光線ボリウムバッファを用いたボリウムとポリゴンの混在シーンのレンダリング方法

A Method of Rendering Scenes Including Volumetric Objects Using Ray-Volume Buffers

梶原 景範[†], 高橋 裕樹[†], 正会員 中嶋 正之[†]

Kagenori Kajihara[†], Hiroki Takahashi[†] and Masayuki Nakajima[†]

Abstract A rendering method that simultaneously renders both volumetric and polygonal objects is described. Using the method also makes rendering volumetric objects intersected by polygonal ones in the same scheme possible. A ray-volume buffer that stores colors and the transmittance of a ray at each voxel is described. The ray-volume buffer treats volumetric objects as textures. Therefore, the method can be achieved using conventional graphics pipeline by generating a ray-volume in advance. In contrast with the Marching Cube algorithm, the method can generate images that maintain the features of volumetric objects using a small number of surfaces because it renders a volume by using textured boundary surfaces obtained through the volumes.

キーワード: コンピュータグラフィックス, バーチャルリアリティ, ボリウムレンダリング

1. はじめに

近年, バーチャルリアリティ(VR)等で用いられる実時間画像生成は, 高性能で安価なビデオアクセラレータやOpenGLが広く使われるようになり, 成熟した技術となりつつある. 一方, 近年ボリウムレンダリングに関する研究が盛んに行われ, 小規模のボリウムであれば, ボリウムレンダリング専用ボードを付加したPCにより実時間表示が可能になった.

今後, 映像のリアリティの向上のためには, ボリウムとポリゴンが混在したシーンの映像生成の必要性が高まると考えられる. 例えば, 手術シミュレータにおける身体あるいは臓器と, それに挿入された手術機器, 訓練用シミュレータにおけるボリウムでできた煙幕とその中を進む戦車, ゲームにおけるボリウムでできたブッシュとポリゴンでできた怪獣などの映像である.

本論文では, ボリウムとポリゴンの両オブジェクトが混在し, 互いに干渉するシーンの映像を生成する効果的な方式について提案する. 特に, 航空機や自動車の訓練シミュレータの模擬視界装置として, 現在広く利用さ

れているデプスバッファ方式のポリゴングラフィックシステムにボリウムオブジェクトの表示能力を付加するのに好適な方法を提案する.

本論文で提案する方式は, フレーム処理の前段でボリウム中を進む光線の色および透過度をボクセルごとに計算し, ボリウムと同サイズの光線ボリウムバッファに記憶しておくことにより, ボリウムとポリゴンの干渉の計算を単純化することが特徴である. 光線ボリウムバッファの生成によって, 後段では従来のポリゴングラフィックパイプラインでボリウムとポリゴンを統一的に処理し, それらの混在したシーンの映像生成を行うことが可能となる.

本提案方式では, ボリウムとポリゴンの干渉は光線ボリウムバッファを用いた3次元のソリッドテクスチャマッピングにより実現する. また, ボリウムそのものもテクスチャ付きのポリゴンで表現する. ここで述べるボリウムは, 一般には煙や土ぼこりや臓器のような半透明オブジェクトであるが, 樹木のような形状が複雑で不透明な物体にも適用することができる.

ボリウムモデルとポリゴンモデルを融合する場合, ポリゴンをより自由度の高いすなわち情報量の多いボリウムに変換するのが通常であるが¹⁾, 本提案方式ではボリウムをポリゴンに変換して取り扱う. その第一の理由は, ポリゴンをボクセルで表現する場合の量子化の問題¹⁾を回避できることである. 第二に, 模擬領域

2001年12月, 映像情報メディア学会画像情報システム研究会で発表

2002年3月4日受付, 2002年8月19日最終受付, 2002年8月27日採録

[†]東京工業大学 大学院 情報理工学研究科

(〒152-8552 目黒区大岡山2-12-1, TEL 03-5734-2183)

[†]Graduate School of Information Science & Engineering, Tokyo Institute of Technology

(2-12-1 Ookayama Meguroku Tokyo, 152-8552 Japan)

全体をボリュメトリックに表現すると、データ量が膨大になり現在ではまだ非現実的なメモリー量および計算量を必要とするからである。VR等のビジュアルシステムは、当分の間はまだポリゴンが中心であり、部分的にボリュームモデルを必要とする状態であると考える。

ボリュームをポリゴンに変換しレンダリングする間接的ボリュームレンダリング法として Marching Cube⁷⁾ が広く知られている。しかし、Marching Cube 法では、質感・量感の表現が不充分であったり、複雑なシーンの場合ポリゴン数が膨大になるという問題がある。

2. 関連研究と本提案方式の特徴

2.1 関連研究

Kregerらは、ポリゴンと干渉したボリュームをレンダリングするために、ポリゴンをボリュームの層ごとにスライスすることによって、ボクセルとポリゴンの位置関係を調べて層ごとに Ray-cast でレンダリングしている³⁾。彼等の方式の長所は、半透明ポリゴンのデプスソートが不要なことであり、短所はスライスにより1つの三角形が多数の小さい三角形に分割されること、および処理が複雑なことである。

Giertsenらは、ボリュームとサーフェスを同時にレンダリングするのに Scan-Plane Buffer (SPB) と Multiple-Layer z-Buffer (MZB) を用いてスキャンラインオーダのレンダリングをした。彼等の方式の特徴は、ボリュームのメッシュ構造を問わないことである。欠点としては、エイリアシングが生じやすいことである。

Levoyは Ray-Trace 法をボリュームレンダリングに拡張し、ボリュームとポリゴンの混在したシーンの映像生成を行う方法を提案した⁵⁾。彼の方法の長所は、リアルな映像生成とアルゴリズムが比較的シンプルであることであり、短所としては計算負荷が大きいこと、エイリアシングが生じやすいことである。

Bitterらは、ボリュームレンダリングにおいて、色の合成計算をボリュームの層ごとに計算し積算する Ray-Slice-Sweep Volume Rendering 法を提案している⁶⁾。しかし、Ray の計算に線形補間を使用し、しかも Sweep しているので、本論文で指摘している光線の拡散が生じ映像はボケるという欠点がある。

2.2 本提案方式の特徴

ポリゴンと干渉したボリュームをレンダリングする場合、ボリュームの、ポリゴンの手前にある部分と後ろ側にある部分を知る必要がある。関連研究では、それぞれこのための処理に工夫がなされている。すなわち、Kregerらはボリュームの層ごとにボリュームをポリゴンとともにスライスし、Giertsenらはスキャンラインごとにビュウボリュームをスキャンラインプレーンでスライスし、Levoyは Ray-Trace 法によって、ボリューム中のポリゴンの位置を確認しながらレンダリングしている。

本提案の方式では、このための処理を必要とせず、光線ボリュームバッファと3Dテクスチャマッピングによってこの機能を実現する。光線ボリュームバッファの生成は、ポリゴンとの干渉を全く考慮する必要がなく、単純なボリュームレンダリングと等価な計算量で可能である。また、3Dテクスチャマッピングは、最近のグラフィックハードウェアではレンダリング速度に対するペナルティなしにポリゴンのレンダリングができる。

本方式によれば、光線ボリュームバッファの生成部分をソリッドテクスチャジェネレータとして局所的に追加することによって、従来型のグラフィックスシステムの機能拡張が可能である。

本方式では、半透明ポリゴンはデプスソートされている必要があるが、本方式の適用目的である現在主流のデプスバッファ方式のグラフィックスシステムでは、半透明ポリゴンは本来 BSP (Binary Space Partitioning) 等によりソートされている必要があるため、このことは特に制限にはならない。

本方式の欠点は、ボリュームデータを直接レンダリングする方式に比べて、光線ボリュームバッファ用の大量のメモリーを必要とすることである。

ボリュームもポリゴンとして表示するが、全ボクセルを処理して得たテクスチャをマッピングするので Marching Cube 方式と異なりボリュームの持つ質感・量感も十分に表現できる。

3. 光線ボリュームバッファ方式

本論文では、投影法は平行投影とする。この節では、まず基本的なアプローチを述べ、光線ボリュームバッファ生成すなわち光線の伝播の模擬について述べ、ボリュームとポリゴンの干渉の模擬方法について述べる。

3.1 基本的なアプローチ

ボリュームとポリゴンの混在シーンの光線ボリュームバッファを用いたレンダリングの基本的なアルゴリズムは次のとおりである。

Step 1: 光線ボリュームバッファの生成

光線ボリュームの各ボクセルに、視線と平行な光線がそのボクセルを通過するときの色とそこまでの透過度を計算し格納する。

Step 2: ポリゴンの表示

ポリゴンがボリュームと干渉するとき上記の光線ボリュームバッファの内容をソリッドテクスチャマッピングして表示する。

Step 3: ボリュームの表示

ボリュームは、光線ボリュームバッファをソリッドテクスチャマッピングしたボリュームのバウンダリポリゴンとして表示する。

Step 1 から Step 3 は毎フレーム実行される。バウンダリポリゴンはボリュームの形状を直方体とするとその6面であって、直方体の内向きの法線を持つものとする。

3.2 光線ボリュームの生成-光線の伝播の模擬

図1にボリュームデータの構造を平面図で示す。ボ

図1 ボリュームデータの構造
Volume Data Structure

ボリュームは直方体とし、ボクセルの位置を規定する座標を (i, j, k) とする。 k 軸はボリュームの直交する3稜線の中で視線方向ベクトル $\mathbf{n}_{eye}$ と最も平行な稜線の上に奥行き方向にとる。 i, j 軸は右手系になるように他の2つの稜線の上に取り、方向は $\mathbf{n}_{eye}$ の各成分が正となるように設定するものとする。

光線ボリュームの位置 (i, j, k) のボクセルにおける色 $c_{tx}(i, j, k)$ と透明度 $t_{tx}(i, j, k)$ は、スクリーンからボクセル (i, j, k) までの奥行きを z とすると、次式で与えられる。積分はボクセル (i, j, k) を通る光線に沿って行う。

$$c_{tx}(i, j, k) = \int_0^z \psi(\eta) e^{-\int_0^\eta \phi(\xi) d\xi} d\eta \quad (1)$$

$$t_{tx}(i, j, k) = e^{-\int_0^z \phi(\xi) d\xi} \quad (2)$$

ここで、 $\psi(\eta)$ は積分路上 η におけるボリュームの色、 $\phi(\xi)$ は同じく ξ における不透明度である。

(1) 線形補間とその問題点

光線ボリュームバッファの生成において、ボクセル (i, j, k) を通過する光の色 $c_{tx}(i, j, k)$ とそこまでの透過度 $t_{tx}(i, j, k)$ を計算するとき、ボリュームは離散的に表現されているので式 (1) および式 (2) は数値計算をすることになり次式となる。また、光線は視点に近い $k-1$ 層のボクセルの間を通過してくる。これに対処するために Raycast 法などでは次のような線形補間を行う。

$$c_{tx}(i, j, k) = c(i, j, k) + t(i, j, k) \phi(i, j, k) \psi(i, j, k) \quad (3)$$

$$t_{tx}(i, j, k) = t(i, j, k) (1 - \phi(i, j, k)) \quad (4)$$

ここで、 $\psi(i, j, k)$ はボクセル (i, j, k) における色、 $\phi(i, j, k)$ は不透明度である。 $c(i, j, k)$ は次式で与えられ、 $t(i, j, k)$ も同様な式になる。

$$\begin{aligned} c(i, j, k) = & (1 - \beta) \{ (1 - \alpha) \cdot c_{tx}(i, j, k-1) \\ & + \alpha \cdot c_{tx}(i-1, j, k-1) \} \\ & + \beta \{ (1 - \alpha) \cdot c_{tx}(i, j-1, k-1) \\ & + \alpha \cdot c_{tx}(i-1, j-1, k-1) \} \end{aligned} \quad (5)$$

ここで、 $\alpha = e_i / |e_k|$, $\beta = e_j / |e_k|$, $\mathbf{n}_{eye} = (e_i, e_j, e_k)$

図2 線形補間によるロブスター
Lobster rendered by linear-interpolation

図3 線形補間による光線拡散
Ray divergence by linear-interpolation

である。

しかし、この線形補間では図2に示すように光線の拡散により像がボケてしまう。 $\alpha = 0.5$ のときすなわち視線が k 軸と約 26.6 度をなすときを考えると k 層のあるボクセル (i, j, k) は手前の $k-1$ 層の相隣るボクセル $(i, j, k-1)$ とボクセル $(i-1, j, k-1)$ から2分の1ずつの輝度を受け取る。これは逆に $k-1$ 層のボクセルの輝度は k 層の相隣る2つのボクセルに2分の1ずつに分散し、 $k+1$ 層の3つのボクセルに4分の1, 2分の1, 4分の1と拡散していくことを意味する。

図3は、この様子を2次元で表したものである。すなわち、奥行き方向が k 軸、手前右手方向が i 軸、高さ方向が光線の輝度であって、 i 軸上で $i=0$ のみが輝度1で他で輝度0のパターンが k 軸と $1/2$ の傾斜をなして奥行き方向に光が伝播するとき光は拡散し減衰してしまう。しかも、このボケの程度は見る方向により異なる。見る方向がボリュームの軸に対して平行または 45 度のときは光線の拡散が起こらずボケは生じず、 $\tan^{-1} 0.5$ 度のときボケは最大になる。

(2) ビームの先鋭化1-高階の補間

ボケを改善するために光ビームの先鋭化を次のように行う。すなわち、式 (6) に示すように k 層に到達する光線の色を、 $k-1$ 層の左右 $2m+1$ ボクセルの色を $g(u)$ という重み関数で重み付けして加算して求め、そして光線ができるだけ拡散せず遠方まで到達するように $g(u)$ を決める。以下簡単のため2次元で説明する。

$$c(i, k) = \sum_{u=-m}^m g(u - \alpha) c(i - u, k - 1) \quad (6)$$

式 (6) において、次式が成立するように $g(u)$ を決める。

$$c(i, n) = c(i - \alpha n, 0) \quad (7)$$

すなわち、0層の光線パターン $c(i, 0)$ が n 層まで正確に伝播しているということは、 n 層の光線パターンが0層の光線パターン $c(i, 0)$ をちょうど αn だけずらしたのになっていることであって、式 (7) で表せる。

式 (6) を式 (7) に代入しフーリエ変換すると、 $m = \infty$ の場合、方程式 (7) を満たす $g(u)$ は $\text{sinc}(u)$ であることがわかる。

図4 $g(u)$ による光線ビーム先鋭化 ($m=4$)
Ray sharpened by $g(u)$, $m=4$

図5 $g(u)$ による光線ビームの先鋭化 ($m=8$)
Ray sharpened by $g(u)$, $m=8$

式(6)において m が有限, 特に小さいとき $c(i, k)$ は発散するので, 窓関数 $w(u)$ を用いて $g(u) = \text{sinc}(u)w(u/m)$ として用いる. 実験では $w(u/m)$ として次式で与えられる Blackman の窓関数を用いた.

$$w(u/m) = .42 + .5 \cos(\pi u/m) + .08 \cos(2\pi u/m) \quad (8)$$

このように定めた $g(u)$ と式(6)を用いたときの光線の伝播の様子を, $m=4$ の場合を図4に, $m=8$ の場合を図5に示す. 座標系は図3と同じである. 拡散がかなり改善されているのがわかる.

この $g(u)$ を用いると $c(i, j, k)$ は次式で与えられ, $t(i, j, k)$ も同様な式で与えられる.

$$c(i, j, k) = \sum_{v=-m}^m \sum_{u=-m}^m g(v-\beta)g(u-\alpha) \cdot c_{tx}(i-u, j-v, k-1) \quad (9)$$

(3) ビームの先鋭化2-無補間

光線の拡散を避ける他の方法として, 補間を行わず光線に近い方のボクセルのみを使用して光線の色と透過度を計算する方法がある.

これにより光線の拡散を避けることができるが, 当然この場合最大 ± 2 分の1ボクセル間隔の位置誤差を生じることになるので, それを許容するかマッピングに使用前に補正が必要である.

3.3 ボリュームと干渉したポリゴンの表示

図6は, スクリーン上の画素 (p, q) において奥行き z_1, z_2 に, それぞれポリゴン P_1, P_2 が挿入された状態を示す. 色 C_{p_1} の不透明なポリゴンを位置 z_1 に挿入したときに見える色 C_{pq} は次式で与えられる.

$$C_{pq} = C(z_1) + t(z_1)C_p + t(z_1)t_p D(z_1, z_e) \quad (10)$$

ここで, $C(z_1), t(z_1)$ は次式で与えられ, 前処理として計算され光線ボリュームバッファに格納されている.

$$C(z_1) = \int_{z_0}^{z_1} \psi(\eta) e^{-\int_{z_0}^{\eta} \phi(\xi) d\xi} d\eta \quad (11)$$

$$t(z_1) = e^{-\int_{z_0}^{z_1} \phi(\xi) d\xi} \quad (12)$$

また, $D(z_1, z_e)$ は次式で与えられる.

$$D(z_1, z_e) = \int_{z_1}^{z_e} \psi(\eta) e^{-\int_{z_1}^{\eta} \phi(\xi) d\xi} d\eta \quad (13)$$

図6 Polygon の挿入
Volume and inserted polygons

$D(z_1, z_e)$ は次のように変形できる.

$$D(z_1, z_e) = e^{\int_{z_0}^{z_1} \phi(\xi) d\xi} \left(\int_{z_0}^{z_e} \psi(\eta) e^{-\int_{z_0}^{\eta} \phi(\xi) d\xi} d\eta - \int_{z_0}^{z_1} \psi(\eta) e^{-\int_{z_0}^{\eta} \phi(\xi) d\xi} d\eta \right) \quad (14)$$

$$= (t(z_1))^{-1} (C(z_e) - C(z_1)) \quad (15)$$

式(10)は結局次式のようにになる.

$$C_{pq} = C(z_1)(1 - t_p) + t(z_1)C_p + t_p C(z_e) \quad (16)$$

透明度がそれぞれ t_{p_1}, t_{p_2} の2枚の半透明ポリゴン P_1, P_2 が順に z_1, z_2 に挿入されたときに見える色は次式となる.

$$C_{pq} = C(z_1) + t(z_1)C_{p_1} + t(z_1)t_{p_1} \{ D(z_1, z_2) + s(z_1, z_2)C_{p_2} + s(z_1, z_2)t_{p_2} D(z_2, z_e) \} \quad (17)$$

$s(z_1, z_2)$ は次式で表され, 結局 C_{pq} は式(19)で表される.

$$s(z_1, z_2) = e^{-\int_{z_1}^{z_2} \phi(\xi) d\xi} = t(z_2)/t(z_1) \quad (18)$$

$$C_{pq} = C(z_1)(1 - t_{p_1}) + t(z_1)C_{p_1} + t_{p_1} \{ C(z_2)(1 - t_{p_2}) + t(z_2)C_{p_2} \} + t_{p_1}t_{p_2} C(z_e). \quad (19)$$

この式から, 半透明ポリゴンが挿入されると, アルファバッファの透明度 $t_{p_1} \cdots t_{p_{i-1}}$ で減じた $C(z_i)(1 - t_{p_i}) + t(z_i)C_{p_i}$ をフレームバッファに加算し, アルファバッファの透明度にポリゴンの透明度 t_{p_i} を乗じておくことによって任意枚の半透明ポリゴンの挿入を模擬できることがわかる.

4. 実験

4.1 シミュレーション

提案の方式に基づいて, 水素分子の電荷密度分布を表

図 7 水素分子と三角形
Electron cloud of H_2 and inserted opaque polygon

図 8 雲中の紙飛行機
Cloud and paper planes

図 9 水素分子と半透明および不透明三角形
Opaque and translucent triangles in electron cloud of H_2

図 10 ロブスターとナイフ
Lobster and inserted knife

すボリウムの中に差し込まれた三角形と、ボリウムモデルの雲の中を飛行する紙飛行機をシミュレーションした結果をそれぞれ図 7 および図 8 に示す。図 9 は、半透明ポリゴンと不透明ポリゴンが挿入された水素分子のシーンを、図 10 は、ボリウムモデルのロブスターにナイフを入れたシーンをシミュレーションした結果である。

光線ボリウムバッファの生成法はすべて無補間法で行った。光線ボリウムバッファの生成において、 $m = 4$ でビームの先鋭化を行った場合と、通過する光線の近い方のボクセルのみを用いた無補間の場合とでは、水素の電荷密度分布や雲ようなぼんやりした物体では画質の差はほとんどなく、ロブスターにおいては大きな画質の差はないが無補間の方が幾らかざらついた画質であった。

シミュレーションは PentiumIII 700MHz を用いてソフトウェアで行った。水素分子の電子密度分布はボリウムサイズは $160 \times 80 \times 100$ ボクセルである。光線ボリウムバッファの生成には無補間で約 4.9 秒、ボリウムと不透明および半透明の 2 枚の三角形の 640×480 画素スクリーンへのレンダリングに約 1.0 秒を要した。

雲と紙飛行機はボリウムサイズは $180 \times 80 \times 140$ ボク

セルである。光線ボリウムバッファの生成に無補間で約 3.9 秒、ボリウムと 2 機の紙飛行機の 640×480 画素スクリーンへのレンダリングに約 1.2 秒を要した。

ロブスターのボリウムサイズは $160 \times 160 \times 34$ ボクセルである。ロブスターとナイフにおいて、光線ボリウムバッファの生成に線形補間で約 13.5 秒、ビーム先鋭化法で $m = 4$ のとき約 87.1 秒、無補間で約 9.7 秒を要した。ボリウムおよび干渉ポリゴンの 640×480 画素スクリーンへのレンダリングに約 0.57 秒を要した。

4.2 考 察

処理量について Kreeger らの方法と比較検討する。Kreeger らのレンダリングはボリウムの各層に対して次のステップを繰り返す。

- Step 1: ボリウムのスライスをテクスチャ付きポリゴンとして、デプスチェック付きで表示
- Step 2: bucket ソートにしたがって次のスライスとの間のポリゴンをスライスおよびその三角形化
- Step 3: 三角形をデプスチェック付きでテクスチャマッピングして表示
- Step 4: bucket ソートシステムの更新

Step1 は本提案方式の光線ボリウムの生成に対応し、

本方式の場合デプステストは不要である。デプステストなしのポリゴンのレンダリングは、実験によると10%早くなった。文献³⁾によればポリゴンは数が2-3.5倍に細分化されるので、ポリゴンのシェーディング面積は一定としても、三角形のセットアップ処理(エッジの傾斜や輝度の変化率の計算等)は、本方式に対してそれだけ掛かることになる。ポリゴンが比較的小さいとして(大きければポリゴン数の増加率は2-3.5倍より遥かに大きくなる)、三角形のセットアップ処理とシェーディング処理が同等とすると、Kreegerらの方法は本提案方式に対して3/2から4.5/2倍のポリゴン表示時間を要することになる。また、Step2およびStep4は本提案方式では不要である。

各ステップの処理量の割合をStep1およびStep3がそれぞれ全処理の50%および40%、Step2とStep4で10%とする推測すると、本提案方式はKreegerらの方法に対して $50 \times 0.1 + 40 \times (1/3 \sim 2.5/4.5) + 10\% = 28 \sim 37\%$ 処理が軽いことになる。

また、光線ポリウムバッファの生成において、Ray-cast法などで一般的に用いられている線形補間は光線の発散を招き不適当であることが明らかになった。

本論文では、投影方法として平行投影について述べたので、このままでは眼球の手術のような奥行き的小さい応用に適用が制限され、広い模擬領域の模擬には透視投影に拡張する必要がある。しかし、本論文で、従来型のデプスバッファ方式のポリゴングラフィックスにポリウムとポリゴンが干渉したシーンのレンダリング機能を簡単に付加するのに、光線ポリウムバッファとソリッドテクスチャを用いる本手法が有効であることを充分に示すことができたと考える。

半透明ポリゴンは奥行きにしたがってソートされている必要があるが、本方式の適用目的のデプスバッファ方式のグラフィックスシステムは本来半透明物体はBSP法などによりソートされている必要があり、特にこのことは制限にならないと考える。

光線ポリウムバッファの生成には、ポリウムサイズの3乗に比例した大きな計算パワーを必要とするが、最近は一チップで 256^3 ボクセル程度のポリウムをリアルタイムでレンダリングすることが可能となったので⁸⁾、そのようなチップを開発すればポリウムを含むシーンをPCレベルで実時間発生できるものと考えられる。

本方式を用いたビジュアルシステムの構成を図11に示す。ポリウムデータに関する処理を行う局所的な拡張のみで、構成は従来のレンダリングパイプラインとほぼ同じ構成になることを示している。

5. おわりに

本論文では、フレーム処理の前前段で光線ポリウムバッファを生成しておくことにより、ポリウムとポリ

図11 本方式によるビジュアルシステムのブロック図
A block diagram of a visual system based on the proposed method

ゴンを統一的にレンダリングする方法を提案した。ポリウムとポリゴンの干渉の計算には、複雑な処理を必要とせず、テクスチャマッピングの手法を用いて実現したことが特徴である。

本論文では、平行投影に限定した方式を提案したが、今後の課題としては、訓練用シミュレータ等に応用するには透視投影に拡張する必要がある。特に視点がポリウムの中に入った場合も許すようなアルゴリズムを検討する予定である。

【文 献】

- 1) A. Kaufman, D. Cohen, R. Yagel, "Volume Graphics", IEEE Computer, vol.26,no.7,pp51-64, July 1993.
- 2) A. Kaufman, D. Cohen, R. Yagel, "サーフェスグラフィックスからポリウムグラフィックスへ", 日経CG, 1月号, pp148-155, 2月号 pp130-137, 1994(上記邦訳).
- 3) K. Kreeger, A. Kaufman, "Mixing Translucent Polygons with Volumes", Proceedings Visualization '99, pp.191-198, 1999.
- 4) C. Giertsen, A. Tuchman, "Fast Volume Rendering with Embedded Geometric Primitives", Visual Computing —Integrating Computer Graphics with Computer Vision, T.L. Kunii(ed), Springer Verlag, pp.253-271, 1992.
- 5) M. Levoy, "A Hybrid Ray Tracer for Rendering Polygon and Volume Data" IEEE Computer Graphics & Applications 10(2), 33-40, Mar. 1990.
- 6) I. Bitter, A. Kaufman, "A Ray-Slice-Sweep Volume Rendering Engine", In Proc. of Eurographics/SIGGRAPH workshop on graphic hardware 1997, pp.121-130, 1997.
- 7) W. Lorensen, E.H.Cline, "Marching Cubes: A High Resolution 3-D Surface Construction Algorithm", In Proc. of SIGGRAPH '87 in Computer Graphics 21,4, July 1987.
- 8) H. Pfister, J. H. Hardenburg, H. Lauer, S. Saylor, "The Volume-Pro Real-Time Ray-Casting System" Proceedings of the ACM SIGGRAPH '99 Conference pp131-138, Aug. 1999.

梶原 景範 1962年、東京大学工学部卒業。1964年、同大学院博士前期課程数物系研究科了。1967年、同大学院博士後期課程数物系研究科中退。同年、三菱電機株式会社入社。後に三菱プレシジョン(株)に転籍。フライトシミュレータ、特にその模擬視界装置の開発に従事。2001年、東京工業大学大学院博士後期課程情報理工学研究科入学。コンピュータグラフィックスの研究に従事。

高橋 裕樹 1990年、東京工業大学工学部制御卒業。1992年、同大学院博士前期課程物理情報工学専攻修了。1994年、同大学院博士後期課程物理情報工学専攻中退。1994年、同大工学部情報工学科助手。画像処理、ニューラルネットワーク、Genetic Algorithmsなどの研究に従事。

中嶋 正之 1969年、東京工業大学工学部電気卒業。1975年、同大学院博士課程了。同年、同大勤務。1983年、同大映像情報工学研究施設助教授。1992年、同大電気電子工学専攻教授。1994年、同大学院情報理工学研究科教授。現在に至る。コンピュータグラフィックス、コンピュータアニメーション、画像処理の研究に従事。工学博士。正会員。