4AD－8

時間的に変化するボリュームデータの高速レンダリング手法

平井 哲　山本 強
北海道大学工学部　北海道大学大型計算機センター

1　はじめに

ボリュームレンダリングは計算量が多いため、実時間処理が困難であった。本研究ではテクスチャーマッピングを応用した高速ボリュームレンダリングアルゴリズムを提案し、その性能評価を行った。提案手法は時間的に変化する透明度を持つボリュームデータをリアルタイムでレンダリングすることを目的とするものであり、可視化のみならずビデオゲームなどの新しい分野への応用を目指すものである。

2　従来手法とその問題点

従来手法として、ボリュームデータの各スライスデータをテクスチャーとして四辺形ポリゴンに貼付けてアルファブレンディングを使用してレンダリングを行う手法 [1] が提案されている。しかし、この手法の場合、occlusion により視点から見えないボリュームデータもテクスチャーとして処理し、アルファブレンディングを行う。したがって、レンダリングの効率が悪い。

3　提案手法

そこで、我々は以上の問題点を解決するために、本提案手法を開発した。この手法のおおまかな手順は以下の通りである。

1. **Occlusion Checking**　ボリューム空間内の各サブボリューム（部分領域）が最終画像に寄与するかどうかを調べる。
2. **可視可能なサブボリュームのレンダリング**　最終画像に寄与するサブボリュームのみのレンダリングを行う。
3. **次のフレームのサブボリューム分割解像度の決定**　2. でレンダリングされた画像のレンダリング時間をその前のフレームでレンダリングされた画像のレンダリング時間と比べ、次に使うサブボリュームの分割解像度を決める。

以上の三つのステップをレンダリングされる各フレーム毎、実行する。以下でこの三つのステップをさらに詳しく説明する。

4　Occlusion Checking

各サブボリュームが最終画像に寄与するかどうかを調べるために最低アルファ値データ構造というものを使う。このデータ構造のいくつかの例が図 1に示されている。

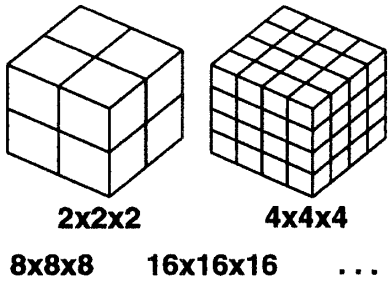

図 1: 最低アルファ値データ構造の例

このデータ構造の各部分領域はボリューム空間内のある部分領域に対応している。各部分領域は１つのアルファ値を持っていて、このアルファ値は、該当するサブボリュームの最低アルファ値である。この最低アルファ値データ構造は複数個存在していて、各データ構造は特定な解像度を持つ（図 1参照）。各フレームのレンダリングを行う度にこの中の一つの解像度を選び、occlusion check を行う。Occlusion check 自身は、各サブボリュームの６面の内、視点に近い３面を低解像度のアルファ値バッファーに投影することにより行う（図 2参照）。これにより、どのサブボリュームが可視可能かが決まる。

A Method for High Speed Rendering of Time Varying Volume Data
Tetu Hirai, Tsuyoshi Yamamoto
Hokkaido University

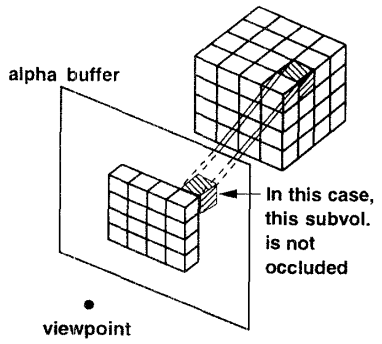

図 2: 可視可能なサブボリュームの検出

5 可視可能なサブボリュームのレンダリング

Occlusion check で可視可能と決定されたサブボリュームのレンダリングをアルファブレンディングをしながら行う。

6 次のフレームのサブボリューム分割解像度の決定

レンダリングされた画像に要した時間を、前のフレームのレンダリング時間と比べてみる。この比較により、次のフレームで使う最低アルファ値データ構造の解像度を決定し、レンダリング時間が最小になる解像度に近づける。

7 計算量の比較

我々は両手法の計算量を平均的なデータを想定して、見積もってみた。計算を行うための仮定として、1. レンダリングされる各四辺形ポリゴンは 256 のピクセルからなる、2. occlusion check 時にスキャンコンバージョンされる各四辺形ポリゴンは 16 のピクセルからなる、3. occlusion check の結果として、512 サブボリューム中 32 サブボリュームがレンダリング対象であると判定されたことが挙げられる。この見積もりにより、従来手法が 11,700,000 の実数演算と 3,100,000 の整数演算を要するが、提案手法は 1,410,000 の実数演算と 350,000 の整数演算を必要とすると分かった。つまり提案手法は約 1/8 の計算量である。なお、この計算はメモリーのアクセス時間等の他の要素は考慮していない。

8 従来手法の実験結果

現在、従来手法の実装のみが終わっているのでそのレンダリング結果の例を図 3 で示す。現在、128^3 のボリュームデータを SGI O2(R5000,180MHz) 上で 256^2 の解像度で 3.46 frames/sec (0.29 sec/frame) でレンダリングしている。

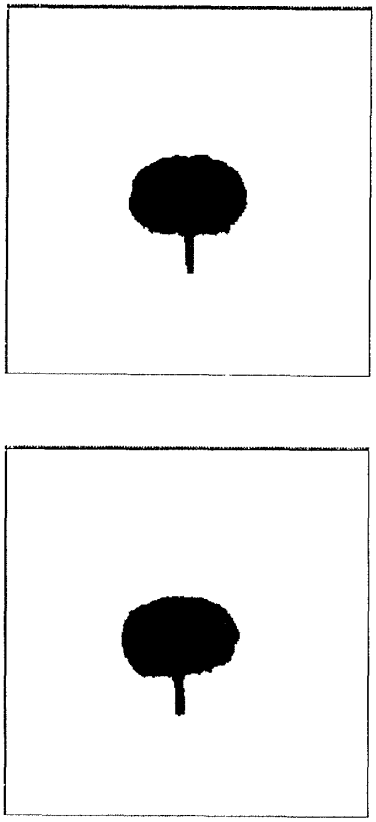

図 3: 従来手法でレンダリングされた画像の例

9 おわりに

計算量の比較により提案手法は従来手法の約 1/8 の計算量でレンダリングができることが分かった。これから、提案手法を実装して、レンダリング時間の比較を行う予定である。

参考文献

[1] P. Haeberli, M. Segal, "Texture Mapping as a Fundamental Drawing Primitive," Proc. Fourth Eurographics Workshop on Rendering, Paris, France, June, 1993. pp. 259-266.