J. Comput. Chem. Jpn., Vol. 1, No. 4, pp. 135–142 (2002)

分子骨格操作に伴う分子軌道変化のリアルタイムボリュームレンダリング

中 貴俊[a]*, 山本 茂義[b], 秦野 やす世[a], 山田 雅之[a], 宮崎 慎也[a]

[a] 中京大学情報科学部, 〒 470-0393 愛知県豊田市貝津町床立 101
[b] 中京大学教養部, 〒 466-8666 愛知県名古屋市昭和区八事本町 101-2
**e-mail: h10205m@media.sccs.chukyo-u.ac.jp*

(Received: March 29, 2002; Accepted for publication: November 1, 2002; Published on Web: December 13, 2002)

本研究では，多原子分子内において分子骨格中の特定の原子をある経路に沿って移動させることによって生じる分子軌道の変化を，雲状オブジェクトとしてリアルタイムで表示する方法を提案する．移動原子が移動経路上の任意の位置にあるときの状態を表示するためには，移動経路上の有限個の離散点における軌道のデータをあらかじめ求めておき，それ以外の位置における軌道は離散点で求めておいた軌道データから補間により生成する必要がある．この軌道の補間をレンダリングにおける混合処理のみで行うことにより，近隣２つの軌道データからの補間値を計算することなく，分子軌道の変化をリアルタイムで観察することを可能としている．

キーワード : 分子軌道, 電子密度雲, サイエンティフィック・ビジュアリゼーション, ボリュームレンダリング, リアルタイムＣＧ

1　はじめに

分子軌道は電子の挙動を記述するための基本となるものであり，分子軌道の形状を把握することは，反応のメカニズムを解明するうえで重要である．しかしながら，最も構造が簡単な水素原子でさえ，量子数が大きくなると軌道の関数式から電子のふるまいをイメージすることは困難となる．例えば原子軌道や比較的簡単な分子軌道の図形表現については，断面の等高線や鳥瞰図による表現，等値面のワイヤーフレームによる３Ｄ表示等が量子化学の教科書や専門書で従来から用いられている [1–3]．これらは原子分子の電子状態や特徴を把握，理解するうえで大きな助けとなっているが，ある断面での２Ｄ表現や等値面表示は基本的に電子密度情報の一部しか表現できないため，軌道全体を理解するためには複数の画像が必要となる．また，軌道の特徴がよく表現された画像を得るためには，適切な断面や等値面を選ぶ必要があり，経験的な知識や試行錯誤が必要であった．

その後，急速に進歩したコンピュータグラフィックスの技術を用いてこれらの情報を３次元的に可視化しようという試みが，ここ十数年の間に盛んに行われてきた [4]．レンダリングの方法は等値面などの面情報のみを表示するサーフェスレンダリングから内部の情報を半透明表示するボリュームレンダリングへと発展した．その方法も当時の主流であったレイトレーシング法，レイキャスティング法 [5] などの，基本的には静止画を数値計算で得るタイプの方法から３Ｄグラフィックスのハードウェアによるリアルタイムの混合処理を利用した方法 [6, 7] へと移行し，半透明表示された空間的な情報を，視点を変えながら自由に観察することが可能となった．現在では，それらの成果の一部はすでに Gauss View[8] や Chem3D[9]，Win MOPAC[10], micro AVS[11] といった市販の製品の機能として提供されており，身近に利用できる環境も整いつつある．それらは基本的にポリゴンモデルのレンダリングに基

づいており，分子軌道の等値面を半透明のサーフェースモデルとしてレンダリングすることにより内部の分子骨格を透過表示したものや，複数の等値面の多重表示したものが存在する．

他方，分子軌道の関数値分布全体を把握するにはむしろ分布状態を雲状に表示する方が有効である．このような例としては文献 [5] が挙げられるが，これはレイキャスティング法に基づいたものであり，リアルタイム性は考慮されていない．これに対して，最近のリアルタイム手法を用いれば，分子骨格を故意に変形させて，その際の分子軌道の変化等を観察することが可能となる．このような現象をリアルタイムでシミュレーションできる環境が整えば，それは化学者にとって様々な化学反応の予測や検証をする上で有効な手段となる．

そこで本研究では，多原子分子内において分子骨格中の特定の原子をある経路に沿って移動させることによって生じる分子軌道の変化を，雲状オブジェクトとしてリアルタイムで表示する方法を提案する．分子軌道データは，原子の配置が変化すると再計算する必要があるため，移動原子が移動経路上の任意の位置にあるときの状態を表示するためには，移動経路上の有限個の離散点において軌道データをあらかじめ求めておき，それ以外の位置における軌道は離散点で求めておいた軌道データのうちの近隣２つから補間により生成する必要がある．本研究ではこの補間状態のレンダリング結果を OpenGL ライブラリの API 関数のうち基本的なものの組み合わせのみによって生成する方法を提案している．これにより，近隣２つの軌道データからの補間値を計算により求める必要がなく，その結果として原子の移動に伴う分子軌道の変化をリアルタイムで観察することを可能としている．

2 ボリュームレンダリングの基本方法

分子軌道は，空間中の関数として定義される量であるが，実際には，数値シミュレーションにより空間中の格子点におけるサンプル値（ボクセルデータ）を求めることになる．ここではボクセルデータは分子軌道計算ソフト Gaussian 98[12] により作成し，cube フォーマットの形式で出力したものを用いる．

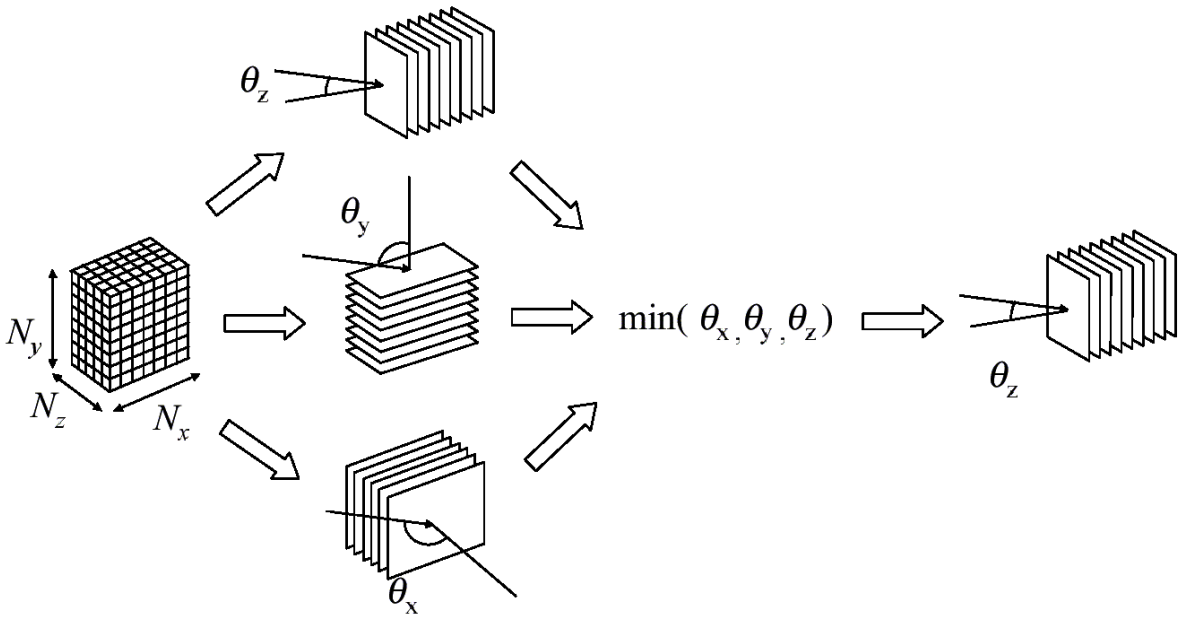


Figure 1. Selection among three directional piled texture images in the rendering process. Only one piled texture image is selected in the rendering procedure. The selected one has the smallest angle between the normal vector face and the view-line vector.

透過率をもつボリュームデータのレンダリングにおいては各ピクセルの輝度値を積算するレイキャスティング法（ray casting method）が基本となるが，この方法では各ピクセルにおける輝度値の積算に時間がかかるためリアルタイム処理が困難となる．これに対し，リアルタイム CG の技術を利用したボリュームレンダリングの方法として，透過率をピクセル値としてもつ半透明のテクスチャをポリゴンにマッピングし，それを多重に配置することにより空間分布データを雲状にレンダリングする方法が確立されている [6, 7]．後者の方法ではプログラムの実行開始時に多数のボリュームデータをファイルから読み込み，OpenGL のテクスチャマッピング用の API に渡す処理にはある程度の時間を要するが，その後のレンダリングはリアルタイムで実行可能である．また，レイキャスティング法と比べて描画結果の厳密性には欠けるが，それに近い描画結果を高速に得ることができる．本研究でもリアルタイム性を前提としているためこの方法を用いる．

なお，開発は Microsoft 社の Visual C++上で用い，グラフィックスライブラリは OpenGL[13]，GUI ライブラリは GLUT[14] を用いた．

2.1 透過率をもつボリュームデータの作成

軌道関数は原子分子中の個々の電子座標を変数とする 1 電子関数であり，この関数の組合せによって原子分子の電子構造が表現される．軌道関数値の絶対値の高い部分を雲が濃い，すなわち光が透過しにくい，また値の低い部分を雲が薄い，すなわち光が透過しやすい，とすることにより軌道関数値の分布を雲として表現することができる．軌道関数は正負の値をとりうるが，ここでは正負の違いは色により区別する．

したがって，基本的には軌道関数値の絶対値を透過レンダリングにおける不透明度とすればよいが，雲の濃淡を調節して良好な結果を得る必要がある．ボリュームデータとして与えられる立方格子上の離散点 (x, y, z) の軌道関数値を $\gamma(x, y, z)$ として，その位置での不透明度 $\alpha(x, y, z)$ を以下の式で与える．

$$\alpha(x,y,z) = k_1 \cdot \gamma(x,y,z)^{k_2} \quad （k_1, k_2 は定数） \qquad (1)$$

ここで，k_1，k_2 は調節パラメータであり，それぞれ係数調節と指数調節を示す

2.2 テクスチャ画像の生成

ボリュームレンダリングにおけるテクスチャ画像生成の概要を Figure 1 に示す．ボクセルデータのサイズが $N_x \times N_y \times N_z$ であり，例えば xy 平面に平行な同一平面上のボクセルをテクスチャ画像とする場合，テクスチャ画像数は N_z となり，テクスチャ画像 i の各ピクセルのアルファ値は

$$\alpha_i(x,y) = \alpha(x,y,i) \quad (i = 1, 2, \ldots, N_2) \qquad (2)$$

となる．これらのテクスチャ画像はプログラムの実行開始時に生成され，レンダリング準備のための OpenGL API に渡される．レンダリング時には視点から最も遠い面から順に混合処理が適用されながら描画される．このレンダリング処理はグラフィックスハードウェアによって行われるため高速に処理される．

ただし，視線ベクトルとテクスチャ画像が平行に近くなるにつれて描画結果が不適当になるので，yz 平面，zx 平面に平行な方向ついても同様にテクスチャ画像群を生成しておく．レンダリング時には，これら 3 方向のうちから視線ベクトルと画像面の法線ベクトルとのなす角が最も小さいものが選択され描画に用いられる (Figure 1).

３ D テクスチャリングの技法を利用すれば，多重プレーンを視線に平行に設置し，より良い効果を得ることが知られている．しかしながら，今回のように対象物体が雲状の場合は，その恩恵は少なく，グラフィックスチップが３ D テクスチャをハードウェアでサポートしていることが前提となる．そこで今回は現存する多くのグラフィックスハードウェアでリアルタイム処理が可能な２ D テクスチャマッピングのみで実現する方法を用いた．

2.3 分子骨格の描画

各原子の中心位置には原子の種類に応じて色分けされた球を描画する．原子間をつなぐ骨格については，両端の原子の原子半径に基づいて，その原子間に骨格を描画すべきであるかどうかを前もって判定しておき，円柱として描画する．

3 補間レンダリング

分子骨格が変化すれば分子軌道は再計算する必要があるが，用意できる軌道データの数は有限である．そ

こで，分子骨格の動きを単一のパス上に限定し，そのパス上の代表となる離散点のみであらかじめ軌道データを作成しておく．軌道データが存在する離散点以外の点においては，近隣２つの軌道データを混合した状態をレンダリング処理により生成する (Figure 2).

3.1 近似的解法

補間位置における軌道データは近隣の離散点における軌道データを内分することによって得られるが，この処理はボクセル数に比例した計算処理となるのでリアルタイム処理は難しい．そこで，以下のようなOpenGL の混合処理に関する API を組み合わせて近似的に２ボクセルデータの間の内分結果をレンダリングにより生成する．

今，キーとなるボクセルデータ A および B の内分比が 1-k : k で，データ A およびデータ B の第 i 画像のアルファ値をそれぞれ，α_{Ai}, α_{Bi} とすると，アルファ値の内分値 α_i は以下の式 (3) となり，画像 i までの積算濃度値 d_i は画像 i -1 までの積算濃度値 d_{i-1} と α_i を用いて以下の式 (4) で表される．

$$\alpha_i = (1-k)\alpha_{Ai} + k\alpha_{Bi} \quad (3)$$

$$d_i = d_{i-1}(1-\alpha_i) + \alpha_i \quad (4)$$

これをまとめると，

$$d_i = d_{i-1}(1-((1-k)\alpha_{Ai} + k\alpha_{Bi})) + (1-k)\alpha_{Ai} + k\alpha_{Bi} \quad (5)$$

となり，これを OpenGL の混合処理の組合せで実現することはできない．そこで，近隣のキーボリュームデータ間の対応するボクセル値は近い値であると仮定し，α_{Ai}= α_{Bi} とおけば，式 (5) は以下のように簡単化される．

$$d_i = d_{i-1}(1-\alpha_{Ai}) + (1-k)\alpha_{Ai} + k\alpha_{Bi} \quad (k < 0.5) \quad (6)$$

$$d_i = d_{i-1}(1-\alpha_{Bi}) + (1-k)\alpha_{Ai} + k\alpha_{Bi} \quad (k > 0.5) \quad (7)$$

これらの式によるレンダリング処理は，第 i -1 画像までの既積算濃度値 d_{i-1} に，まずデータ A（または B）のアルファ値を１から引いたものを乗算し，次にデータ A のアルファ値を 1-k 倍して加算し，最後にデータ B のアルファ値を k 倍して加算するものである．これは逐次的な３回の混合処理により実現できる (Figure 3).

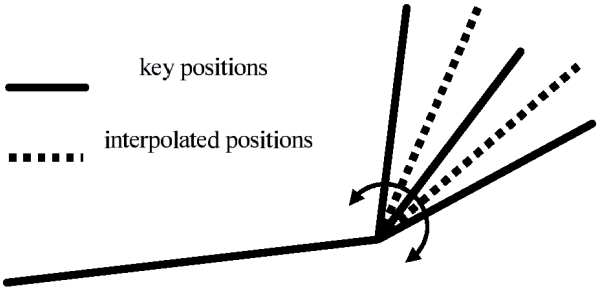


Figure 2. Interpolated position’s state is rendered by blending neighboring key position’s volume. The volume data on the key position are calculated beforehand.

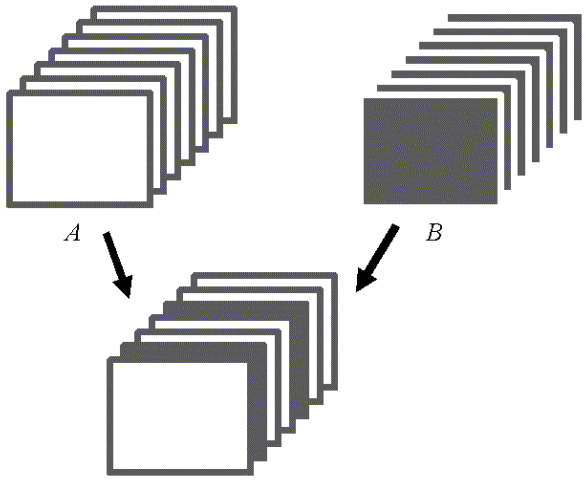


Figure 3. The interpolated state is generated by sequentially alternatively rendering two key position’s volume data sets.

3.2　適用例

ここでは，応用例としてレチナールのプロトン化シッフ塩基（protonated Schiff base of retinal，PSBR）を扱う．この分子は色素タンパク質バクテリオロドプシンの発色団である．バクテリオロドプシンは光エネルギーを水素イオン濃度勾配に変換する．568 nm の光によって励起された trans 体から，13-cis 体への異性化反応がそのサイクル反応の第１ステップである [15]．C_{13}=C_{14} 結合の２面角（θ）が-180°（trans）から-90°へと回転する過程で，第一励起状態（S_1）から基底状態（S_0）への遷移が起こると考えられている．反応機構については共著者 [16] による研究が既に発表されているが，本論文では分子軌道のリアルタイム表示が目的であり，簡単のために Hartree-Fock レベルでの分子軌道を表示している．S_1 は HOMO（π）から LUMO（π*）への遷移で表される電子状態であり，C_{13}=C_{14} 結合の回転に伴って HOMO と LUMO がどのように変化するかを観察することによって反応機構の理解をより深めることができる．現実の反応では θ 以外の分子座標も変化してゆくが，ここでは１変数 θ に対する分子軌道の変化を表示している．

計算時間を短縮するため，レチナール分子のメチル側鎖は水素原子で置き換え，イオノン環は１個の二重結合で置き換えている（$C_{12}H_{16}N^+$）．基底関数は 6-31g である．Gaussian 98 プログラム [12] の cube オプションを用いて，-180° から 0° までの 15 個の θ について，あらかじめ計算してある．計算した角度は，-180°, -150°, -120°, -110°, -105°, -100°, -95°, -90°, -85°, -80°, -75°, -70°, -60°, -30°, 0° である．

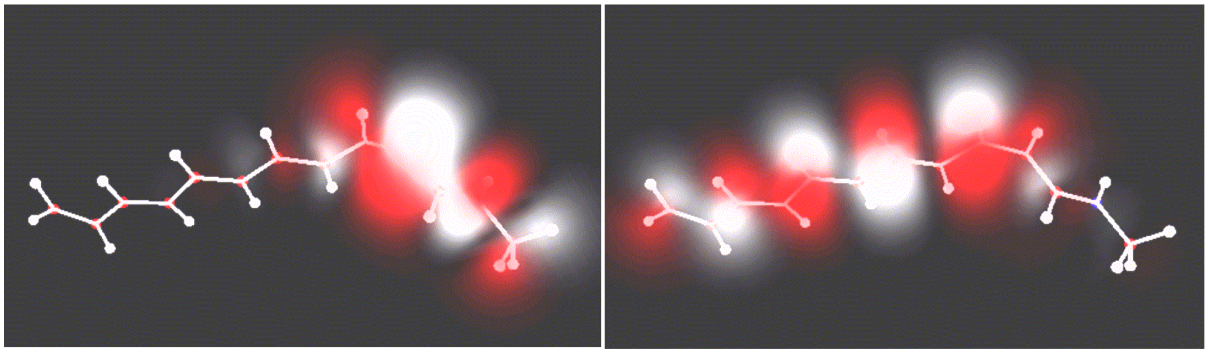
これら以外の角度については前節で述べた方法で補間して表示する．trans（-180°）では，分子はほぼ平面状である．回転にともなって分子軌道の形状は複雑になる．-90° 付近で分子軌道が大きく変化するため，この付近では θ の間隔を若干密にとって計算してある．全体で 9×15MB の容量であった．

Figure 4 は-90° での分子軌道を表示している．赤色，白色の雲がそれぞれ分子軌道の正負を表している．左側に HOMO，右側に LUMO を θ に連動させて表示している．HOMO は向かって右側（窒素原子のある側），LUMO は左側に局在化していることが分かる．マウスの水平方向のドラッグにより θ を変えるようになっている．表示物体の回転や拡大縮小といった観察視野の変更はキーボードのキー操作で行えるようになっている．マウス操作に追随して，遅滞なく円滑に描画されることが確認できた．

4　リアルタイム性の検証実験

リアルタイム性への影響が大きい要素は，分子軌道データ自体の離散化の解像度と，原子の移動パス上の離散点の数，すなわち移動パスの離散化解像度である．

今回用いた軌道データのボクセルサイズは 82×71×126 としたが，表示オブジェクトが雲状であること，現状では透過率の表現が８ビットであることから，この程度の解像度で良好な描画結果を得ており，ボリュームデータ自体の離散化解像度についてリアルタイム性に問題はないといえる．



HOMO　　LUMO

Figure 4. Molecular orbitals of PSBR at θ = -90°

これに対して，原子の移動パスの離散化解像度は，特に電子が軌道間を遷移する瞬間には軌道関数値の分布状態が大きく変化するため，離散点の間隔を細かくとる意味がある．また，移動パスの範囲を大きくし，原子の移動範囲の自由度を上げるためにも，離散点の数を増やす必要がある． これらのためには，プログラムの実行時に大量の軌道ボリュームデータをテクスチャデータとして保持する必要がある．そこで，本章では現存する標準的なグラフィックスチップにおいて，ボリュームデータの数に対するリアルタイム性について調査した結果を示す．本研究で提案しているレンダリング方法では，大量のボリュームデータを保持するものの，１回のレンダリングで使われるのはそのうちの２ボリューム分のみであるという特殊なケースとなる．したがって本章で述べる実験結果は一般的な描画性能のベンチマークとは異なり，本レンダリング方法に特化したパフォーマンス計測の結果であるという意味で意義がある．

4.1 ボリュームデータの保持に必要なメインメモリ量

OpenGL をはじめとするリアルタイム CG の API においては，テクスチャ画像は縦横のピクセル数が２のべき乗となるように再サンプリングされる．ここでは，そのサイズを 128×128 とし，最大の場合としてフルカラーの透過画像（４チャンネル）を用いた場合，K 個のキーボリュームデータのために必要となるメモリ量は次の式で定められる．

$$128^2 \times (N_x + N_y + N_z) \times 4 \times K \quad \text{(byte)}$$

例えば 3.2 章のボクセルサイズの場合にはおよそ 18.3 $\times K$（MB）となり，ボリュームデータ数が 15 なので，およそ 275 MB となる．

4.2 計測実験

２種類のハードウェア構成について描画時間の計測実験を行った．

(1) CPU: Pentium III 600 MHz
メモリ帯域幅: 1.06GB/sec (PC133)
Graphics chip: nVidia GeForce2 MX (32MB)

(2) CPU: Athlon XP 1900+ (1.6GHz)
メモリ帯域幅: 2.1GB/sec (PC2100)
Graphics chip: nVidia GeForce4 MX440 (64MB)

(1)，(2) はともに一般向けの廉価版のグラフィックスチップである．(1) の構成は旧世代のものであるが，現在すでに幅広く普及している標準的なものである．(2) の構成は最近の標準的なものであり，(1) と比較して CPU クロック，メモリ帯域幅がともに２倍程度に向上している．GeForce4 MX は廉価版のチップの中で最も新しい部類に入る．

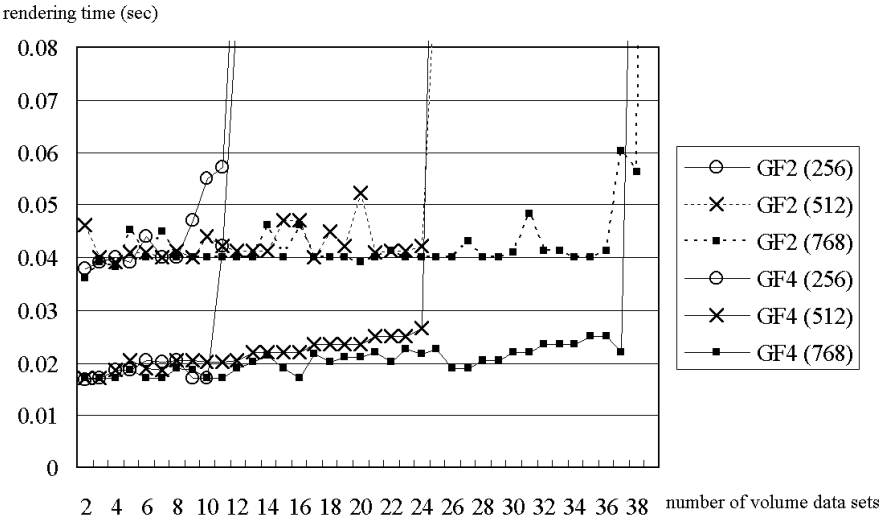


Figure 5. Rendering time depending on the number of volume data sets.

Figure 5 は，上記の 2 種類の各構成について，メインメモリの容量が 256MB，512MB，768MB の 3 種類の場合について描画時間を計測した結果である．横軸はボリュームデータの数を表している．メインメモリとグラフィックスボード上のビデオメモリとの間でテクスチャデータのスワップが発生すると一時的な描画速度の低下が発生する。一時的であっても描画速度の低下はリアルタイム性の低下を著しく体感させる要因となるので，ここではこうしたスワップ時を含めた計測値のうちの最大値を計測結果としている．

メインメモリの量を増やすことによりリアルタイムで描画できるボリューム数が増加することからグラフィックスチップのビデオメモリではなく，メインメモリの量がリアルタイム性のボトルネックとなっていることがわかる．これにより各場合においてリアルタイム応答を確保できるボリューム数に上限が存在することがわかるが，その範囲内で (1) では秒間 20 フレーム，(2) では秒間 30 フレームでの描画が可能となっている．例えば 3.2 章で実現した例では 275MB のメモリを必要とするので，512MB でリアルタイム性が十分確保できる．現在の標準的な PC で，充分な性能を出せることが検証された．

今回実現したのは骨格の一部を回転させるという原子の 1 次元の経路上のみの移動であったが，更に多くのメモリを搭載することにより，より自由度の高い原子の移動を可能にすることができる．

5 むすび

本文ではボクセルデータとして与えられた分子軌道関数値の空間分布をリアルタイム CG により雲状オブジェクトとして表示し，分子骨格の一部を回転させる等の操作をリアルタイムで対話的に行える機能を実現した．本システムは，分子軌道計算結果の可視化解析ツールとして役立つことが期待できる．

本論文の英語表記の誤りを親切にご教授下されました Newbold 教授に心から感謝します．本研究の一部は文部省私立大学ハイテク・リサーチ・センター補助金による財政的支援を受けた．

参考文献

[1] 鐸木啓三, 菊池修, 電子の軌道, 共立出版 (1984).

[2] 神沼二真, 鈴木勇, 分子を描く, 啓学出版 (1988).

[3] T. Helgaker, P. Jorgensen, and J. Olsen, *Molecular Electronic-Structure Theory*, John Wiley & Sons (2000).

[4] 時田澄男, 木戸冬子, 杉山孝雄, 渡部智博, 時田那珂子, 東千秋, *J. Chem. Software*, **8**, 7-16 (2002).

[5] S. Handa, T. Takada, *Visual Computing: Integrating Computer Graphics with Computer Vision* (1992), pp.313-328.

[6] K. Akeley, *In Computer Graphics*, SIGGRAPH ’93, Anahiem, CA (1993).

[7] B. Cabral, N. Cam, and J. Foran, *Proceedings of the 1994 symposium on Volume visualization* (1994), pp.91-98.

[8] Gauss View : Gaussian,Inc.

[9] Chem3D : Cambridge Soft Corporation.

[10] WinMOPAC : Fujitsu Limited.

[11] microAVS : Advanced Visual Systems Inc.

[12] Gaussian 98（Revision A.5）, M. J. Frisch et al., Gaussian, Inc, 1998.

[13] J. Neider, T. Davis, M. Woo, *OpenGL Programming Guide*, Addison-Wesley (1993), p.478.

[14] http://reality.sgi.com/employees/mjk_asd/glut3/glut3.html

[15] T. Kobayashi, T. Saito and H. Ohtani, *Nature*, **414**, 531-534 (2001).

[16] S. Yamamoto, H. Wasada, and T. Kakitani, *Theochem*, **451**, 151-162 (1998).

Real-time Volume Rendering of Molecular Orbital Metamorphosis According to Molecular Frame Transformation

Takatoshi NAKA[a]*, Shigeyoshi YAMAMOTO[b], Yasuyo HATANO[a], Masashi YAMADA[a] and Shinya MIYAZAKI[a]

[a]School of Computer and Cognitive Sciences, Chukyo University
101 Tokodachi, Kaizu-cho, Toyota, Aichi, 470-0393 Japan
[b]Faculty of Liberal Arts, Chukyo University
101-2 Yagotohonmachi, Showa, Nagoya, Aichi, 466-8666 Japan
**e-mail: h10205m@media.sccs.chukyo-u.ac.jp*

This paper presents a novel way of visualization of a molecular orbital as a cloud rendered by real-time computer graphics. Manipulation of molecular structure, such as rotating a certain bond, is realized and the change of orbital is observed in real time. The values of orbital function are sampled and represented by voxel data in rendering. Data sets are constructed when a certain atom stays at some position along a moving path, called the key position. Interpolated position's state is also rendered by blending neighboring key position's volume. That process is performed only by using the blending function in the rendering.

Keywords: Molecular orbital, Electron density cloud, Scientific visualization, Volume rendering, Real-time computer graphics