グラフィクスとCAD 71−2
（1994. 10. 14）

周波数領域におけるボリュームレンダリングの高速なインプリメンテーション

横山琢，戸塚卓志

ソニー 中央研究所 システム・LSI 研究部門 橋本研究室

周波数領域におけるボリュームレンダリングは、ボリュームレンダリングの高速アルゴリズムである。このアルゴリズムは、空間領域のボリュームレンダリングに比べて計算量での優位性が明らかであるが、実用にあたっては、スライスを抜き出す際の補間と複数回の周波数逆変換という2つの高価な処理が問題となる。本稿ではこれらの問題について詳しく説明し、その解決法について述べる。

A Fast Implementation of The Frequency Domain Volume Rendering"

Taku Yokoyama, Takashi Totsuka

Hashimoto Signal Processing Laboratory, Systems & LSI Res.Div.,
Research Center, Sony Corporation

The frequency domain volume rendring is a fast algorithm for volume visualization. Although it is apparently superior to spatial domain algorithms in terms of the computational complexity, its implementation has to address two major problems — the cost of interpolation filtering during the slice extraction process and the cost of multiple inverse Fourier transform operations. The paper describes these practical aspects of the frequency domain volume rendering and shows how the problems can be solved.

1 まえがき

3 次元 CG のレンダリング技法の一つにボリュームレンダリング [1] がある。ボリュームレンダリングは 3 次元の離散データを可視化する手段であり、医用、工学、科学等に幅広く応用されている。

従来のボリュームレンダリングは、扱うデータ量が多いため表示速度が遅い問題があった。Frequency Domain Volume Rendering[2](以下、FDVR) は、ボリュームレンダリングの処理を周波数領域上で行なうことにより、高速な処理を可能にするアルゴリズムである。

従来のボリュームレンダリングと比較すれば、FDVR の計算量は $O(n^3)$ から $O(n^2 \log n)$ に減少し、しかもデータの性質、例えばボリュームデータ内の非 0 の値を持つデータの密度に依存しない。このように計算量では明らかに優位であるが、インプリメンテーションに注意を払わないと $n^2 \log n$ にかかる係数が大きくなり、n が充分大きくない場合には処理時間において大きな優位性が得られない。

その第 1 の理由は、周波数領域において、3 次元スペクトルを平面で切断して取りだす作業 (スライスの抜き出し) に必要となる、フィルタによる補間処理が高価であるためである。また、第 2 の理由として、カラー画像の生成には複数回の周波数逆変換が必要であり、そのコストが無視できないことが挙げられる。

本稿では、上記の問題に対するインプリメンテーションの工夫について述べる。

2. 周波数領域におけるボリュームレンダリング

最初に、FDVR のアルゴリズムの簡単な解説をする。

ボリュームレンダリングが投影方向の線積分値を求めるアルゴリズムであることから、信号処理の理論が応用できる。即ち、空間領域におけるボリュームの投影像は、周波数変換したボリュームから、原点を通り投影方向に垂直な断面を抜き出し、これに周波数逆変換を施した像と等価になる [3][4]。

図 1において左上が 3 次元のボリュームデータ、左下が求めたい 2 次元の投影画像である。このとき、(左上→左下) の過程が通常のボリュームレンダリングである。また、(左上→右上→右下→左下) が FDVR において用いる過程である。

FDVR の最初の過程 (左上→右上) において、空間領域のボリュームデータを周波数領域に変換する。この過程は前処理で 1 回だけ行なえばよい。以後、レンダリングを更新するときには周波数変換後のボリュームデータを用いることになる。

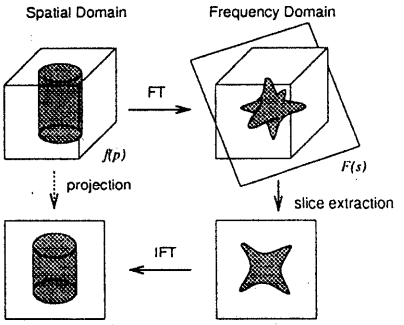

図 1: FDVR の概要図

レンダリングを更新するときには、このボリュームから原点を通り視線方向に垂直なスライスを抜き出し (処理 1)、周波数逆変換 (処理 2) を施せばよい。ここでスライス上のデータを算出するには、フィルタによる補間処理を施す。

このとき、処理 1 の計算量は $O(n^2)$ であり、処理 2 の計算量は FFT(高速フーリエ変換) を用いた場合に $O(n^2 \log n)$ である。従って、FDVR の全体の計算量は $O(n^2 \log n)$ になり、従来のボリュームレンダリングの計算量 $O(n^3)$ と比較して優位である。

更に FDVR では、視覚効果としてデプスキュー (視線方向へ明度の重みを付ける) とシェーディング (視線方向と垂直に平行光を照射する) を与えている。これらの視覚効果は全て周波数領域において処理されるため、周波数領域におけるボリュームは前処理において計算されたものを用いればよい。

ここで、周波数領域でシェーディングを施すには、ボリュームから抜き出したスライスの 1 次モーメントを計算する。また、光源が複数存在する場合には、光源数と同じ個数の 1 次モーメントの算出を行なう。

3. 高速なインプリメンテーション

前節で解説したように、FDVR には、スライス抜き出しと周波数逆変換の 2 つの処理が必要になるが、

それぞれの処理において高速化のために以下の問題を有している。

最初に、補間処理における問題について説明する。

FDVR において、3 次元スペクトルから原点を通るスライスを抜き出すには reconstruction フィルタによる補間処理が必要になる。ここで、3 次元のスペクトルの各軸と、そのスライス上に定める 2 次元の座標軸は当然ながら一致しない。そのため、精度の良い補間には小数以下の微細な距離に対応した密なフィルタ係数表が必要である。例えば我々のインプリメントではボクセル間を 256 に分割する精度で係数を求めているが、これを 3 次元の係数表に格納しようとすると $3 \times 3 \times 3$ 程度の小さなフィルタでさえ数 10MB のメモリを必要とし、現実的ではない。

そこで、フィルタが可分、すなわちフィルタ係数 $w(x, y, z)$ が $w_x(x) \times w_y(y) \times w_z(z)$ と表現できることを利用してそれぞれの軸方向の係数表だけを用意し、毎回乗算によって $w(x, y, z)$ を求めることになる。これによって記憶スペースは大幅に削減できるが、補間処理にフィルタ係数算出のコストが上乗せされる。そのため、フィルタ係数の算出回数の削減は重要になる。

補間処理に関わる別の問題として、FDVR では性能の良いフィルタが不可欠であることが挙げられる。空間領域では、補間フィルタの性能が悪くてもそれは消えずに残った高周波ノイズとなってあらわれるため致命的な問題にはならない。しかし、周波数領域でのフィルタでは、フィルタの性能が悪ければ空間領域におけるボリュームの複製が消えずに残ることになり画像に致命的なエイリアシングを引き起こす。このため、どうしてもフィルタサポートの大きなフィルタを使用することになり、補間処理は $O(n^2)$ でありながら $O(n^2 \log n)$ である FFT と同じ程度の時間を消費する。以上が、スライス抜きだし時の補間処理に関わる問題である。

次に、周波数逆変換に関する問題について説明する。前節で述べたように、周波数領域でシェーディングを施すには、光源数と同じ枚数のスライスの 1 次モーメントが必要である。これらをそのままそれぞれ周波数変換したのでは光源数に比例した処理時間が必要になる。

本節では以上の問題に対する工夫について解説する。

3.1 スライス抜き出しに関する高速化の工夫

3.1.1 対称性を利用したフィルタ係数の算出回数の減少

フィルタによる補間処理では、値を補間によって求めたい場所 (出力点) を中心として、フィルタサポートの範囲内に存在するサンプル (入力点) をフィルタ係数で重み付けして加算する。FDVR においては入力点は離散的な 3 次元スペクトルであり、出力点はスライス上の 2 次元離散スペクトルである。

ここで出力点、入力点、それにフィルタ係数に共通の対称性が存在する場合は、対称性が成り立つ座標の組合せだけフィルタ係数を共通に使用することができる。

FDVR においてはフィルタは対称なものを使用するので、図 2に示すように、スライス上の出力点を A1、A1 を中心とするフィルタサポート内の入力点を B1 とし、それぞれの原点対称点を A2、B2 とすれば、(A1,B1) の組合せと (A2,B2) の組合せにおいて使用するフィルタ係数を共有できる。従って、フィルタ係数の算出回数は $\frac{1}{2}$ で済む。

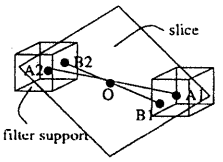

図 2: フィルタ係数の共有

一般に、n 次元の補間フィルタを作成する場合は、フィルタ係数が n 次元空間の各基底の交換に対して対称で、各基底に関して線対称であるように設計する。このとき、n 次元空間の各基底の交換に対して $n!$通りの対称性、n 次元空間の各基底に関する線対称に対して 2^n通りの対称性があり、これらを掛け合わせた $n! \times 2^n$通りの対称性が存在する。

従って、これらの対称性の中から、出力点と入力点に共通の対称性が m 通り存在する場合は、フィルタ係数の算出回数は $\frac{1}{m}$ に減少する。

3.1.2 境界検出によるフィルタ処理数の減少

出力点に補間フィルタを畳み込むとき、フィルタサポート内の入力点の値が全て0である場合は、出力点の値も0になるので計算は不用である。

従って、このような出力点の領域を予め算出しておけば、この領域内ではフィルタ処理を行なわずに、出力点の値を0にすればよい。

周波数領域でのボリュームデータ、すなわち3次元スペクトルは原点中心の立方体であるから、それを平面で切断して得られるスライスの形状は最大6頂点をもつポリゴンである(図3参照)。処理を簡単にするために実際にはそれを含む長方形の平面を考えてその内側で補間処理を行なうが、この際に3次元スペクトルの外側に相当する部分はフィルタ計算を行なわずに値を0にすることで補間の処理時間を短縮している。スライス面上の各点が3次元スペクトルの領域内かどうかの判定はポリゴンフィルのスキャンライン法を用いた。

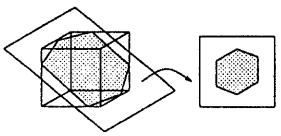

図3: スライス上でフィルタ処理が必要な領域

3.1.3 適応型フィルタリング

補間処理において小さいフィルタを用いれば積和演算の回数が減るから、当然処理時間は短くなる。従って、なるべく小さなフィルタを用いて補間を行ないたい。しかし、小さいフィルタを用いると出力精度は低下する。そのため、以下の2つの方法を考えることにする。

最初に、一定の出力精度を保証しながら適応的に小さいフィルタを用いる方法を説明する。この方法は[2]において既にインプリメントされた手法である。

3次元スペクトルの各点において、その周波数成分のパワーが小さければ空間領域に変換したときの影響が少ないので、補間処理に小さいフィルタを用いても画質は大きく変わらない。従って、周波数成分のパワーの小さい点は小さいフィルタを用い、パワーの大きい点は大きいフィルタを用いる(図4参照)。

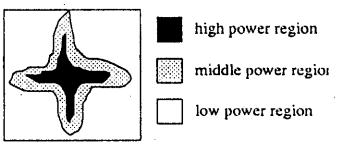

図4: 周波数変換のパワー

次に、要求される表示速度に応じてフィルタを適応的に交換する方法を説明する。この方法はFDVRへまだインプリメントされていない。

静止画において用いるフィルタを小さくすると、画像の精度の劣化として表れる。しかし、「高速に移動している物体の細部は認識できない」という人間の視覚特性より、動画において物体が高速に移動している場合は、画像の精度が劣化しても大きな問題にならない。従って、高速な表示が要求されるときは小さいフィルタを用い、低速な表示でもよいときは大きいフィルタを用いる。

3.2 周波数逆変換に関する高速化の工夫

3.2.1 演算順序の交換による周波数逆変換の回数の減少

FDVRにおいて環境光による効果は以下の処理により与えられる。即ち、3次元スペクトルから抜き出したスライスに周波数逆変換を施し、環境光の色係数を乗算してRGBの3枚のスライスを作る(図5参照)。

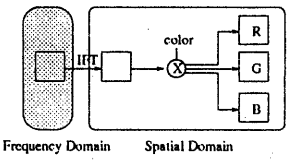

図5: 環境光の処理

また、平行光による効果は以下の処理により与えられる。まず、3次元スペクトルから抜き出したスライスの1次モーメントを計算する。このスライスに周波数逆変換を施し、平行光の色係数を乗算してRGBの3枚のスライスを作る(図6参照)。

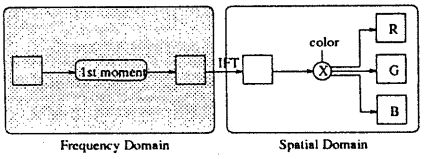

図 6: 平行光の処理

ここで、光源による効果を、(1) 環境光のみを与える場合、(2) 環境光と 1 個の平行光を与える場合、(3) 環境光と n 個の平行光を与える場合、の 3 通りに分けて考える。

(1) の場合は、環境光に関して図 5の処理を施せばよい。

(2) の場合は、環境光に関して図 5の処理を施し、平行光に関して図 6の処理を施し、算出された (3+3) 個のスライスを RGB 成分ごとに加算すればよい (図 7参照)。

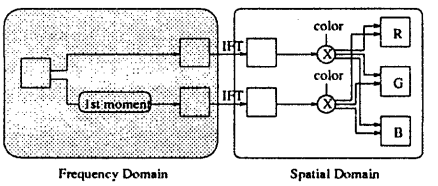

図 7: 環境光と平行光の処理

(3) の場合は、環境光に関して図 5の処理を施し、平行光に関して図 6の処理を平行光の数 n だけ施し、算出された $3(n+1)$ 個のスライスを RGB 成分ごとに加算すればよい (図 8参照)。

この場合の処理時間を計算する。スライスの 1 次モーメントを算出する時間を T_1、スライスに周波数逆変換を施す時間を T_2、スライスをスカラー倍する時間を T_3、2 枚のスライスを加算する時間を T_4 とすると、このときの処理時間 W_1 は、

$$W_1 = nT_1 + (n+1)T_2 + 3(n+1)T_3 + 3nT_4 \quad (1)$$

となる。

式 (1) において、周波数逆変換に要する時間 T_2 は、T_1, T_3, T_4 に比べて非常に大きく、そのコストが無視

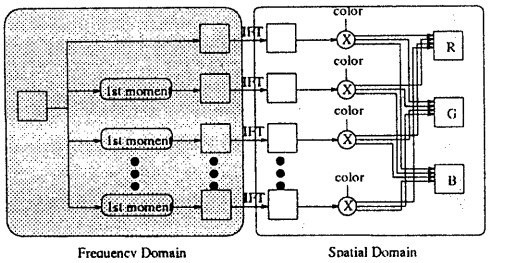

図 8: 環境光と複数の平行光の処理 (1)

できない。従って、T_2 の係数を減らすことが出来れば処理時間は大幅に短縮する。

環境光と n 個の平行光に対応した各スライスにおいて、それぞれの色係数を乗算して $3(n+1)$ 個のスライスを作り、これを RGB 成分ごとに加算する演算は線形演算である。また、周波数逆変換も線形演算である。

従って、これら 2 つの演算順序を交換することが出来る。即ち、各光源に対応するスライスの色成分を算出して RGB 成分ごとに加算する演算を、周波数変換の前に行なう (図 9参照)。

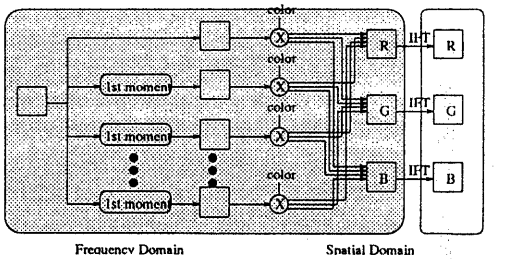

図 9: 環境光と複数の平行光の処理 (2)

このときの処理時間を W_2 とすると、

$$W_2 = nT_1 + 3T_2 + 3(n+1)T_3 + 3nT_4 \quad (2)$$

となる。式 (1) と式 (2) の差分は、

$$W_1 - W_2 = (n-2)T_2 \quad (3)$$

となるので、平行光の個数 n が 3 個以上の場合は、演算順序を交換したほうが高速になることが分かる。

4. むすび

FDVRにおいて充分な高速化を実現するには、(1)スライスを抜き出す際の補間処理、(2)カラー画像生成の際の複数回の周波数逆変換、が高価であることが問題となっていた。

しかし、補間処理や演算順序に関するアルゴリズムを改良することで、充分に高速な結果を得ることが出来た。

表1に、従来の空間領域でのボリュームレンダリング、周波数領域でのボリュームレンダリング、改良後の周波数領域でのボリュームレンダリング、を用いた場合の処理時間を示す。ここで用いたボリュームのデータサイズは128^3、マシンはIRIS Crimson(SGI)、CPUはR4000である。

表1: ボリュームレンダリングの処理時間

処理領域	時間(sec)
空間領域	3.15
周波数領域(改良前)	0.54
周波数領域(改良後)	0.37

表1より、本稿で紹介したインプリメンテーションにより、空間領域のボリュームレンダリングの11%、周波数領域のボリュームレンダリングの68%、に処理時間が短縮したことが分かる。

一般に、処理を高速にするには画像の精度を犠牲にすることが多いが、FDVRのアルゴリズムの改良では画質の精度を損なわずに処理時間を短縮できた。

FDVRに適切なインプリメンテーションを施すことで、空間領域のボリュームレンダリングに対して、ハードの性能に頼らずにソフトで処理時間を1桁程度まで短縮できた。これにより、ボリュームレンダリングの用途は一層広がるであろう。

文献

[1]R.A.Drevin, L.Carpenter and P.Hanrahan, "Volume Rendering", Computer Graphics, Vol.22, No.4, pp.65-74, July 1988.
[2]T.Totsuka, M.Levoy, "Frequency Domian Volume Rendering", Computer Graphics Annual Conference series, pp.271-278, 1993
[3]T.Malzbender, "Fourier Volume Rendering", ACM Transactions on Graphics, Vol.12, No.3, pp.233-250, July 1993.
[4]S.Napel, S.Dunne, and B.Rutt, "Fourier Projection for MR Angoiography", Magnetic Resonance in Medicine, Vol.19, pp.393-405, 1991.