社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－CG－124 (3)
2006／8／17

不透明物体をとりまくボリュームデータのGPUを用いた高画質レンダリング

今給黎 隆[†] ヘンリー・ジョハン[††]
田村 尚希[†††] 西田 友是[†††]

GPUを用いてボリュームデータ（煙や霞など）をレンダリングする手法として，ボリュームデータをスライスしたサンプル面によってレンダリングする手法が知られている．しかし，この方法では，ボリュームデータ内に不透明物体を設置した場合に交差する境界領域等でエイリアシング（縞模様を伴う誤差など）が発生する．本研究では，GPUを用いてボリュームデータのサンプリング領域を求め，その領域内でのみサンプル点をとることによって，不透明物体の境界付近においてエイリアシングが発生しない高画質かつ高速なボリュームレンダリングの手法を提案する．また，不透明物体が落とす影が生じるエイリアシングに関しても，シャドウマップを参照するとき，一つの位置の情報だけではなく，近傍の複数の位置での情報を補間することにより，エリアシングを取り除く．

High Quality Rendering of Scenes Consisting of Volume Data and Opaque Objects Using GPU

TAKASHI IMAGIRE,[†] HENRY JOHAN,[††] NAOKI TAMURA[†††] and TOMOYUKI NISHITA[†††]

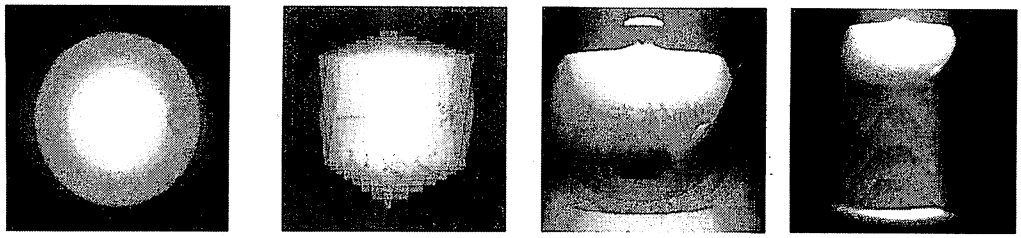
The technique for rendering volume data (smoke, dust, mist, etc.) using GPU by rendering sample planes which are the cross sections of the volume data, is well known. However, when we place opaque objects inside the volume data, this method generates aliasing (stripes pattern) at the boundaries where the sample planes intersect the opaque objects. In this paper, we propose a high quality and real-time volume rendering technique that does not generate aliasing at the boundaries between opaque objects and the volume data by using GPU to determine the visible regions to sample the volume data. Moreover, we remove aliasing at the shadows casting by opaque objects by interpolating the information at four nearby locations instead of using the information at one location when we compute the occlusion from a shadow map.

1. はじめに

ボリュームレンダリングは3次元のデータセット（ボリュームデータ）を2次元の画面に表示するための可視化手法である．ボリュームレンダリングには幾つかの手法が知られているが，その方法は大きく2種類に分類できる．1つはマーチングキューブ法[1]に代表される等値面を計算してポリゴンを生成し，そのポリゴンを表示する間接法である．この方法は，等値面を表現するためのポリゴンを生成するための中間的なデータの追加のメモリ領域や処理負荷を必要とする．もう1つは，レイトレーシング法やレイキャスティング法[2]等の描画面から視線方向に沿って輝度を積分する直接法である．直接法では，視線上の複数の点でデータをサンプリングすることで近似的に積分計算を行う．これらの手法は画素ごとの色計算を行うため，現状ではCPUによるリアルタイム表示の計算は難しい．

近年，GPU(Programmable Graphics Processing Unit)の進歩を背景として，リアルタイムにボリュームデータを可視化する研究が行われている[3][4]．GPUを利用したボリュームレンダリングの高速化として，サンプル面でボリュームをスライスしたプリミティブを重ね描きする方法が提案されている[5]．この手法は，画素ごとに並列に計算できるためGPUの機能を利用

† 株式会社バンダイナムコゲームス
Namco Bandai Games Inc.
†† 南洋理工大学
Nanyang Technological University
††† 東京大学
The University of Tokyo



(a) 濃度が大きく変わる境界　(b) スライス面の境界　(c) 不透明物体との接触　(d) 不透明物体が落とす影

4 枚のスライス面による一様濃度の球

図 1　ボリュームレンダリングにおけるエイリアシング

した高速な描画が可能である．しかし，画質を高めるためには，スライス面の枚数を適宜増やす必要がある．ゲームなどのリアルタイム性が求められるアプリケーションでは，GPU の高速化が利用できるとはいえ，その負荷は低いとはいえず，少ない枚数のスライス面による表現が用いられている．

本稿では，テクスチャベースのボリュームレンダリングを扱う．ボリュームデータは，3 次元のボリュームテクスチャ(ソリッドテクスチャ)，もしくは幾つかの 2D や 1D のテクスチャの組み合わせとして与えられる．テクスチャの組み合わせとしては，2D のテクスチャを重ね合わせたり，複数のテクスチャの値を乗算などの算術的に合成する方法がある．どの場合にも，ボリュームデータをサンプリングするための 1 つの 3 次元的なローカル座標系の値 $(x, y, z : 0.0 \sim 1.0)$ から色および不透明度が計算できるものとする．ボリュームデータには不透明な部分も含まれるが，本稿ではポリゴンなどの他の手法によって描画される不透明物体と区別するために，ボリュームはボクセルごとに散乱光の強さを格納したデータで構成されているものとして扱う．また，ボリューム以外には半透明な物体はないものとする．描画するシーンは，ボリュームと不透明物体が含まれるものとする．

スライス面によるボリュームレンダリングにおいて，スライス面の枚数が不十分な場合には，以下の状況で縞模様を伴うエイリアシングが発生する（図 1）．

- 濃度が大きく変わる境界
- スライス面とボリュームの境界
- 不透明物体とスライス面が接触する周辺
- 不透明物体が落とす影の空間の境界

これらのエイリアシングは，動的に画面が変化した場合に縞模様の動きが鑑賞者の目に止まりやすく，特にリアルタイム性を求められるアプリケーションでは好ましくない．幾つかのエイリアシングを除去する手法が提案されているが，全てのエイリアシングを除去できる有効な手法はまだ提案されていない．本稿では，リアルタイム技法において既存手法よりも高品質なエイリアシングの除去を行う手法を提案する．

提案法では，視線方向のボリュームの厚みを検知し，その区間でのみ積分計算を行うことによりエイリアシングの発生を抑える描画方法を提案する．また，分散シャドウマッピング法[6] をスクリーンの画素の間隔に関係付けられたサンプル点で評価することによって，不透明物体が落とす影のエイリアシングを除去する．

本稿の構成は次の通りである．まず，2 節で関連研究を紹介する．次に本稿で提案するアルゴリズムを 3 節で説明する．さらに実装の要点を 4 節で述べる．そして，提案法による計算結果を 5 節に示す．最後に提案法の利点と今後の課題を 6 節にまとめる．

2. 関連研究

本節では関連研究を述べる．

2.1 GPU を利用したボリュームレンダリング

テクスチャベースのスライス法によるボリュームレンダリング[3] では，ボリュームとサンプル面との交わりを計算して多角形を生成し，これを描画する．プリミティブの各頂点には，3 次元のローカル座標を割り当てる．多角形は，画面の奥から手前等の一様な順に並べられ，整列順にフレームバッファに合成される．多角形をレンダリングする画素ごとの計算では，ローカル座標からボリュームデータが参照され，必要に応じて照明技法によって補正されて色や不透明度が決定される．多角形の生成にはローカル座標の軸に沿ってボリュームをスライスする方法 (object-aligned slices) と，視線方向を軸にしてスライスする方法 (image-aligned slices) がある．object-aligned slices では，ボリュームの回転にあわせて不透明度を補正する必要がある．image-aligned slices は，視線が変わるたびにスライス面を再計算する必要があり，リアルタイム性を保障するためには，スライス面の枚数を一定以下に抑える必要がある．

2.2 エイリアシングの除去

エイリアシングを除去する方法として，幾つかの手法が提案されている．

インターリーブサンプリング法[7]は，スライス面の位置をずらして描画した複数の画像を作成し，最終的に画素ごとに各画像を合成する重みを変えることによってエイリアシングが目立たない画像を作成する．この手法は，合成する重みを変化させる周期をパターンに持つノイズが発生することや，高速なレンダリングのためには，処理する画素数を減らすために小さなフレームバッファに描画したボリュームの画像を用いる必要があり，画質が下がるといった欠点がある．

球面ビルボード法[8]は，球形のボリュームデータをレンダリングするために，球形のボリュームと同じ広がりをもつポリゴンを視線方向に正面を向けてレンダリングする．その際に不透明物体の深度を参照することによって画素ごとにボリュームがレンダリングされる部分の厚みを求め，その厚みから算術的に不透明度を計算してエイリアシングのない描画をする．球面ビルボード法は，球形に近い形のボリュームデータしか扱うことができず，また不透明物体が落とす影を表現することができないという欠点がある．

土橋ら[9]は，光跡の表現に関して，影が落ちる領域を検知し，その領域にサンプル面を追加することにより影が作るエイリアシングを軽減する方法を提案した．あらかじめ，光が届く範囲のバウンディングボリュームを計算する．ボリュームのレンダリング時に，スライス面とバウンディングボリュームの交差判定を行い，交差しているスライス面の前後にサンプル面を追加することによりエイリアシングの軽減を図る．しかし，この手法は，不透明物体とめり込む境界におけるエイリアシングを除去するのが難しい．

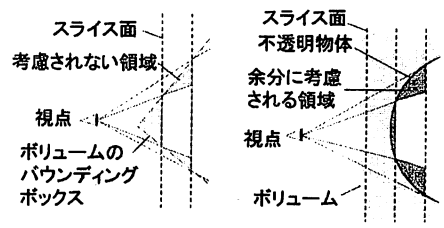
本提案法では，スライス面の代わりに凸包のプリミティブを描画することによって，スライス間隔に起因する問題を取り除く．その際に，不透明物体の深度を参照することによって，不透明物体とのエイリアシングも取り除く．また，影に関して，複数の値を補間することでエイリアシングのない画像を生成する．

3. 提 案 法

本節では，提案法のアルゴリズムを説明する．

3.1 提案法の基本概念

スライス法によるボリュームレンダリングでは，大きく分けて２つの要因でエイリアシングが発生する．１つは，ボリュームをレンダリングする際に考慮されない体積の誤差が空間的に不連続に変化することである．ボリュームをスライスして描画する方法では，あるスライス面から次のスライス面までの間の領域でスライス面と視線に囲まれない領域は考慮されない (図2(a))．ボリュームとスライス面が交差する周辺ではボリュームの厚みの評価の誤差が不連続に変化するため，エイリアシングが発生する．また，ボリュームが不透明物体と交差する周辺では，ボリュームのスクリーンからの深度がスライス面の深度で評価されるために，交差によるめり込みによって排除されるべき領域もボリュームが存在するものとして計算されてしまう (図2(b))．この問題では，不透明物体とスライス面の交わる点でボリュームの厚みが不連続に変化するように見えるため，エイリアシングが発生する．



(a) ボリューム境界での問題　(b) 不透明物体との問題

図 2 不正確な領域の見積もりによるエイリアシング

提案法では，ボリュームデータのバウンディングボックスを分割したプリミティブを作成し，ボリュームデータの３次元のローカル座標系に変換された不透明物体の位置座標および，プリミティブのローカル座標値を記録することによって，不連続な誤差を取り除く．

もう１つの要因は，サンプリング間隔とデータの分解能の不一致である．サンプリング間隔がデータの画素の間隔よりも広い場合には，読み込まれないデータが存在するため，エイリアシングが発生する．本稿では，影の生成にシャドウマップ法[10]を用いる．深度の評価を周辺の４点で行い，それらの結果を線形に補間することでエイリアシングを除去する．

3.2 凸包のプリミティブによるボリュームレンダリング

通常のスライス法では，スライス面をボリュームのバウンディングボックスで切り取った多角形を描画するが，この手法では，多角形の境界でエイリアシングが発生する．これを除去するために，凸包のプリミティブ (以降，単にプリミティブ) を導入する．ボリュームを覆うバウンディングボックスを計算し，スクリーンに平行に分割してプリミティブを生成する (図3)．プリミティブの各頂点には，ボリュームデータを参照するためのローカル座標値を割り当てる．

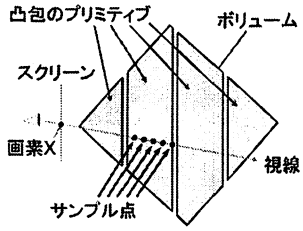


図 3 凸包のプリミティブ．ボリュームを一定の間隔で分割した要素の各表面を凸包のプリミティブとする．プリミティブは，等分割した複数のサンプル点を読み込みレンダリングする．

画素 X において，視線がプリミティブと交差した後に再び出て行く点のローカル座標系の値 $\mathbf{t}(X)$ に着目する．ローカル座標値 $\mathbf{t}(X)$ は，プリミティブが不透明物体と交差しない場合にはプリミティブの裏面のローカル座標値であり，交差する場合には交差する面の最前面におけるローカル座標値になる．ローカル座標値 $\mathbf{t}(X)$ は次の式で与えられる．

$$\mathbf{t}(X) = \begin{cases} \mathbf{t}^B(X), & d(X) \le z; \\ M^T\mathbf{x}(X), & otherwise, \end{cases} \quad (1)$$

ここで，$\mathbf{x}(X)$ は 4 次元の同次座標系のスクリーン座標系での最前面にある不透明物体の位置座標，z は不透明物体のスクリーンからの深度，M^T はスクリーン座標系からローカル座標系への変換行列，$\mathbf{t}^B(X)$ は画素 X におけるバウンディングボックスの背面におけるローカル座標値，$d(X)$ は画素 X におけるバウンディングボックスの背面のスクリーン座標系での深度である (図 4)．

プリミティブ内を N 回のサンプリングによって色や不透明度を決定する (図 3)．サンプル点 i におけるローカル座標値 $\mathbf{t}_i(X)$ は，次の式で計算される．

$$\mathbf{t}_i(X) = \mathbf{t}^B(X) + i * d\mathbf{t}(X), \quad (2)$$

$$i = 1, 2, \cdots, N, \quad (3)$$

$$d\mathbf{t}(X) = (\mathbf{t}^F(X) - \mathbf{t}(X))/N, \quad (4)$$

ここで，$\mathbf{t}^F(X)$ は画素 X におけるバウンディングボックスの表面におけるローカル座標値である．プリミティブを描画する色 $\hat{C}_N$ および不透明度 $\hat{A}_N$ は，以下の式の N 回目の試行の結果として計算される．

$$\hat{C}_i = A_i C_i + (1 - A_i)\hat{C}_{i-1}, \quad (5)$$

$$\hat{A}_i = (1 - A_i)\hat{A}_{i-1}, \quad (6)$$

$$\hat{C}_0 = (0,0,0), \quad \hat{A}_0 = 1, \quad (7)$$

ここで，サンプル点における色 C_i および 不透明度 A_i は，ローカル座標 $\mathbf{t}_i(X)$ から決定され，不透明度 A_i は読み込まれた値 A_i^0 から，以下の式で与えられるサンプリング間隔による補正を受ける．

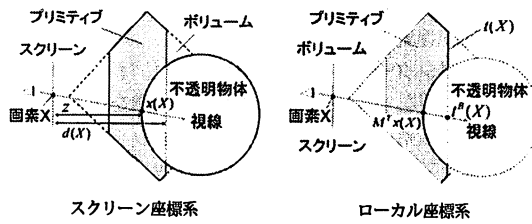


図 4 視線に伴う変数の定義

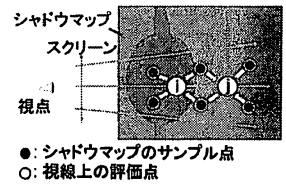


図 5 シャドウマップのサンプル点．評価点 i の計算に使用したサンプル点の内 2 点は評価点 j の計算にも使用される．

$$A_i = (||d\mathbf{t}(X)||/\Delta t)A_i^0, \quad (8)$$

ここで，$||\mathbf{a}||$ はローカル座標系におけるベクトル $\mathbf{a}$ のユークリッド距離であり，定数 Δt は基準となるサンプリング間隔である．物理的には，厚みに応じた指数関数的な補正が必要であるが，高速化のために，厚みに比例する近似的な計算を採用した．C_i は，ボクセルデータの色成分をそのまま用いた．

最終的な結果は，プリミティブを視線方向の奥から手前の順に描画し，得られた色 $\hat{C}_N$ および不透明度 $\hat{A}_N$ を，あらかじめ不透明物体が描画されたフレームバッファに下記の α ブレンディングの式で合成する．

$$C(X) = \hat{A}_N D(X) + \hat{C}_N, \quad (9)$$

ここで，$D(X)$ は，画素 X における描画前のフレームバッファの色である．

3.3 エイリアシングのない影の描画

ボリュームを指向性を持つ光が塵などによって散乱された光跡として表現する場合には，不透明物体が空間中で光を遮蔽する影の効果を考慮する必要がある．本稿では，分散シャドウマッピング法[6] により影を生成する．なお，ボリュームによる光の減衰は考慮せず，フレームバッファとの合成には加算の合成を行う．

サンプリング間隔がシャドウマップの画素の間隔よりも広い場合には，隣接するサンプル点に対する値の変化が激しくなり，影の境界部分にエイリアシングが生じる．エイリアシングの除去のために，サンプル点の前後および左右との中点における遮蔽情報の評価を行い，それらの平均値を影の強さとして照明計算に使用する (図 5)．サンプル点の中点は，サンプル点を

$\mathbf{t}_i(X)$ として，$\mathbf{t}_{i\pm0.5}(X \pm 0.5 * B(X))$ で与えられる．ここで，$B(X)$ は視線方向と光源方向に垂直で画素間の距離の大きさを持つベクトル $\mathbf{b}(X)$ をスクリーンに投影したベクトルで，$\mathbf{b}(X)$ は以下で与えられる．

$$\mathbf{b}(X) = normalize(\mathbf{V} \times \mathbf{L})/d(X) \qquad (10)$$

ここで，$\mathbf{V}$ は視線ベクトル，$\mathbf{L}$ はライトベクトル，$d(X)$ は サンプル点における視点からの深度である．

4. GPU 実装

本節では，実装の要点を示す．

4.1 ローカル座標値の格納とプリミティブの描画

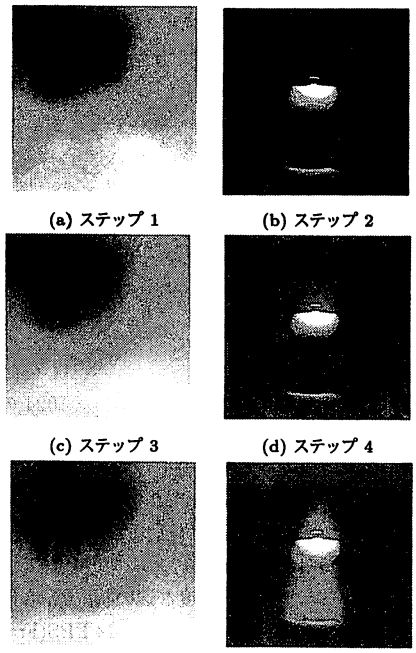
視点およびボリュームの位置が変化した時に複数のスライス面を設定し，バウンディングボックスを分割した M 個のプリミティブを生成する．

レンダリングは，$2M+2$ 回のステップで行われる (図 6)．ローカル座標値を格納するために別に設けたフレームバッファに不透明物体の位置座標をローカル座標系に変換し，値の x, y, z 成分を赤，緑，青 の各色成分に対応させて出力する．ローカル座標系への変換は，不透明物体をスクリーンに投影する行列にボリュームをローカル座標系からスクリーンに投影する行列の逆行列を掛けた変換行列を用意し，各頂点の位置座標に，その変換行列を作用させて行う．レンダリングするフレームバッファはローカル座標値の誤差が発生しないように十分な精度のフレームバッファを用意する必要がある．ボリュームデータの一辺のボクセル数が 256 より小さければ 8 ビットの整数フォーマットを用い，ボクセル数がそれよりも大きければデータを表現できるだけの浮動小数点フォーマット等の画像フォーマットを用いる必要がある．

次のステップでは，最終的なフレームバッファに不透明物体をレンダリングする．この描画はボリュームレンダリングとは無関係な通常の描画である．

3 ステップ目では，描画するフレームバッファを再びローカル座標値を格納するフレームバッファに戻し，最奥に位置するプリミティブの裏面を描画する．3次元のローカル座標値を色成分とみなした値を出力する．深度テストを有効にし，深度バッファは以前のステップの深度バッファと共有する．この結果，プリミティブの深度が不透明物体の奥に隠れる画素は深度テストにより破棄され，不透明物体のローカル座標値が記録される．

4 ステップ目では，最終的なフレームバッファに描画先を切り替え，プリミティブの表面を描画する．ステップ 3 で作成したローカル座標値の画像を描画する画素の位置から読み込み，端点のローカル座標値とする．もう一方の端点のローカル座標値をプリミティブの表面のローカル座標値とし，これらのローカル座標値を線形に補間したサンプル点を読み込んで色や不透明度を決定しフレームバッファと合成する．色や不透明度の計算では，GPU を利用して 1 度の描画で複数回のサンプリングを行う．



(a) ステップ 1　(b) ステップ 2

(c) ステップ 3　(d) ステップ 4

(e) ステップ 2M+1　(f) ステップ 2M+2

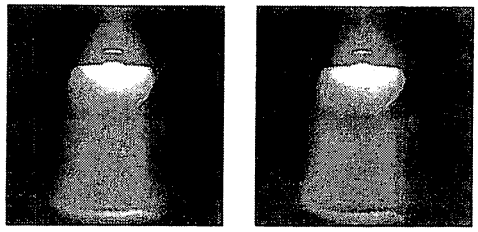
図 6 提案法の各レンダリングステップの結果

以降のステップでは，分割した回数 M だけ前 2 回の処理を繰り返す．

プリミティブの厚さやサンプリング回数 (サンプル点の個数) は，生成される画像の品質と GPU の性能によって決定される．今回は実験的にパフォーマンスを調査することによって回数を決定した．

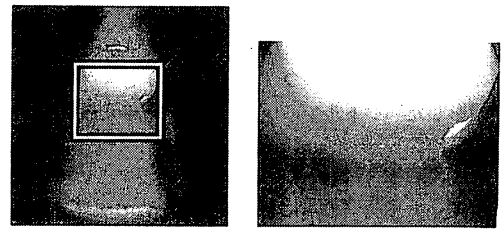
4.2 遮蔽情報の計算

シャドウマッピングを行う場合は，シーンを描画する前に，シャドウマップを作成する．プリミティブをレンダリングする際に，サンプル点を視線方向と反対側にサンプリング間隔の半分だけ戻しておき，視線および光源方向に垂直な方向にピクセル間隔の半分の大きさでローカル座標系を変位させた位置からシャドウマップを読み込んで各点における深度の差から遮蔽情報を計算する．次のサンプル点の評価時には，前のサンプル点との間にある 2 点の結果が流用でき，新たにサンプリングする点は半分の 2 点となる (図 5)．



(a)8 分割 4 回サンプリング　(b)4 分割 2 回サンプリング

図 7　結果画像



(a) 描画結果　(b) 中央部の拡大

図 8　128 枚のスライス面による描画

5. 結　果

本研究の結果を示す．シーンは，1024 × 1024 の 16 ビット精度の浮動小数点フォーマットのフレームバッファにレンダリングする．シャドウマップは点光源として作成し，サイズは 256 × 256 である．

プリミティブの数と，サンプリング回数を変えて速度や画質を調査した．エイリアシングが視覚的に現れない画質で，もっとも処理速度が速い状況は，8 個，4 回サンプリングの場合 (図 7(a)) で，43 FPS(Frames Per Second) で描画される．4 個，2 回サンプリングの場合 (図 7(b)) には 105 FPS で描画される．この場合も，画質は下がるが縞模様のエイリアシングは発生しない．

視線を軸にしたスライス法と比較した場合，32 枚のスライス面を描画したときには 98 FPS であったが，この場合には，不透明物体へのめり込みによるエイリアシングが視認できる (図 8)．エイリアシングが消えるまでスライス面を増やした場合には，256 枚のスライス面が必要になり，実行速度は 18 FPS であった．

計算時間の計測には，CPU: PentiumD 3.2GHz, メインメモリ: 1024MB, GPU: GeForce 7800 GTX (グラフィックスメモリ 256MB)) を用いた．API には DirectX 9 を用いている．

6. おわりに

本稿では，ボリュームレンダリングで発生するエイリアシングを，ボリュームの視線方向の厚みをプリミティブを用いて評価することによって除去する手法を提案した．さらに，ボリュームデータを光跡とみなす時に，交差する物体が落とす影によるエイリアシングも，複数のサンプル点による結果を補間することにより除去する手法を提案した．処理の大半を GPU のプログラマブルシェーダーで実装可能であり，アルゴリズム全体を通して高速に処理できる．

ただし，提案法はレンダリングされる画質が，プリミティブの個数やサンプリング回数に依存する．今回は，実験的にこれらの値を決定したが，今後は，ボリュームデータの解像度，不透明物体の細かさ，シャドウマップの大きさ，ハードウェアの性能から最適な値を自動的に決定したい．また，複数のボリュームデータが交差する状況や，複数のボリューム間の交差には対応していない．これらを今後の課題としたい．

参 考 文 献

1) W. E. Lorensen and H. E. Cline. Marching cube: a high resolution 3D surface construction algorithm. *Computer Graphics (Proc. SIGGRAPH 1978)*, pages 163–169, 1978.
2) H. Tuy and L. Tuy. Direct 2D display of 3D objects. *IEEE Computer Graphics and Applications, 4(10)*, pages 29–33, 1984.
3) M. Brady, K. Jung, H.T. Nguyen, and T Nguyen. Two-phase perspective ray casting for interactive volume navigation. *Visualization ' 97*, pages 183–190, 1997.
4) J. Kruger and R. Westermann. Acceleration techniques. for GPU-based volume rendering. *IEEE Visualization 2003*, pages 287–292, 2003.
5) T. J. Cullip and U. Neumann. Accelerating volume reconstruction with 3D texture hardware. *Tech. Rep. TR93-027*, University of North Carolina, Chapel Hill N.C..
6) W. Donnelly and A. Lauritzen. Variance shadow maps. *Proc. of the Symposium on Interactive 3D Graphics and Games*, pages 161–165, 2006.
7) A. Keller and W. Heidrich. Interleaved sampling. *Proc. of the 12th Eurographics Workshop on Rendering Techniques*, pages 269–276, 2001.
8) T. Umenhoffer, L. Szirmay-Kalos, and G. Szijarto. Spherical billboards and their application to rendering explosions. *Proc. of the 2006 Conference on Graphics Interface*, pages 57–63, 2006.
9) Y. Dobashi, T. Yamamoto, and T. Nishita. Interactive rendering of atmospheric scattering effects using graphics hardware. *Proc. of Graphics Hardware 2002*, pages 99–108, 2002.
10) Lance Williams. Casting curved shadows on curved surfaces. *Proc. of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, pages 270–274, 1978.