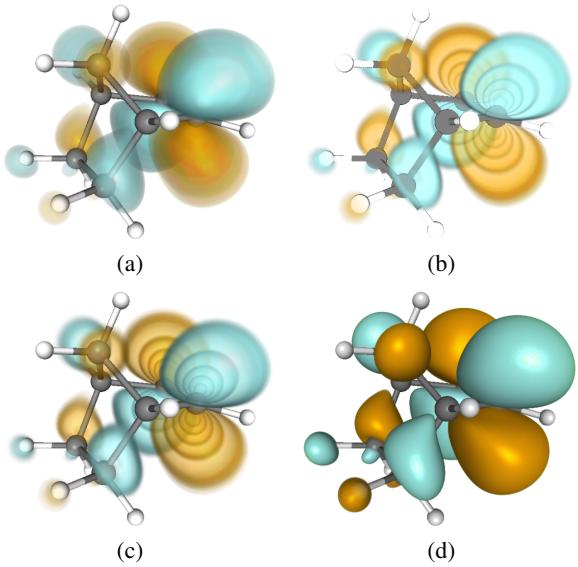
4.4 Ball and Stick Rendering

Most of molecular visualizations provide drawing of the atoms and bonds between atoms and common primitives to draw atoms and bonds are *balls and sticks*. Many molecular visualization tools represent *balls and sticks* with triangular meshes and render the meshes with other features, such as isosurfaces. In order to render the meshed *balls and sticks* together with the volumetric representation, however, visibility test with sorting is necessary for proper rendering results with a single rendering pass. In this work, we avoid the visibility test by evaluating the *balls and sticks* as functional forms. Balls are represented as spheres and sticks are represented as cylinders. The sphere and cylinder equations are evaluated as solid volume prior to the evaluation of the functional representations. If a fragment is inside either a sphere or a cylinder, then we avoid the expensive evaluation of the functional representation.

The ball radius is read from an atom element table [3], which is the van der Waals radius, and the radius varies depending on atomic numbers. The stick radius is set as a half of hydrogen's van der Waals radius, which is the smallest in the table. We also provide a control of both radii in our user interface, so that users can change the relative size. The atom color is also read from the atom element table, so that chemists understand the molecular structures easily as they have been seeing.

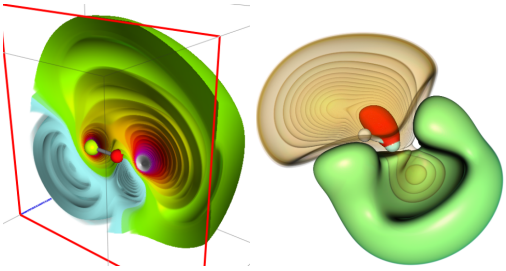
4.5 Transfer Function and Illustrative Rendering

In this volume rendering of quantum chemistry data, multiple isovalues are preferable since more isovalues show detail of the molecular structures. However, the higher absolute isovalues are nested in the lower absolute isovalues in the molecular data. In order to show the internal structures of a molecule clearly, we design 5 different 2D transfer functions (TFs), including uniform, Gaussian, left half of Gaussian, and right half of Gaussian, sinusoidal as shown in Figure 5 (a). The uniform TF is appropriate to render isosurfaces and the Gaussian TF is used for the volume rendering of the molecules. Specifically the right and left half of Gaussian TF are preferable to see the nested orbital structures since the highest or lowest values, which are found mostly at the cores of atoms, are inside outer shells. The right half of Gaussian TF is used for the negative values and the left half of Gaussian TF is used for the positive values. Figure 5 (b) and (c) present the transfer function comparison of the right half of Gaussian TF and whole Gaussian TF with one of GTO basis function data (C_2H_6). The right half of Gaussian TF shows the core of orbital, whereas, whole Gaussian TF hides the internal structures. We also apply illustrative rendering techniques to the molecular data visualization, such as enhancing boundaries with the sinusoidal TF proposed in [6, 29]. Note that we refer to the works in [6, 29] for the detail. The sinusoidal TF is used to show multiple isovalues (e.g., contour volume) in order to see the atomic and molecular orbital structures. Since visualizing more isovalues at the same time is preferable, the sinusoidal TF with boundary enhancements is used to show the contour volume in an illustrative way, so that we can provide more isovalues and structures in the volume. Figure 6 shows rendering results of the 27^{th} molecular orbital with GTO basis function data ($C_4H_2CH_2CH_2CH_4$). Note that warm color (orange) indicates positive values and cool color (sky blue) represents the negative values of the molecular orbital (Equation 4). In the figure, (a) shows a volume rendering of the molecular orbital and (b) presents a volume rendering with contours using the sinusoidal TF. (c) is a rendering result of sinusoidal TFs with the boundary enhancement technique. (d) shows an isosurface rendering. Especially the boundary enhancement with the sinusoidal TF produces very clear multiple isovalue structures of the nested molecular data.



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

Fig. 6. Volume rendering of the molecular orbital for $C_4H_2CH_2CH_2CH_4$. Warm color represents positive values and cool color represents negative values. (a) is a conventional volume rendering with local illumination and (b) is a volume rendering with the edge coloring using the sinusoidal TF without illumination. (c) shows a boundary enhanced contour volume rendering with the sinusoidal TFs and (d) presents an isosurface rendering with local illumination.

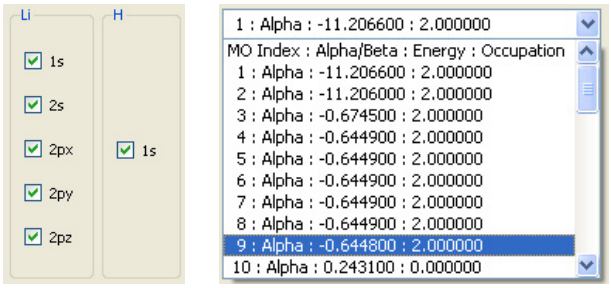


(a) (b)

Fig. 7. (a) is a contour volume rendering of the atomic orbitals for BeO (Be is the green atom) with the volume clipping technique. (b) is a boundary enhanced illustrative rendering (positive values, orange) with an isosurface rendering (negative value, green) for HF (H is the white atom). The highest positive value is rendered with the red isosurface.

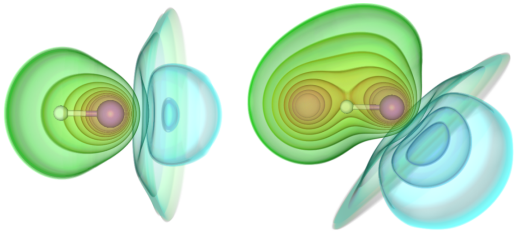
4.6 Volume Clipping

Volume clipping is a technique to hide unimportant parts in the volume rendering. In Section 4.5, we mention that visualizing multiple isovalues is preferable to show the molecular structures. The left image in Figure 7 shows our volume clipping rendering with Gaussian basis function data (BeO) and the atomic orbital structure in the middle of the volume is clearly shown by clipping a half of the volume out. Users can adjust our clipping plane in arbitrary normal direction and place the clipping plane where they are interested. We decide whether a subvolume is visible or not by evaluating the clipping plane equation and the subvolume location in our fragment program. Then we use `clip` function from Cg library to remove unwanted subvolumes.



(a) (b)

Fig. 8. User interfaces of the atomic and molecular orbitals. (a) is our user interface to choose the atomic orbitals. (b) is the interface for the molecular orbitals and it shows spin, energy, and occupation of each molecular orbital in order to help users select interesting molecular orbitals.



(a) (b)

Fig. 9. Renderings of the atomic orbitals for a molecule (LiH, H is the white atom) with boundary enhanced volume contours. (a) shows the atomic orbitals rendered with only $1s$, $2s$, and $2p_x$ of Li. (b) shows the atomic orbitals rendered with $2p_y$ of Li, and $1s$ of H on top of (a).

4.7 User Interface

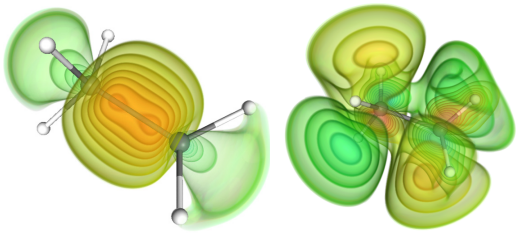
In our system, there are mainly four parts of user interfaces including data loader, colormap loader, transfer function control and rendering control. Data loader and colormap loader parts are simple file loading interfaces. In the transfer function control, we design the user interface to control multiple transfer functions with 5 different transfer function modes, which is described in Section 4.5. Also data values, such as minimum and maximum, can be adjusted for various data ranges over many atomic and molecular orbitals in the same data. In the rendering control, we can select lighting modes, illustrative rendering mode, specific atomic orbitals, and molecular orbitals, and control the clipping plane for the volume clipping. Especially the atomic orbital and molecular orbital selection interfaces can be used to explore the various atomic and molecular orbitals interactively. Figure 8 presents our atomic and molecular orbital interfaces. Users can select any of atomic orbitals (a) and also choose any of molecular orbitals based on the information shown in (b).

5 Results and Discussion

We have implemented our system on Core 2 Quad CPU 2.4GHz processor with NVIDIA GeForce GTX 260 graphics hardware. We have extended our slice-based volume rendering system by evaluating the functional values directly on GPU. Ray casting [9, 13, 25] could be also used for this application and easily integrated into our system since the ray casting has advantages such as adaptive sampling. One issue in our slice-based volume rendering is the number of slices to reveal all properties of quantum chemistry study. However, the resampling with very high resolution takes up to several hours and it is difficult to visualize the high resolution data on a desktop PC. In this sense, changing the number of slices in our system is much easier and faster than resampling and transferring the resampled data.

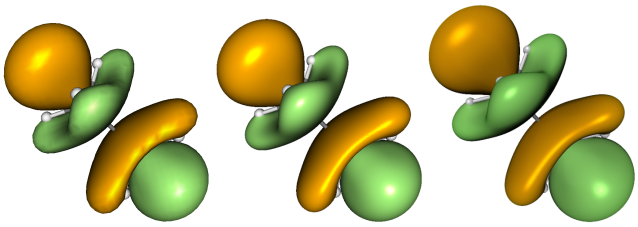
We have tested various datasets summarized in Table 2 and 3. We specify the number of basis functions with the number of corresponding parameters.

BeO, HF, and LiH are datasets using Gaussian basis functions. BeO is presented in Figure 7 (a) with multiple isosurfaces and volume clipping. Clipping along the bond between two atoms shows the internal orbital structures of BeO. Figure 7 (b) is a rendering result of HF. Positive values of the atomic orbitals are rendered as contour volume (orange) and a negative value is shown as an isosurface (green). In the image, some of the positive orbitals are nested in the negative orbitals. (a) and (b) in Figure 9 are progressive atomic orbitals of LiH with boundary enhanced volume contours. (a) shows a combination of $1s$, $2s$, and $2p_x$ of Li, whereas (b) is a rendering result of $2p_y$ of Li, and $1s$ of H on top of (a). Comparing two images, we can see that the $1s$ of H orbital (left core) and $2p_y$ of Li orbital (rotation of negative values) change the atomic orbital structure.



(a) (b)

Fig. 10. Molecular orbitals (C_2H_5, H is the white atom). (a) is the 3^{rd} molecular orbital and (b) is the 21^{st} molecular orbital.



(a) (b) (c)

Fig. 11. Two isosurface, 0.5 (orange) and -0.5 (green), comparison of *Molekel* [20] and our system. (a) and (b) show low (31x24x24) and high (170x128x137) resolutions of the isosurface using *Molekel*. (c) presents the isosurface rendering using our system.

C_2H_5, C_2H_6, $C_4H_2CH_2CH_2CH_4$, and $N_2C_4O_2H_4N_2C_4O_2H_4$ have Gaussian type orbital (GTO) as the basis function. C_2H_5 has 38 molecular orbitals and Figure 10 shows the 3^{rd} and 21^{st} molecular orbitals with contour volumes. The energy level of the 3^{rd} molecular orbital is -0.7074 with two occupation and Alpha spin, whereas the energy level of the 21^{st} molecular orbital is 0.9449 with zero occupation and Alpha spin. C_2H_6 data is used in Figure 11 compared to the results of *Molekel*. (a) and (b) of the figure are isosurface rendering of isovalues (± 0.05) with different grid resolutions. The grid resolution of (a) is 31x24x24 and that of (b) is 170x128x137. The computation timings for (a) and (b) in *Molekel* are 0.22 seconds and 32.49 seconds. As seen in (a), the low resolution isosurface shows artifacts on the surface due to the low resolution of resampling. Figure 11 (c) is the isosurface rendering using our system. Our system does not produce any artifact since we perform per-fragment evaluation of the functional representations. Figure 12 (a) and (b) show different molecular orbitals of C_2H_6. (a) is rendered with the volume clipping technique and multiple isosurfaces and (b) is generated by the volume contours with the boundary enhancement. Both images present the orbital structures among the atoms and bonds. Figure 6 presents different rendering techniques on $C_4H_2CH_2CH_2CH_4$ and Figure 12 (c) shows the volume clipping and Figure 12 (d) shows the isosurface with the volume contours of $N_2C_4O_2H_4N_2C_4O_2H_4$. The molecule is very complicated but both images show the molecular orbital structures in the volume.

We also measure performances on a viewport of 600×517 with 256 slices. The performances and storages of datasets are summarized in Table 4. Datasets with GTO basis functions are compared with the performances using *Molekel*. The performance for our system indicates the evaluation and rendering speed, whereas, the performance

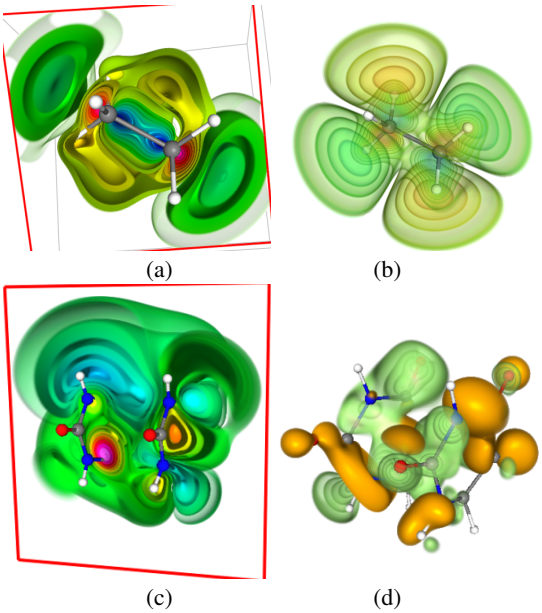


Fig. 12. More rendering results of C_2H_6 (a, b) and $N_2C_4O_2H_4N_2C_4O_2H_4$ (c, d). (a) and (b) are the 17^{th} and 20^{th} molecular orbitals with the boundary enhanced contour volume. (c) shows the isosurface rendering of the 27^{th} molecular orbital with multiple isosurfaces and the volume clipping. (d) presents both isosurfaces (orange) and volume contours with the sinusoidal TF (green) of the 30^{th} molecular orbital.

for the *Molekel* includes only the resampling and triangulation for one isosurface. Note that *Molekel* does not support the data format for Gaussian basis functions and $N_2C_4O_2H_4N_2C_4O_2H_4$ is not readable in *Molekel*. The grid resolution in *Molekel* for the comparison is set as $122 \times 97 \times 97$ and *Molekel* performs resampling in the grid and marching cube for the isosurfaces. As seen in the table, our system provides greater performance improvement compared to *Molekel* although we do not include the rendering speeds in *Molekel*. The performance in our system highly depends on the number of basis functions and the atomic orbital types. Especially d-orbitals require more computation compared to s-orbital or p-orbitals. The storage columns in Table 4 indicate the data sizes, which need to be transferred to GPU. Since we evaluate data values on the fly, only parameters and coefficients of the basis functions are required. The storage in *Molekel*, however, is the size of the resampled data, which could be used in generic volume renderers to generate similar images to our system.

Our system has enabled scientists to interactively analyze atomic and molecular orbitals that give a clear idea of intra-molecular bonding properties such as the sites where bonds are more likely to form (bonding sites) and the sites where bonds are not likely to form (anti-bonding sites). This is achieved by applying a sinusoidal transfer function which results in the rendering of multilevel (i.e. multiple isovalues) colored surface where the intensity of the color can be set to match the probability of finding electrons at a specific point in space. Especially, Figure 10 (a) presents the probability of finding electrons clearly. There is a strong bond between two carbon atoms and there is a weak bond between two hydrogen atoms on the right side, which is not found on the left side of the molecule. The peanut shape of the strong bond was difficult to visualize due to the lack of interactivity in generating many isosurfaces. Also the weak bond is not clearly shown without multiple isovalues. Figure 12 (a) and (b) show how different two molecular orbitals are according to the energy. Our approach also proves usefulness for divulgation purposes by giving an intuitive idea of how atoms might bond or not bond to each other by simply looking at the intensity of the colors. Note that this visualization can only be achieved with proper support for transparency when one has triangulated meshes since the sites where bonds are created are found in the innermost part of the orbital volumes. It is therefore important to show lighter outer areas and inner darker volumes of space at the same time. When applying mesh-based techniques multiple passes are required for correct rendering of overlapping transparent surfaces. Another issue that scientists meet is the performance in the analysis tool. In order to perform the required analysis, scientists must be able to quickly generate a number of molecular orbitals while varying parameters such as isovalues, bounding box, color and transparency, to help in understanding inter- and and intra-molecular bonding properties; this is simply impossible with the currently available tools which may take hours as compared to tenths of seconds with our approach to generate the entire set of orbitals.

Table 2. Three datasets with Gaussian basis functions

Dataset	number of basis functions (x,y,z,b,a)	number of atomic orbitals types	number of atomic orbitals
LiH	20	4	6
HF	66	9	19
BeO	112	12	28

6 Conclusion and Future Direction

We have presented our interactive volume rendering of molecular data by evaluating the functional representations on the GPU. Our system does not require the resampling on grid structures for the volume rendering therefore, there is no data transfer issue for the high grid resolution. Direct per-fragment evaluation of the functional representation in our fragment program allows us to interactively explore the functional representations of molecular data and generate images without artifacts, which are often seen in grid structures. We also use illustrative rendering techniques to show the nested atomic and molecular structures and our user interface enable us to select any of interesting atomic and molecular orbitals.

As seen in Table 4, the performance is degraded as the number of basis functions and parameters increases. A possible solution is to use hierarchical spatial structures proposed in [12]. We will investigate the spatial data structures of the molecular data and we also would like to compare the evaluation quality and the performance with the ray casting as an extension of the spatial structure study in the future. Another future work is related to computing electron density and electrostatic potential. In this work, the atomic and molecular orbitals are computed on the fly. The computation of electron density and electrostatic potential, however, requires many redundant processes. In order to improve the performance for electron density and electrostatic potential, it could be better to use multi-level rendering, so that we can reduce the large amount of redundant computations. Moreover, we would like to study our *ball and stick* rendering with this multi-level rendering because the computation of our visibility test is expensive when we have more atoms and bonds and investigate the surface rendering proposed by Loop and Blinn [17]. Since we can evaluate the scalars and gradients in our fragment program, we are also interested in interactively exploring vector fields in the molecular data, such as topological structures of orbitals and directions of orbital gradient fields.

Acknowledgements

The authors would like to thank Jean Favre, Mario Valle, John Biddiscombe, Maria Grazia Giuffreda and the anonymous reviewers for many helpful discussion and comments. This work was supported in part by the Swiss National Science Foundation under grant 200021_124642.

References

[1] F. H. Allen, O. Kennard, D. G. Watson, L. Brammer, A. G. Orpen, and R. Taylor. Table of bond lengths determined by x-ray and neutron diffraction. *Journal of the Chemical Society, Perkin Transactions 2*, pages S1–S19, 1987.

Table 3. Four datasets with GTO basis functions

Dataset	number of atoms (x,y,z)	number of atomic orbital types	number of basis functions (b,a)	number of molecular orbital coefficients (g)	number of molecular orbitals
C_2H_5	7	27	50	40	38
C_2H_6	8	30	54	42	40
$C_4H_2CH_2CH_2CH_4$	17	62	124	125	118
$N_2C_4O_2H_4N_2C_4O_2H_4$	24	120	856	264	162

Table 4. Comparison of performance and data storage required between our system and *Molekel*. Evaluation and rendering performance for our system is measured on a viewport of 600×517 with 256 slices whereas performance for *Molekel* is measured on a regular grid, $122 \times 97 \times 97$ only for resampling and triangulation of one isosurface. The storage indicates the data size to be transfered to GPU.

Dataset	Our system		*Molekel*	
	speed (sec)	storage (Kbytes)	speed (sec)	storage (Kbytes)
LiH	0.047	0.4	N/A	N/A
HF	0.094	1.3	N/A	N/A
BeO	0.204	2.2	N/A	N/A
C_2H_5	0.108	6.6	11.1	1147.9
C_2H_6	0.140	7.2	12.7	1147.9
$C_4H_2CH_2CH_2CH_4$	0.270	60.2	27.2	1147.9
$N_2C_4O_2H_4N_2C_4O_2H_4$	1.000	178.2	N/A	N/A

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

[3] BlueObelisk. Properties of the elements. Atom element properties from elements.xml distributed at http://sourceforge.net/projects/bodr.

[4] H.-L. Cheng and X. Shi. Guaranteed quality triangulation of molecular skin surfaces. In *Proceedings of the conference on IEEE Visualization 2004*, pages 481–488, 2004.

[5] G. Cipriano and M. Gleicher. Molecular surface abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1608–1615, 2007.

[6] D. Ebert and P. Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *Proceedings of the conference on IEEE Visualization 2000*, pages 195–202, 2000.

[7] D. S. Ebert, K. F. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & Modeling: A Procedural Approach, Third Edition (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, December 2002.

[8] S. Grottel, G. Reina, J. Vrabec, and T. Ertl. Visual verification and analysis of cluster detection for molecular dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1624–1631, 2007.

[9] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. H. Gross. Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005.

[10] M. Hu, W. Chen, T. Zhang, and Q. Peng. Direct volume rendering of volumetric protein data. In *Computer Graphics International*, pages 397–403, 2006.

[11] Y. Jang, R. P. Botchen, A. Lauser, D. S. Ebert, K. P. Gaither, and T. Ertl. Enhancing the interactive visualization of procedurally encoded multifield data with ellipsoidal basis functions. *Computer Graphics Forum*, 25(3):587–596, 2006.

[12] Y. Jang, M. Weiler, M. Hopf, J. Huang, D. S. Ebert, K. P. Gaither, and T. Ertl. Interactively visualizing procedurally encoded scalar fields. In *EG/IEEE TCVG Symposium on Visualization VisSym '04*, pages 35–44, 339, 2004.

[13] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings IEEE Visualization 2003*, pages 287–292, 2003.

[14] O. D. Lampe, I. Viola, N. Reuter, and H. Hauser. Two-level approach to efficient visualization of protein dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1616–1623, 2007.

[15] C. H. Lee and A. Varshney. Representing thermal vibrations and uncertainty in molecular surfaces. In *In SPIE Conference on Visualization and Data Analysis*, pages 80–90, 2002.

[16] F. Liu, Y. Liu, and D. Fordham. Interactive molecule construction with gpu acceleration. In *In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008*, pages 5298–5301, 2008.

[17] C. Loop and J. Blinn. Real-time gpu rendering of piecewise algebraic surfaces. *ACM Transactions on Graphics*, 25(3):664–670, 2006.

[18] K. Mehta and T. Jankun-Kelly. Detection and visualization of defects in 3D unstructured models of nematic liquid crystals. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1045–1051, September/October 2006.

[19] S. Mehta, K. Hazzard, R. Machiraju, S. Parthasarathy, and J. Wilkins. Detection and visualization of anomalous structures in molecular dynamics simulation data. In *Proceedings of the conference on IEEE Visualization 2004*, pages 465–472, 2004.

[20] Molekel. Multiplatform molecular visualization. Software distributed at http://cscs.ch/molekel.

[21] NVIDIA. Cg language specification. Cg Language Specification, available at http://developer.nvidia.com/page/cg_main.html.

[22] W. Qiao, D. S. Ebert, A. Entezari, M. Korkusinski, and G. Klimeck. Volqd: Direct volume rendering of multi-million atom quantum dot simulations. In *Proceedings of the conference on IEEE Visualization 2005*, pages 319–326, 2005.

[23] W. Qiao, M. McLennan, R. Kennell, D. S. Ebert, and G. Klimeck. Hub-based simulation and graphics hardware accelerated visualization for nanotechnology applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1061–1068, 2006.

[24] C. Rezk-Salama and A. Kolb. A Vertex Program for Efficient Box-Plane Intersection. In *Proceedings of Vision, Modeling and Visualization (VMV)*, pages 115–122, 2005.

[25] S. Röttger, S. Guthe, D. Weiskopf, T. Ertl, and W. Straßer. Smart hardware-accelerated volume rendering. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 231–238, 2003.

[26] G. Schaftenaar and J. Noordik. Molden: a pre- and post-processing program for molecular and electronic structures. *Journal of Computer-Aided Molecular Design*, 14(2):123–134, 2000.

[27] J. Schmidt-Ehrenberg, D. Baum, and H. C. Hege. Visualizing dynamic molecular conformations. In *Proceedings of the conference on IEEE Visualization 2002*, pages 235–242, 2002.

[28] J. E. Stone, J. Saam, D. J. Hardy, K. L. Vandivort, W.-m. W. Hwu, and K. Schulten. High performance computation and interactive display of molecular orbitals on gpus and multi-core cpus. In *IACAT, GPGPU-2: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pages 9–18. ACM, 2009.

[29] N. Svakhine, Y. Jang, D. S. Ebert, and K. P. Gaither. Illustration and photography inspired visualization of flows and volumes. In *Proceedings of the conference on IEEE Visualization 2005*, pages 687–694, 2005.

[30] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing to enhance real time molecular visualization. *IEEE Transaction on Visualization and Computer Graphics*, 12(5):1237–1244, 2006.

[31] I. S. Ufimtsev and T. J. Martinez. Quantum chemistry on graphical processing units. 1. strategies for two-electron integral evaluation. *Journal of Chemical Theory and Computation*, 4(2):222–231, February 2008.

[32] O. Wilson, A. V. Gelder, and J. Wilhelms. Direct volume rendering via 3D textures. Technical Report UCSC-CRL-94-19, Santa Cruz, CA, USA, 1994.