ベクトル処理によるボリューム・レンダリングの高速化の研究

6Ｐ－4

武井　利文

日本電気㈱　情報処理システム技術本部　科学技術システム部

１．はじめに

最近、特に科学技術分野において、コンピュータによる可視化が注目されているが、そこではボリューム・データと呼ばれるものが扱われることが多くなっている。その典型的な例として、医療分野における三次元画像再構成、科学工学の分野における三次元数値シミュレーションの結果の可視化等が挙げられる。

このようなデータの可視化技術として最近登場したのが、ボリューム・レンダリングによる方法である[1)2)3)]。これは、全てのボリューム・データの寄与をデータ値に対応する色と透過率を考慮して画像に反映させる方法であり、従来からのＣＧの応用[4)]と違って、生成された一枚の画像を見るだけでボリューム・データの三次元的な変化の様子が理解できるという利点がある。しかしながら、この方法の最大の欠点は計算時間がかかることであり、視点の変更やアニメーション作成などの際に障害となっている。そこで今回、アルゴリズムをベクトル化してスーパーコンピュータで実行することにより、計算時間の短縮を実現した。本稿では、レイ・キャスティング法を用いたアルゴリズムのベクトル化の概要を述べると共に、実行結果についても紹介し、どの程度の高速化が図れるかについて一つの目安を与える。

２．ベクトル化アルゴリズム

効率の良いベクトル化を行うためには、ベクトル長を長くする必要がある。

レイ・キャスティング法を用いたアルゴリズムから直ちに考えられるのは、一つの画素の色を決定する過程をベクトル化する方法である（図１）。しかし、実際には、一つの視線上のサンプリング点数はベクトル・レジスタの大きさと同程度またはそれ以下である場合が多く、この方法はあまり効率的であるとは言えない。これに比べ、スクリーン上における画素数はずっと多い。

そこでここでは、全サンプリング点のうち、スクリーンに平行な面上に存在するサンプリング点を一まとまりとして考え、これらの点に対する計算をベクトル化する方法をとった（図２）。この方法では、一画面分の各画素の色（ＲＧＢ値）及び透過率等を格納しておく記憶領域をあらかじめ確保しておく必要があるが、記憶容量が足りない場合は画面をいくつかに分割して計算することが可能である。以下にアルゴリズムの概要を述べる（図３）。尚、ここでは等間隔又は非等間隔直交格子点上に定義されたボリューム・データを対象とする。

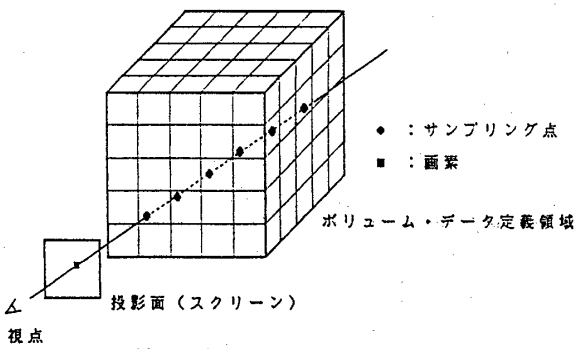

図１　ボリューム・レンダリング

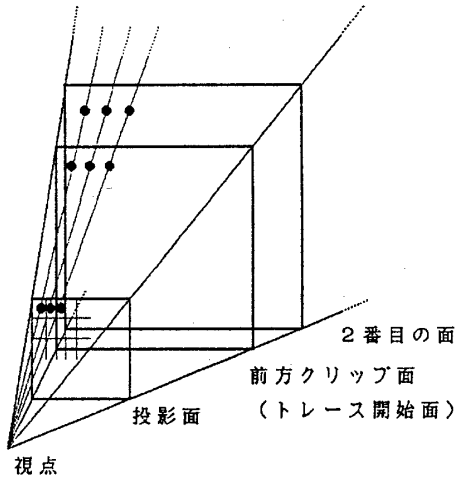

図２　ベクトル化ボリューム・レンダリングの概要図

２．１　ボリューム・レンダリングの準備

まず初めに、各画素に対応する色を０に、透過率を１に初期化すると同時に、トレース開始点、視線ベクトル、サンプリング点の間隔を求めておく（処理①）。この処理は、全画素を対象とするベクトル化が可能である。

２．２　サンプリング点リストの生成

処理②～④はボリューム・レンダリングの主要部分であり、スクリーンに平行な面ごとに、手前の面から奥に向かって実行していく。

ここでは、処理①で求めたトレース開始点、視線ベクトル及びサンプリング点の間隔から、スクリーンに平行なる面上のサンプリング点の座標値を求める。次に、これらの中から、ボリューム・データ定義領域内に存在し、かつ視点からの透過率が０でないものをリストアップし、リスト・ベクトルを生成する。この面と視点との間の透過率は、処理④で既に計算されているものとする。この処理は、面上の全サンプリング点を対象とするベクトル化が可能である。

次に、リストアップされたサンプリング点に対しては、その座標値からこの点を含む格子及びこの点におけるデータ値を求める。データ値の補間には三段線形補間法[2)]を用いる。等間隔直交格子の場合、この処理は前述のリスト・ベクトルを対象とするベクトル化が可能である。一方非等間隔直交格子の場合、サンプリング点を含む格子の算出はサーチ処理となり、これが内部的にループを含むため、ベクトル化されない。

２．３　データ値の勾配方向の算出

サンプリング点においてライティング計算を行う場合、それに必要な法線ベクトルをデータ値の勾配方向にとる。処理③では、まず各サンプリング点を含む格子の８つの頂点上でデータ値の勾配方向を求める。格子点におけるデータ値の勾配方向は、次の式で求められる。

$$(\nabla x\ D)_{x=xi} = \{(D)_{x=xi+1} - (D)_{x=xi-1}\} / (x_{i+1} - x_{i-1})$$

$$(\nabla y\ D)_{y=yi} = \{(D)_{y=yi+1} - (D)_{y=yi-1}\} / (y_{i+1} - y_{i-1})$$

$$(\nabla z\ D)_{z=zi} = \{(D)_{z=zi+1} - (D)_{z=zi-1}\} / (z_{i+1} - z_{i-1})$$

これから、三段線形補間法によりサンプリング点におけるデータ値の勾配方向を算出する。この処理はリスト・ベクトルを対象とするベクトル化が可能である。

Vectorized Volume Rendering

Toshifumi Takei

NEC Corporation

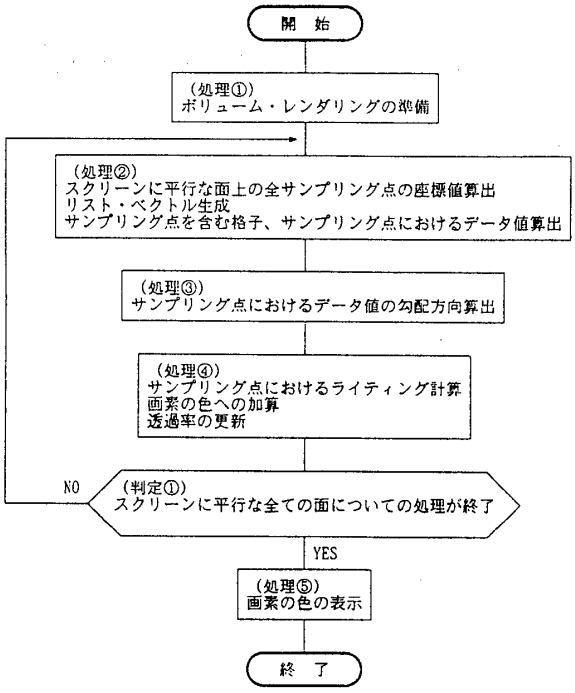

図３　処理の流れ

２．４　ライティング計算

処理④では、処理②でリストアップされたサンプリング点に対して、これらの点におけるデータ値に対応するボリュームの色、透過率等の属性と、処理③で求めた法線ベクトルを用いてライティング計算を行う。この計算にはフォンの式[5]を用い、計算は各光源（環境光源、平行光源、点光源、スポット光源）ごとに行う。

ライティング計算により求められた各サンプリング点での色は、この点でのボリュームの透過率及びこの点と視点との間の透過率を考慮して、次のように画素の色に加算する。

$$Ci \leftarrow Ci + Ii * (1-A) * \alpha \quad (i=R, G, B)$$

IR，IG，IB：サンプリング点でのシェーディングされた色
CR，CG，CB：画素の色
A：サンプリング点におけるボリュームの透過率
α：サンプリング点と視点との間の透過率（画素の透過率）

最後に、画素の透過率に、対応する各サンプリング点におけるボリュームの透過率とサンプリング点の間隔を掛け、新たに画素の透過率とする。即ち、

$$\alpha \leftarrow \alpha * A * LEN$$

LEN：サンプリング点の間隔

この処理は、各光源ごとにリスト・ベクトルを対象とするベクトル化が可能である。

２．５　画像の表示

処理⑤では、計算された画素の色をワークステーションに表示したりファイルに書き出したりする。

３．実行結果

図４には、シリコン結晶内の電子密度分布のボリューム・レンダリングによる表示を行った例を示す。このデータは、４８×４８×４８の等間隔直交格子上に定義されている。画像の解像度（画素数）は縦横共に２５０で、各画素ごとに奥行き方向のサンプリング点の数が７２となるような間隔でトレースしている。光源としては、環境光源と平行光源を使用している。

今回使用したのはＮＥＣのスーパーコンピュータ「ＳＸ－２Ａ」で、そのピーク性能は１．３ＧＦＬＯＰＳである。この計算を第２章で述べたアルゴリズムで実行したところ、ベクトル化率は９９％以上を達成した。因みにＣＰＵ時間は約８．５秒であったが、これはボリューム属性の設定のしかたで多少変化する。同じ計算を「ＳＸ－２Ａ」でスカラー処理した場合（マシン・サイクル：６ナノ秒）のＣＰＵ時間は約１分であった。従って、ベクトル処理を行うことにより、７倍程度の高速化が実現されていることがわかる。処理能力を表す指標として、１秒当たりに処理できるサンプリング点数を挙げることができるが、今回の場合は４０万サンプリング点／秒以上であった。ただ、ベクトル化率が高いわりには全体の処理効率があまり上がっていない。これはボリューム・データ値やボリューム属性等に対するランダム・アクセスが頻繁に発生するためであろう。

同じ計算を非等間隔直交格子として計算すると、処理②のコストが高くなると同時にその効率も大幅に悪くなる。全体のＣＰＵ時間は、今回の場合約２倍になった。この場合は、第２章で述べたサーチ処理をいかに行うかが重要になろう。

図４　表示例

４．おわりに

今回、ボリューム・レンダリングのアルゴリズムのベクトル化を行うことでどれだけ高速化できるかという問題に一つの回答を与えることができた。

最近のハードウェアの進歩は目覚ましく、現時点での最高速のスーパーコンピュータはＮＥＣの「ＳＸ－３」で、そのピーク性能は２２ＧＦＬＯＰＳである[6]。この「ＳＸ－３」をはじめとして、スーパーコンピュータの処理速度をさらに向上させるために、主記憶共有型マルチ・ベクトル・プロセッサ構成が主流となりつつある。今回述べたアルゴリズムによれば、このようなコンピュータの場合には、画面をいくつかに分割し、それぞれの部分を別々のプロセッサで並列に処理することが可能である。その際、分割数に応じてベクトル長が短くなるが、元々十分に長い（典型的には１万から１００万）ため、その影響は問題にならないと思われる。このような並列処理についても現在検討中である。

謝辞　シリコン結晶内の電子密度分布のデータを提供いただいた、日本電気技術情報システム開発㈱の栗本武氏に感謝いたします。また、日頃有益な議論をしていただく日本電気㈱システム技術本部の皆様に感謝いたします。

［参考文献］

1．R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," Computer Graphics, Vol. 22, No. 4, Aug. 1988, pp. 65-74.
2．M. Levoy, "Display of Surfaces from Volume Data," IEEE Computer Graphics and Applications, Vol. 8, No. 3, May 1988, pp. 29-37.
3．C. Upson and M. Keeler, "V-BUFFER: Visible Volume Rendering," Computer Graphics, Vol.22, No. 4, Aug. 1988, pp. 59-64.
4．W. E. Lorensen and H. E. Cline, "Marching Cubes : A High Resolution 3D Surface Construction Algorithm," Computer Graphics, Vol. 21, No. 4, July 1987, pp. 163-169.
5．B.-T. Phong, "Illumination for Computer Generated Pictures," Comm. ACM, Vol. 18, No. 6, June 1975, pp. 311-317.
6．浅見，桑原，"スーパーコンピュータ、マルチプロセッサ構成で２０ＧＦＬＯＰＳ時代に，" 日経エレクトロニクス，No. 473, May 1989, PP. 183-191.