芸術科学会論文誌 Vol. 9, No. 3, pp. 93 – 101

エネルギー波表現のリアルタイムレンダリング

阿部雅樹† 渡辺大地††

† 東京工科大学大学院バイオ・情報メディア研究科 メディアサイエンス専攻
†† 東京工科大学メディア学部

Real-time Rendering of Energy-wave Expression

Masaki Abe† Taichi Watanabe††

†Graduate School of Bionics, Computer and Media Sciences, Tokyo University of Technology
††Faculty of Media Science, Tokyo University of Technology
becktherippe @ gmail.com, earth @ aqua.media.teu.ac.jp

アブストラクト
近年、アクションや格闘を主題としたアニメーションやビデオゲーム等の創作コンテンツ上では、エネルギーの塊が強く発光、形状変形を伴いながら特定の目標地点へ移動するといった現象をエネルギー波と呼び、攻撃方法の 1 つや特殊効果として用いる事が多い。本研究は、3 次元のビデオゲーム等のインタラクティブな創作コンテンツ内における新たなエネルギー波表現方法を提案する。本手法はボリュームレンダリング手法の 1 つであるレイキャスティング法と基本的な概念は同様である。線積分式となる関数を複数組み合わせエネルギー波の 3 次元分布状況を規定する事で、ボリュームデータ生成処理を省きながらエネルギー波描画を行った。更に積分計算に GPGPU を使用することで、描画処理速度の向上を図った。このことにより、エネルギー波の光の強さを正確に表現しつつ、エネルギー波の形状変形をリアルタイムで実現した。

Abstract
The expressions that a lump of the energy strongly emits light and transforms shape are used as method of attack or special effects in the creation contents. That expressions are called Energy-Wave. This study suggests a new Energy-Wave expression method, in interactive creation contents such as the 3 dimensional video game. The basic concept of this technique is similar to Ray-Casting which is a kind of the Volume Rendering. We omitted a generation of volume data by prescribing 3 dimensional energy distribution data by a line integral. Furthermore, we succeeded of speed up by calculating at line integral function using GPGPU. From these, we drew strength of the light and shape deformation of the Energy-Wave precisely in real-time.

1 はじめに

近年、アニメーションやビデオゲームといった創作コンテンツ上で 3DCG を用いた様々な表現が開発されている。その中でもエネルギーの塊が強く発光、形状変化するといった現象はアクションや格闘を主題とした創作コンテンツ内ではよく見かけ、攻撃方法の 1 つや特殊効果として用いる事が多い。本研究では、そうした特殊効果をエネルギー波と名付け、「空間中のエネルギーの密度が高い場所が強く発光する」「形状変化を伴いながらある地点に向かって移動する」現象と定義する。エネルギーとは空間中に存在するエネルギー波を構成する要素で、3 次元空間上に分布し、密度が高い部分が強く発光するものとする。

3 次元のビデオゲーム内において、エネルギー波を表現する為に用いる技術で現在一般的なものは、テクスチャ表現 [1] である。

1 枚の矩形ポリゴンで表現するビルボードテクスチャでは、エネルギー波を単一の 2 次元画像として事前に複数枚用意し、プレイヤーの視点変更などに応じて画像を回転・切り替える事で、1 つのエネルギー波を表現している。また、球体や円柱形などのプリミティブ形状に対してテクスチャを貼り付けることでエネルギー波を表現する場合もある。テクスチャ表現はデータ容量が非常に軽く、リアルタイム描画性に優れているが、事前に用意したテクスチャ画像のみで表現を行うため、3 次元空間中のエネルギー分布状態を考慮していない。そのため、プレイヤー視点を動的に任意に変更した場合、その視点から見たエネルギー波の光の強さや形状を正確に表現することは不可能となる。

ポリゴンなどの境界表現以外で、3 次元空間中のエネルギー分布状態を表現するデータ構造で代表的なものは 2 通りある。第一にボクセルを用いたボリュームデータ表現 [2] である。ボリュームデータとは任意の 3 次元空間を一定の領域でサンプリングした集合データであり、個々をボクセルと呼称する。ボリュームデータは物体の内部構造 [3] や濃淡、煙等の不定形自然現象 [4] を表現する場合に適当なデータ構造である。高精細な表示が可能であるが、その分データ容量が肥大化する傾向に有り、数値積分など計算コストの高い表示処理を求められる [5]。近年では GPU(Graphics Processing Unit) を用いた高速な表示手法 [6][7] が数多く研究されているが、インタラクティブなコンテンツ内等で使用するには、データ容量や使用状況に制限がかかる事が多い。

第二に関数を用いたエネルギーの分布表現である。代表的なものは陰関数表現であり、BlobbyModel[8] やメタボール [9] といった形状表現が有名である。陰関数表現では関数の値を元に、空間を物体の内側、外側、表面と分類する。等値面を物体の表面として定義する手法であり、有機的な曲面形状や幾何学的形状を容易に表現可能である。また、陰関数の組み合わせによる分布状態の定義は、表面の混合、変形といった位相変化を伴う処理が堅牢かつ容易であるため、古くから多用されている。複数の陰関数を組み合わせることで、水流や飛沫等の流体表現 [10] や、人体や動物といった形状を表現可能である [11][12]。しかし表示物体が持つ透明度や濃度値は表示に反映されることはなく、対象の表面を表示する事が目的である。複数変数の連続実関数による FRep オブジェクト [13] を使用した表現では形状の他に、形状同士の加算や減算といったオペレータや、形状同士の関係性をも関数表現を用いて行う。FRep オブジェクトを元にオペレータや関係性に独自性をもたせ、濃度値をオブジェクト表面に反映させることで透過面や濃淡といった表現が可能である [14][15][16]。しかし任意の空間中においてエネルギーが分布している様子を表すことは困難である。

本研究ではエネルギー波表現に特化した、解析的線積分可能な分布関数を提案する。提案手法では任意視点からエネルギー波の投影面へ視線を向け、その視線上のエネルギーの 3 次元分布関数を、原始関数が全て初等関数で表現できるものとする。数値積分と比較し、状況に応じた適切なサンプリングポイントの探索が必要なく、初等関数の演算処理のみを用いることで、処理コストが低い。分布関数の計算処理には GPU を汎用計算に利用する GPGPU(General Purpose computing on GPU) を用いて更なる高速化を図った。提案手法は従来のボリュームデータ表現や陰関数表現に比べ、表現できる形状は限定的になるが、エネルギー波の光の強さや形状変形操作を、任意視点から見ても高速且つ正確に表現可能とした。本稿は第 25 回 NICOGRAPH 論文コンテストにおいて発表した内容を含む [17]。

本稿ではまずエネルギー波形状の生成とレンダリング方法を述べる。その後エネルギー波の移動と形状変形手法、GPGPU による処理の高速化に関して述べ、レンダリング結果の表示と考察を行い、新たなエネルギー波表現としての可能性を示す。付録には本手法で行った積分計算式の詳細を記載する。

2 エネルギー波形状の生成

本手法では、エネルギー波形状を規定するエネルギーの 3 次元分布を 2 つの関数を組み合わせて行う。3 次元空間中の任意の位置ベクトル $\mathbf{P}$ に対して、球を表す関数 $S(\mathbf{P})$ と円柱を表す関数 $C(\mathbf{P})$ を組み合わせた関数 $E(\mathbf{P})$ を定義する。

$$E(\mathbf{P}) = S(\mathbf{P}) + C(\mathbf{P}) \tag{1}$$

$S(\mathbf{P})$ は球の中心地点での密度が高くなり中心地点から距離が離れるにつれ密度が低くなる関数であり、任意のパラメータ a によって密度の度合いを操作できる。任意の球体の中心座標を $\mathbf{M}$ とする。以下の式 (2) は球を表す関数式である。

$$S(\mathbf{P}) = \frac{a}{|\mathbf{P} - \mathbf{M}|} \tag{2}$$

$C(\mathbf{P})$ は円柱の中心線での密度が高くなり中心線から距離が離れるにつれ密度が低くなる関数であり、任意のパラメータ b によって密度の度合いを操作できる。円柱の中心線が延びる任意の方向ベクトルを $\mathbf{D}$ とし、$\mathbf{D}$ は単位ベクトルとする。中心線が延び始める地点は $S(\mathbf{P})$ で用いた $\mathbf{M}$ とする。以下の式 (3) は円柱を表す関数式である。

$$C(\mathbf{P}) = \frac{b}{\sqrt{|\mathbf{P}-\mathbf{M}|^2 - ((\mathbf{P}-\mathbf{M})\cdot\mathbf{D})^2}} \qquad (3)$$

また本手法では、点 $\mathbf{M}$ を含み $\mathbf{D}$ を法線とする平面を I とし、I を境に法線方向区間では式 (3) を、法線の逆方向区間では式 (2) を用いる事で円柱形状を表現する。I においてパラメータ a と b の値を一致させることで、全ての空間中で密度分布が連続になる。図 1 は提案手法で生成するエネルギー波の模式図である。

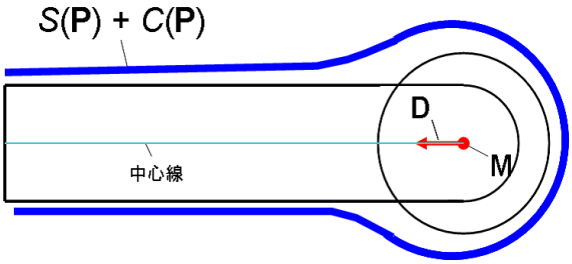


図 1: エネルギー波形状の模式図

3 レンダリング手法

本研究の提案手法はボリュームレンダリング手法の 1 つであるレイキャスティング法と基本的な概念は同様である。レイキャスティング法とは視点の位置から描画面に対して視線を飛ばし、その視線方向に沿ってボリュームデータを数値積分していく手法である。描画面のピクセル毎に同様の処理を繰り返し行う事で、最終的な描画結果を得る。図 2 はレイキャスティング法における視線と描画面、ボリュームデータの関係を図示したものである。

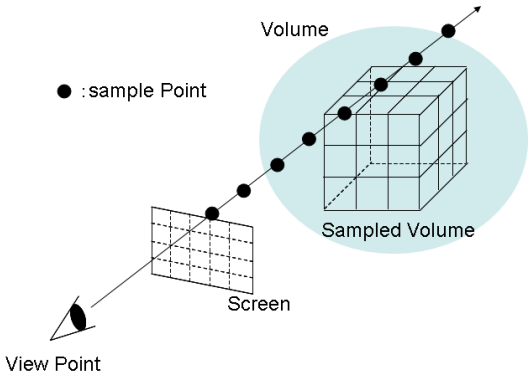


図 2: レイキャスティング法の模式図

提案手法では 3 次元分布状況の近似値であるボリュームデータを生成することなく、視線に沿ったエネルギーの分布関数を、線積分式となるものを用いることで、最終的な描画結果を得る。図 3 は 1 本の視線における 3 次元分布状況の計算結果の違いを比較している。

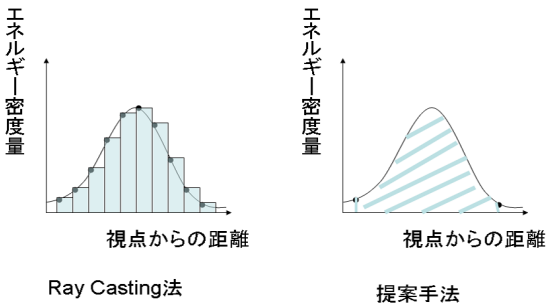


図 3: 積分値の比較

提案手法の手順は以下である。

1. 3 次元空間中にエネルギーの分布状況を規定
2. 積分区間の決定
3. 積分区間中のエネルギー密度の度合いを算出
4. 隠面消去処理
5. エネルギー波の表示

以下に個々の処理について述べる。

3.1 エネルギー分布状況の規定

エネルギーの分布は式 (1) を用いて規定する。

3.2 積分区間の決定

視点の位置ベクトルを $\mathbf{S}$、視点から描画面に対して視線を飛ばし、視点からみて十分遠い距離にある視線上の点を $\mathbf{E}$ とする。$\mathbf{S}$ と $\mathbf{E}$ の範囲内が積分区間となる。$\mathbf{E}$ の位置は視線上の任意地点である図 4 は積分区間と描画面の位置関係を表している。

3.3 積分区間中のエネルギー密度の度合いを算出

位置ベクトル $\mathbf{S}$、$\mathbf{E}$ に対し、この 2 点を結ぶ直線を SE とする。直線 SE 上の任意の点 $\mathbf{R}$ を媒介変数 t を用いて式 (4) のように表す。t は実数とする。

$$\mathbf{R}(t) = (\mathbf{E}-\mathbf{S})t + \mathbf{S} \qquad (4)$$
$$\text{ただし} \quad 0 \le t \le 1$$

このとき $\mathbf{E}-\mathbf{S}$ を $\mathbf{Q}$ とし、式 (5) のように示す。

$$\mathbf{R}(t) = \mathbf{Q}t - \mathbf{S} \qquad (5)$$

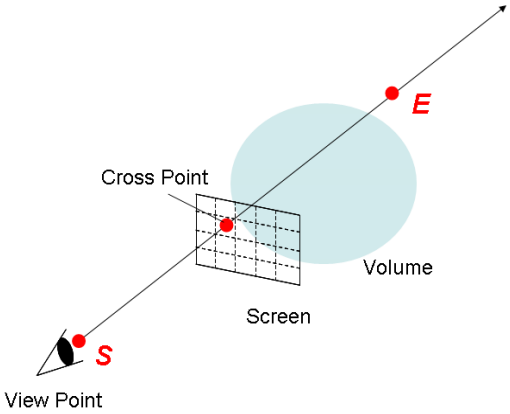


図 4: 積分区間と描画面の位置関係

$\mathbf{R}(t)$ の各成分を式 (2)、式 (3) に代入する。

$$S(t) = \frac{a}{\sqrt{U(t)}} \tag{6}$$
$$U(t) = |\mathbf{R}(t) - \mathbf{M}|^2$$

$$C(t) = \frac{b}{\sqrt{V(t)}} \tag{7}$$
$$V(t) = U(t) - ((\mathbf{R}(t) - \mathbf{M}) \cdot \mathbf{D})^2$$

式 (6)、式 (7) のうち、$U(t)$、$V(t)$ は t の 2 次式といえる。これを $U(t) = At^2 + 2Bt + C$、$V(t) = Dt^2 + 2Et + F$ とすると、$U(t)$、$V(t)$ は常に正の値をとることが保証できる事から、$S(t)$、$C(t)$ の線積分式はそれぞれ式 (8)、式 (9) で求まる。

$$\int_0^1 S(t)dt = \left[\frac{a \sinh^{-1}\left(\frac{2(tA+B)}{4AC-(2B)^2}\right)}{\sqrt{A}} \right]_0^1 \tag{8}$$
ただし $A \neq 0$
$4AC - (2B)^2 \neq 0$

$$\int_0^1 C(t)dt = \left[\frac{b \sinh^{-1}\left(\frac{2(tD+E)}{4DF-(2E)^2}\right)}{\sqrt{D}} \right]_0^1 \tag{9}$$
ただし $D \neq 0$
$4DF - (2E)^2 \neq 0$

本手法で用いる円柱形状の密度分布は、平面 I を境目とした式 (2) と式 (3) の合計値である。式 (2) を用いる区間と式 (3) を用いる区間の調整は、線分 SE と平面 I の交点を利用する。SE と I の交点を $\mathbf{J}$ とし、$\mathbf{J}$ における媒介変数 t の値を以下の式 (10) で求める。

$$t = \frac{(\mathbf{D} \cdot \mathbf{M}) - (\mathbf{D} \cdot \mathbf{S})}{\mathbf{D} \cdot \mathbf{Q}} \tag{10}$$

t を積分区間の境目とし、式 (2) を用いた積分区間と式 (3) を用いた積分区間を足し合わせる。円柱形状を表す密度分布は以下の式 (11) とする。

$$\int_l^k C(t)dt + \int_n^m S(t)dt \tag{11}$$

積分区間を表すパラメータ k、l、m、n は、$\mathbf{S}$ と $\mathbf{D}$ により決定する。$\mathbf{S}$ が I を境に $\mathbf{D}$ 方向にある場合、$(k, l, m, n) = (t, 0, 1, t)$ となる。$\mathbf{S}$ が I を境に $\mathbf{D}$ とは逆方向にある場合、$(k, l, m, n) = (1, t, t, 0)$ となる。図 5 は積分式を平面 I で切り替えることで表現をした円柱形状の模式図である。

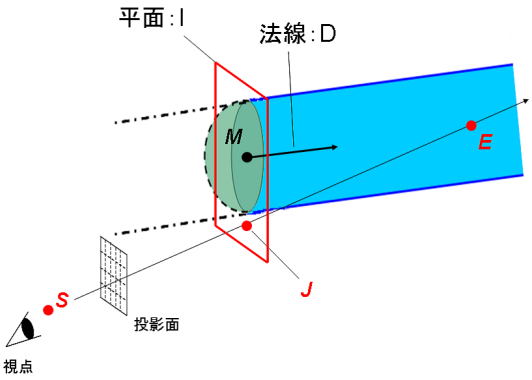


図 5: 積分区間の制御を行った円柱形状

最後に式 (8) と式 (11) を式 (12) のように足し合わせることで、一本の視線におけるエネルギー密度の度合い H が算出できる。

$$H = |\mathbf{Q}| \left(\int_0^1 S(t)dt + \int_l^k C(t)dt + \int_n^m S(t)dt \right) \tag{12}$$

3.4 隠面消去処理

エネルギー波に隠面処理を施すことで、一般的なオブジェクトとの前後関係を表現し、エネルギー波としての有用性を高める。3DCG 空間をレンダリングする場合、投影面の各画素は色値と同時に、その色値が 3DCG 空間中において、カメラから見てどの位置にあるかという位置情報を持つ。この位置情報は一般的にデプス値と呼ばれる。3DCG では視点からは他の物体で影になっていたり、視線とは逆向きになっている面や線を隠面と呼び、隠面部分を描画しない事で、描

画面上での立体感を高めている。視点から見て、デプス値よりも後ろの空間が隠面部分となり、敢えて描画しない区間となる。エネルギー波も一般的なオブジェクトと同様に、視点から見て隠面部分を描画しない事で、他のオブジェクトとの前後関係を表現する。

初めにエネルギー波以外で用意した全てのオブジェクトを 3DCG 空間中に配置し、この状態で各画素のデプス値を取得する。取得したデプス値を d とし、エネルギー波の隠面消去処理に適応するため、d の値を 0 から 1 の範囲に正規化した値 g を以下の式 (13) で求める。

$$g = \frac{d}{|\mathbf{E} - \mathbf{S}|} \tag{13}$$

次に、各画素における積分区間の終了地点 $\mathbf{E}$ を、式 (13) で求めた値 g を用いて変更する。変更した終了地点を $\mathbf{E}'$ とし、以下の式 (14) のように表す。視点 $\mathbf{S}$ からみて $\mathbf{E}'$ よりも後ろの区間が、エネルギー波における隠面部分となる。

$$\mathbf{E}' = (\mathbf{E} - \mathbf{S})g + \mathbf{S} \tag{14}$$

最後に投影面を視点から視線方向へ一番近い位置に配置することで、エネルギー波と他オブジェクトの前後関係を考慮した表現を実現する。エネルギー波における隠面部分は他オブジェクトを配置した状態のデプス値から求めるため、他オブジェクトの個数に因らず処理は一定である。図 6 は各画素のデプス値によって積分区間を変更する状況の模式図である。

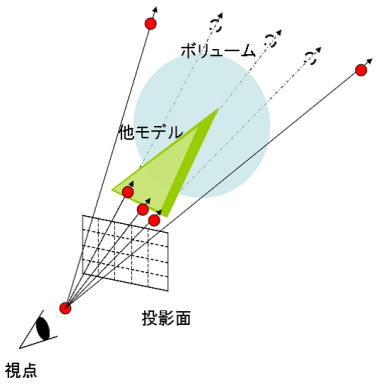


図 6: 各画素のデプス値による積分区間の変更

3.5 エネルギー波の表示

描画面を構成する画素毎に対して視線を飛ばし、以上の手順によるエネルギーの密度分布計算を行うことで、全体の描画結果を得る。

4 エネルギー波の移動と形状変形制御

エネルギー波は形状変化を伴いながらある地点に向かって移動をする。本研究ではエネルギーの分布状況や任意の位置ベクトルを変更することで、エネルギー波の変形と移動を表現する。また、任意点 $\mathbf{M}$ から円柱が延び始める表現は、媒介変数 t を制御することで行う。

球体部分、円柱部分それぞれの移動は、任意の点 $\mathbf{M}$ の位置を変更する事で行う。本手法では球体と円柱に与える任意位置ベクトル $\mathbf{M}$ を同一に設定しているが、個別に設定することも可能である。

エネルギー波の形状変形は、球体における任意のパラメータ a、円柱における任意のパラメータ b をそれぞれ変更する事で表現する。円柱は延びる方向ベクトル $\mathbf{D}$ も任意であるため、$\mathbf{D}$ を変更する事でも変形表現が可能である。パラメータ a を増加させる事で、$\mathbf{M}$ を中心とする球体の大きさを大きく、パラメータ b を増加させる事で、円柱の太さを太くする事が可能である。パラメータ a、b を動的に変更することで、エネルギー波形状の動的な制御を行う。

5 GPGPU を利用した実装

本研究では、GPU を汎用計算機として利用する GPGPU(General Purpose Computing on GPU) を用いて処理速度の向上を図った。通常 GPU は画像処理を専門に行うユニットであるが、近年では補助演算装置として利用する働きがある。GPGPU は CPU と比べ分岐処理や再帰処理は不得意であるが、単純なデータを 1 度に大量に扱う並列処理は得意である。本手法において、エネルギー波の投影面を生成する処理は、1 画素につき式 (12) を 1 度計算することになる。この処理を画素毎に繰り返すことで、最終的な描画面全体を得る。本手法では GPGPU の特性に合わせ、式 (12) を 1 つの処理と見なし、複数の画素値を 1 度に並列計算することで全体としての処理速度を向上させている。図 7 は描画結果の計算処理において、CUP 処理及び、GPGPU 処理の流れを比較した模式図である。GPGPU では汎用計算に必要なデータを CPU から GPU へ転送し、GPU 内での計算結果を CPU へ再び転送するといった CPU・GPU 間のデータ転送を必要とする。このデータ転送は全体の処理時間においてボトルネックになりやすく、複数回に分けるよりも 1 度の転送で大量のデータをやり取りし、データ転送の回数を減らす事で全体としての処理を高速に行うことが可能となる。

6 動作検証

これまでに述べた手法に基づき、PC 上での実装を行った。グラフィクス API は OpenGL を使用し、GPGPU には NVIDIA 社 CUDA[18] を用いて実装

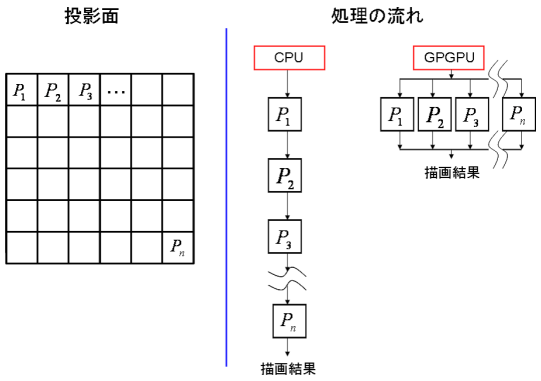


図 7: CPU と GPGPU の処理の流れ比較

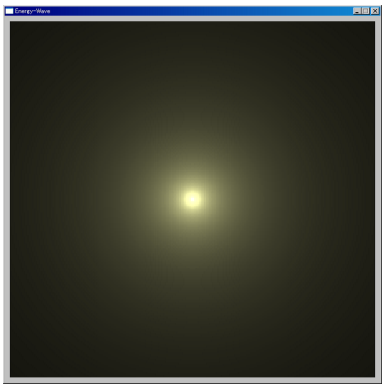


図 8: 実行結果：球体

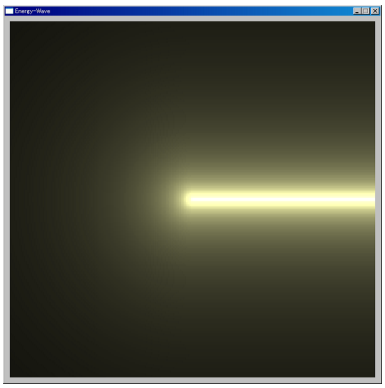


図 9: 実行結果：円柱

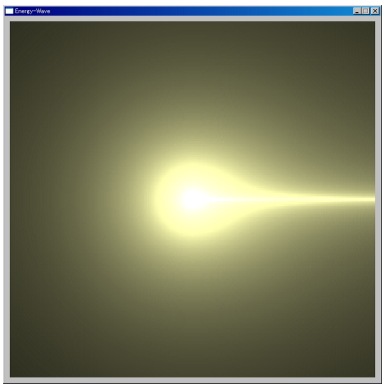


図 10: 実行結果：球体＋円柱

した。以下の表 1 に、今回の検証で使用した環境を示す。

表 1: 検証に使用した環境

CPU	Intel(R) Core(TM)2 Duo 3.00GHz
RAM	2GB
GPU	NVIDIA GeForce 9600 GT

この環境下において、描画面を 256×256 画素、512×512 画素の 2 つの異なる解像度とし、隠面消去処理を行った場合と行っていない場合の処理速度の検証を行った。以下の表 2 に検証結果を示す。処理速度はエネルギー波を描画し始めて 10 秒後から 20 秒後の 10 秒間の平均値とした。

表 2: 異なる解像度における処理速度の検証結果

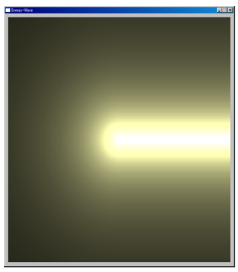
	隠面処理　無し	隠面処理　有り
256×256 画素	115FPS	105FPS
512×512 画素	31FPS	27FPS

図 8、図 9、図 10 はプログラムの実行結果である。任意の方向ベクトル $\mathbf{D}$ の値は $(1.0, 0.0, 0.0)$ とし、視点の位置はエネルギー波の真横とした。各視線上の点 $\mathbf{E}$ は、描画面の 1 辺の長さを 1 とした場合、視点からみて 500 の長さの位置とした。図 11、図 12 は球体と円柱のパラメータを比較した結果である。図 13 はエネルギー波の移動の様子を示し、図 14 は形状変形の様子を示す。

次に隠面消去処理を施した場合と施さなかった場合の比較画像を示す。エネルギー波のパラメータをそれぞれ、パラメータ $a = 60$、パラメータ $b = 30$ と


パラメータ a：60　　パラメータ a：90

図 11: 実行結果：球　パラメータ比較

パラメータ b：60　　パラメータ b：90

図 12: 実行結果：円柱　パラメータ比較


図 13: 実行結果：移動制御


パラメータ a：60　　パラメータ a：60
パラメータ b：30　　パラメータ b：50

図 14: 実行結果：変形制御

し、任意点 $\mathbf{M}$ の z 成分のパラメータ m_z を動的に変更した。エネルギー波以外のオブジェクトとして四面体形状を 2 つ配置した。配置したオブジェクトの位置はそれぞれ $(0.0, 0.0, 0.0)$、$(10.0, 0.0, 250.0)$ である。図 15 は $m_z = 280$ のときの実行結果である。画像左が隠面消去処理を施さなかった場合、画像右が施した場合の実行結果である。図 16 は $m_z = 70$ のときの実行結果である。図 17 は $m_z = 10$ のときの実行結果である。図 18 は $m_z = -140$ のときの実行結果である。

GPGPU を用いて処理を行った場合と、用いずに処理を行った場合との処理速度を比較する。描画面を 512×512 画素、陰面処理を施している状態とした。以下の表 3 に検証結果を示す。処理速度はエネルギー波を描画し始めて 10 秒後から 20 秒後の 10 秒間の平均値とした。

表 3: GPGPU を用いた処理速度の検証

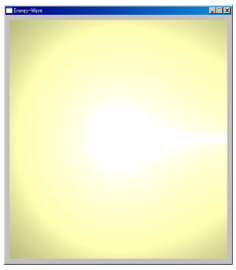
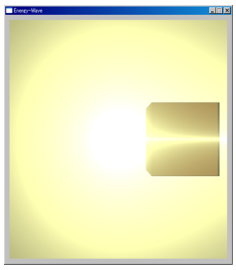
GPGPU 有り	27FPS
GPGPU 無し	1FPS

形状変形や移動処理にかかった時間も極わずかであり、変形や移動処理をしない場合に比べても、描画処理時間に大きな差は見られなかった。よって本手法ではリアルタイム性を確保しつつ、任意視点から見た正確なエネルギーの光の強さ表現や形状変形表現が可能であることが確認できた。

7　考察とまとめ

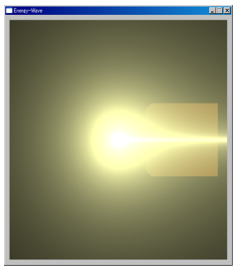
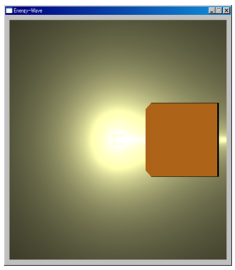
本研究ではエネルギー波表現における新しい表現手法を提案した。不定形な現象表現に適したボリュームデータの概念を踏襲しつつ、エネルギー波表現に特化した、解析的線積分可能なエネルギー分布関数を用いた。エネルギー波形状は、ボリュームデータを用いた場合に比べ限定的であるが、任意視点からでも正確且つ高速に 3 次元分布状況を算出可能となった。分布状況算出には GPGPU を利用することで、提案した分布関数の計算を更に高速に行った。提案する分布関数に対し、任意のパラメータ群を与えることにより、エネルギー波の移動や変形表現を実現した。エネルギー波に隠面消去処理を施すことで、同一空間内における他オブジェクトとの違和感のない前後関係を表現した。

今後の課題として、エネルギー波形状の追加が挙げられる。本稿では基本形状として球体関数と円柱関数を定義し、それぞれの値を加算することでエネルギー波形状を生成した。基本形状を表す分布関数や、減算や論理和、論理積といった関数値に対する演算処理を増やすことで、様々な形状を追加可能である。組み合わせる関数の数を増やすことで、局所的な形状変形も実現可能であると考える。また、他オブジェクトに与える影響が挙げられる。エネルギー波は強く発光する現象ではあるが、現状では光源としての効果はほぼ無

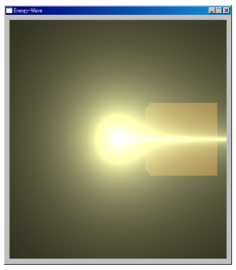
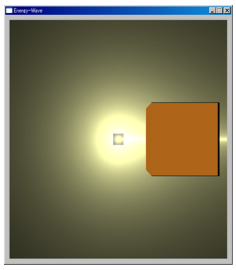
隠面消去処理：無し　　隠面消去処理：有り

図 15: 実行結果：$m_z = 280$ の実行結果

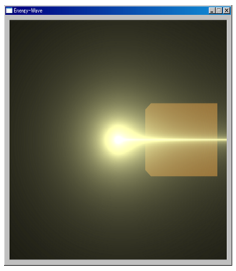
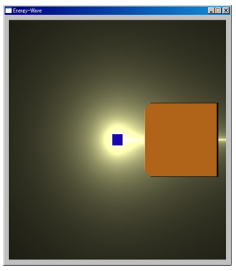
隠面消去処理：無し　　隠面消去処理：有り

図 16: 実行結果：$m_z = 70$ の実行結果

隠面消去処理：無し　　隠面消去処理：有り

図 17: 実行結果：$m_z = 10$ の実行結果

隠面消去処理：無し　　隠面消去処理：有り

図 18: 実行結果：$m_z = -140$ の実行結果

い。またエネルギー波と他オブジェクトが衝突した場合、エネルギー波もしくは他オブジェクトになんらかしらの変化を与えることで、よりエネルギー波の表現を高めることが出来るであろう。

参考文献

[1] CAPCON. STREET FIGHTER　. <http://www.capcom.co.jp/sf4/>.

[2] Rovert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume Rendering. Computer Graphics 22, 4, SIGGRAPH'88, pp.65-74, 1988.

[3] M. Groher, F. Bender, R.-T. Hoffmann, and N. Navab. Segmentation-driven 2D-3D registration for abdominal catheter interventions. Proceedings of the 10th international conference on MICCAI, 2007.

[4] K. Zhou, Z. Ren, S. Lin, H. Bao, B. Guo, and H.-Y. Shum. Real-Time Smoke Rendering Using Compensated Ray Marching. ACM Transactions on Graphics 27, 3(Aug), 36:1-12, 2008.

[5] H. Tuy and L. Tuy, Direct 2d display of 3d objects, IEEE mag. Computer Graphics and Applications, 1984.

[6] J. kruger, R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. Proceedings of the 14th IEEE Visualization, 2003.

[7] 高棹大樹, 金井崇, 山口泰. GPU を用いた高品質ボリュームレンダリングに関する研究. 情報処理学会研究報告, 2007(13), pp.67-72, 2007

[8] J.F. Blinn. A generalization of algebraic surface drawing. ACM Trans. Graphics, 1(3):235-256, 1982.

[9] H. Nishimura, M. Hirai, T. Kawai, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. Journal of papers given by at the Electronics Communication Conference, J68-D(4):718-275, 1985.

[10] Y. Kanamori, Z. Szego and T. Nishita. GPU-based Fast Ray Casting for a large Number of Metaballs. In Proc. Eurographics, 2008.

[11] Y. Ohtake, A. G. Belyaev, M. Alexa, G. Turk, and H.-P.Seidel. Multi-level partition of unity implicits. ACM Transactions on Graphics (Proc. SIGGRAPH 2003), 22(3):463-470, 2003.

[12] T. Kanai, Y. Ohtake, H. Kawai, and Kiwamu Kase. GPU-based Rendering of Sparse Low-degree IMplicit Surface. In Proceedings of GRAPHITE 2006, pages 165-171, 2006.

[13] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modelling: concept, implementation and applications. The Visual Computer 11, 8, 429-446, 1995.

[14] A. Pasko, V. Adzhiev, B. Schmitt, and C. Schlick. Constructive hypervolume modeling. Graphical Models, v.63 n.6, p.413-442, Nov 2001.

[15] B. Schmitt, A. Pasko, V. Adzhiev, and C. Schlick. Constructive Texturing Based on Hypervolume Modeling. Journal of Visualization and Computer Animation, 12(5), pp. 297-310, 2002.

[16] B. Schmitt, A. Pasko, and C. Schlick. Constructive hypervolume modeling using extended space mappings. Heterogeneous objects modelling and applications: collection of papers on foundations and practice, Springer-Verlag, Berlin, 2008.

[17] 阿部雅樹, 渡辺大地. エネルギー波表現のリアルタイムレンダリング. 芸術科学会 第 25 回 NICOGRAPH 論文コンテスト, 2009.

[18] NVIDIA CUDA. <http://www.nvidia.com/object/cuda_home_new.html>

A 付録

ベクトル成分については以下のものとする。

$$\mathbf{S} = (s_x, s_y, s_z)$$
$$\mathbf{M} = (m_x, m_y, m_z)$$
$$\mathbf{D} = (d_x, d_y, d_z)$$
$$\mathbf{Q} = (q_x, q_y, q_z)$$

式 (8) の A、B、C はそれぞれ以下のように求まる。

$$A = q_x^2 + q_y^2 + q_z^2$$

$$B = q_x s_x + q_y s_y + q_z s_z - q_x m_x - q_y m_y - q_z m_z$$

$$C = (s_x - m_x)^2 + (s_y - m_y)^2 + (s_z - m_z)^2$$

式 (9) の D、E、F はそれぞれ以下のように求まる。

$$\begin{aligned} D =& (1 - d_x^2)q_x^2 + (1 - d_y^2)q_y^2 + (1 - d_z^2)q_z^2 \\ & - 2(d_x d_y q_x q_y + d_x d_z q_x q_z + d_y d_z q_y q_z) \end{aligned}$$

$$\begin{aligned} E =& m_x d_x d_x q_x + m_x d_x d_y q_y + m_x d_x d_z q_z \\ & + m_y d_x d_y q_x + m_y d_y d_y q_y + m_y d_y d_z q_z \\ & + m_z d_x d_z q_x + m_z d_y d_z q_y + m_z d_z d_z q_z \\ & - d_x d_x q_x s_x - d_x d_y q_y s_x - d_x d_z q_z s_x \\ & - d_x d_y q_x s_y - d_y d_y q_y s_y - d_y d_z q_z s_y \\ & - d_x d_z q_x s_z - d_y d_z q_y s_z - d_z d_z q_z s_z \\ & - m_x q_x - m_y q_y - m_z q_z \\ & + q_x s_x + q_y s_y + q_z s_z \end{aligned}$$

$$\begin{aligned} F =& 2(m_x d_x d_x s_x + m_x d_x d_y s_y + m_x d_x d_z s_z \\ & + m_y d_x d_y s_x + m_y d_y d_y s_y + m_y d_y d_z s_z \\ & + m_z d_x d_z s_x + m_z d_y d_z s_y + m_z d_z d_z s_z \\ & - d_x d_y s_x s_y - d_y d_z s_y s_z - d_x d_z s_x s_z \\ & - m_x m_y d_x d_y - m_y m_z d_y d_z - m_x m_z d_x d_z \\ & - m_x s_x - m_y s_y - m_z s_z) \\ & + m_x^2 + m_y^2 + m_z^2 + s_x^2 + s_y^2 + s_z^2 \\ & - m_x^2 d_x^2 - m_y^2 d_y^2 - m_z^2 d_z^2 - d_x^2 s_x^2 - d_y^2 s_y^2 - d_z^2 s_z^2 \end{aligned}$$

阿部 雅樹

2008 年東京工科大学メディア学部卒業. 2010 年東京工科大学大学院バイオ・情報メディア研究科メディアサイエンス専攻卒業.

渡辺 大地

1993 年慶應義塾大学環境情報学部卒業. 1995 年慶應義塾大学政策・メディア研究科修士課程修了. 修士 (政策・メディア). 1999 年より東京工科大学メディア学部講師. コンピューターグラフィックスやゲーム制作に関する研究に従事. 情報処理学会, 芸術科学会会員.