修士論文　　平成 21 年度 (2009)

エネルギー波表現の
リアルタイムレンダリング

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

阿部 雅樹

修士論文　　平成 21 年度 (2009)

エネルギー波表現の
リアルタイムレンダリング

指導教員　渡辺 大地 講師

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

阿部 雅樹

論 文 の 要 旨

論文題目	エネルギー波表現の リアルタイムレンダリング
執筆者氏名	阿部 雅樹
指導教員	渡辺 大地 講師
キーワード	リアルタイム 3DCG、 CG エフェクト レイキャスティング、アニメーション、GPGPU、CUDA

[**要旨**]

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。エネルギーとは創作コンテンツ内の 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。エネルギーの集合でエネルギー波を構成する。

現在 3 次元のビデオゲーム上でエネルギー波を表現する一般的な手法はビルボードテクスチャを用いたものである。ビルボードテクスチャでは事前に複数枚エネルギー波画像を用意し、コンテンツ内の 3 次元空間中に表示する際には、視点に応じて用意した画像を回転・切り替えることで 1 つのエネルギー波を表現する。事前に想定した視点から見えるエネルギー波の光の強さや形状を画像として用意するため、それ以外の視点から見たエネルギー波の光の強さや形状変形を正確に表現できない。3 次元空間中の分布データを可視化する技術にボリュームレンダリングがある。任意空間中の分布データをサンプリングし、2 次元へ投影する一連の流れをボリュームレンダリングと呼ぶ。ボリュームレンダリングを用いてエネルギー波を表現する場合、ビルボードテクスチャと比較して、任意視点からでもエネルギー波の光の強さを正確に描画が可能である。しかしボリュームデータの再サンプリングには時間がかかるため、エネルギー波のリアルタイム形状変形には不向きな技術である。このように 3 次元のビデオゲーム等のインタラクティブコンテンツにおいて、任意視点から見たエネルギー波の光の強さおよび形状変形を、正確にリアルタイムに表現できる手法は確立されていない。

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用いてエネルギー波の 3 次元分布状況を規定することで、ボリュームデータ生成処理を省くことを可能とした。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。

A b s t r a c t

Title	Real-time Rendering of Energy-wave Expression
Author	Masaki Abe
Advisor	Lecturer Taichi Watanabe
Key Words	Realtime 3DCG, CGEffect, Ray-casting, Animation, GPGPU, CUDA

[**summary**]

In recent years, various expressions that use 3DCG on the creation contents of animation and video game are developed. Especially, a lump of the energy strongly emits light and transforms shape. It moves to the target spot in the creation contents of fight and action. It is used as method of attack and special effects in the creation contents. We defined the Energy as an element constituting existing Energy-Wave in the space in creation contents. The Energy distributes in the 3 dimensions space, and the part that density is high emits light. A set of the energy constitutes Energy-Wave.

The present, the Energy-Wave expression technique in the 3 dimensions video games, billboard texture is common. In that technique, a plural Energy-Wave image is prepared beforehand. The image which has been prepared for expresses Energy-Wave by turning and exchanging when it is displayed in the 3 dimensional space in contents. Strength of the light and shape of the Energy-Wave seen from expected viewpoint beforehand are prepared as an image. Therefore it cannot express strength of the light an shape of Energy-Wave precisely from viewpoint that is not expected. Volume Rendering is technique to make distribution data in 3 dimensional space visible. The technique sample distributed data in optional space, and reflect it to a 2 dimensional plane. Volume Rendering is superior to billboard texture in expressing strength of the light of the Energy-Wave from an arbitrary viewpoint. However, it is unsuitable for real-time shape deformation of the Energy-Wave because re-sampling speed of volume data is slow. From these, in interactive contents such as the 3 dimensional video game, the technique that can express strength of the light and shape deformation of the Energy-Wave from an arbitrary viewpoint in real-time precisely is not established.

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 that we generated 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.

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 研究背景

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 3 次元コンピュータグラフィクス (以下 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。代表的な例として、漫画「ドラゴンボール [1]」の作中における気功波、テレビアニメーション「機動戦士ガンダム [2]」の作中におけるビーム攻撃が挙げられる。また映画やテレビ番組など、実写映像における特殊効果としても使用する機会が増え、一般的な表現となりつつある。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。エネルギーとは創作コンテンツ内における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に、ビルボードテクスチャがある。ビルボードテクスチャはエネルギー波を単一の 2 次元画像で事前に複数枚用意する。用意した画像を、ゲーム内で矩形ポリゴンなどの単純な形状に貼り付ける。画像を貼り付けたポリゴンはゲームのプレイヤーの視線に垂直になるよう配置し、プレイヤーの視点変更やエネルギー波のアニメーション等の状況変化に応じて画像を回転・切り替えることで、1 つのエネルギー波を表現している。図 1.1 は矩形ポリゴンにエネルギー波のテクスチャを貼り付け、視点のみ変更する事で、矩形ポリゴンを複数の視点位置から示したものである。ビルボードテクスチャはデータ容量が比較的少なく描画にかかる処理は高速であるが、事前に用意したテクスチャ画像のみでエネルギー波表現を行うため、任意視点から見たエネルギー波の光の強さを正確に表現することは不可能となる。変形表現も同様に、特定の形状のみ表現可能であり、場面変化に対して柔軟な対応をとることは困難である。

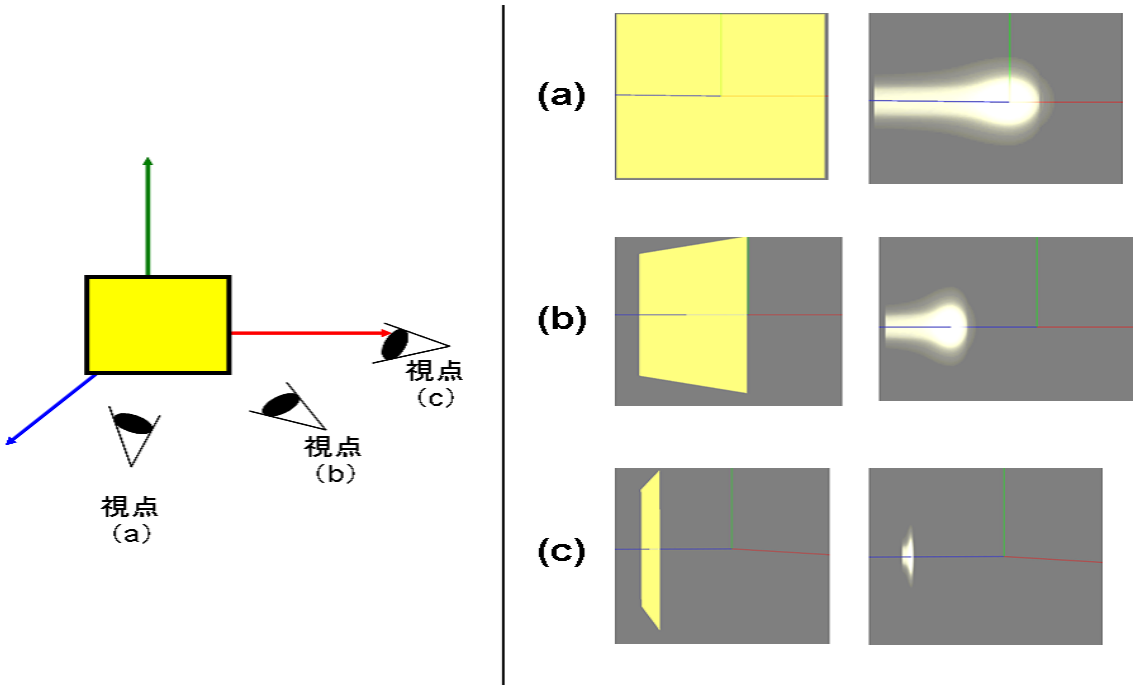


図 1.1: 複数の視点から見たビルボードテクスチャ

3 次元空間中の分布データを可視化する技術にボリュームレンダリング [3][4] がある。ボリュームレンダリングは、人体の内部構造や内容物を含んだ半透明物体、物体の表面が定義できない炎や煙といった不定形自然現象等を描画する技術である。ボリュームレンダリングを行う場合、3 次元空間中の圧力分布や温度分布などを一定の領域で分割し、サンプリングしていく。この一定の領域で区切ったデータ群をボリュームもしくはボリュームデータと呼ぶ。3 次元空間中の温度や密度等の分布状況からボリュームデータを生成し、そのボリュームデータを投影する一連の流れをボリュームレンダリングと呼ぶ。図 1.2 はボリュームデータを 2 次元平面へ投影を行う模式図である。

ボリュームデータは MRI 画像等の実測値を利用する場合や、温度分布や水分等の密度分布の変移を計算する事によって生成する。図 1.3 は球体を表す関数を用いた計算処理によってボリュームデータを生成する様子の模式図である。

ボリュームレンダリングでは大量のデータ処理を必要とし、従来では高速なレンダリングには高価なスーパーコンピュータや専用ハードウェアを用いてきたが

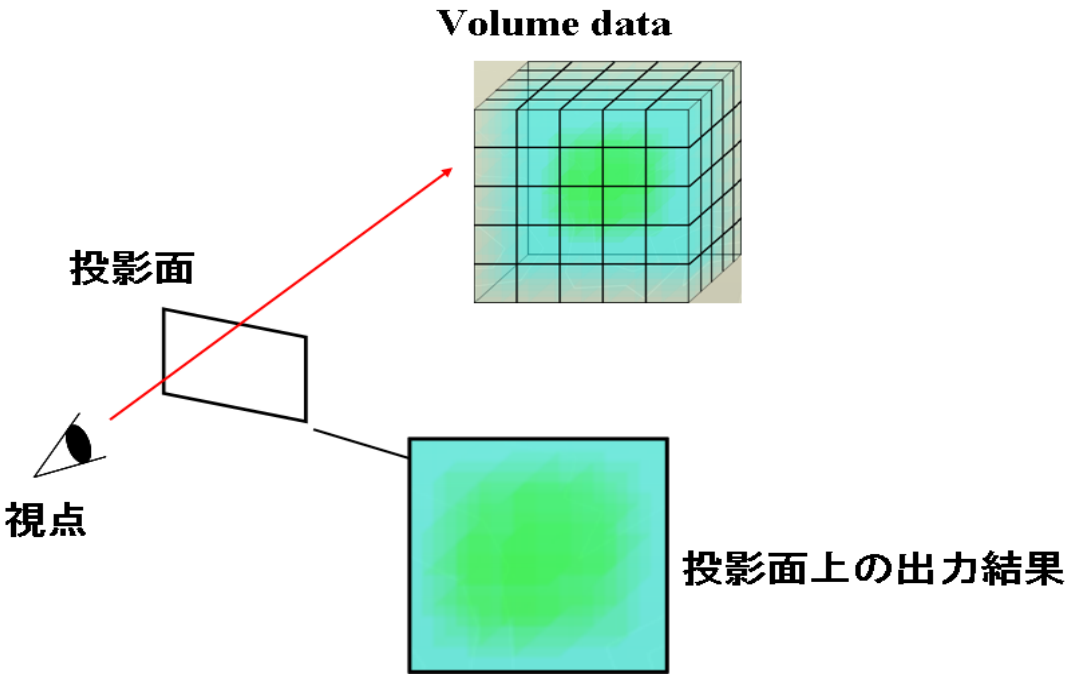


図 1.2: ボリュームデータを 2 次元平面へ投影

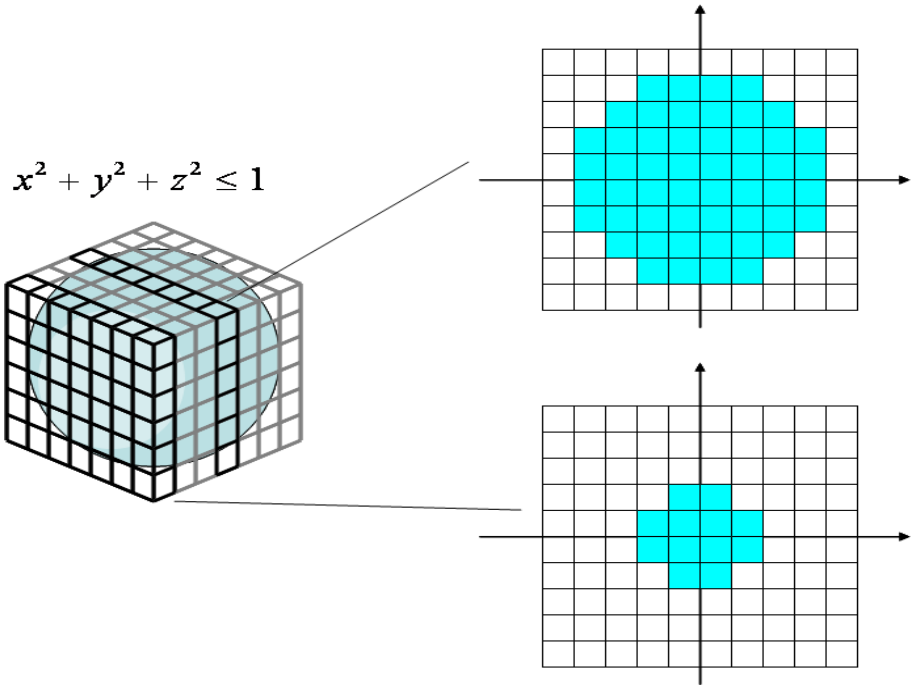


図 1.3: 関数によるボリュームデータの規定

[5][6][7]、近年の PC 性能進化によりソフトウェアでの高速化が主流となってきた。代表的なボリュームレンダリング手法として、レイキャスティング法 [8]、Marching Cubes 法 [9]、テクスチャベース法 [10][11] などが挙げられる。ボリュームレンダリングでは高精細なレンダリング画像を得ようとする場合、元となるボリュームデータも高精細である必要がある。そのためボリュームデータの記憶容量の確保や、再サンプリングの面でリアルタイム性の低下につながる場合がある [11]。3 次元空間中の圧力分布や温度分布の変更、ボリュームデータの生成は非常に計算コストの高い処理となっているため、ボリュームデータの生成は通常 1 度で終了する。ボリュームデータのリアルタイム描画に関する研究では、ボリュームデータを 2 次元平面へ投影するレンダリング部分の高速化を主目的とし、ボリュームレンダリング専用のハードウェアの開発 [12][13][14]、レンダリング処理を並列化する [15][16][17] など、多岐にわたる方法でボリュームレンダリングの高速化は研究されている。また、近年ではグラフィクスハードウェアである GPU(Graphics Processing Unit) の性能進化が目覚しく、GPU を利用してボリュームレンダリングの処理速度の高速化を図るアプローチが多数存在する [18][19][20][21][22][23][24][25]。しかし高速に高精細な描画結果が得られても、状況が限定的であったり、ボリュームデータの動的な変更は考慮しない事が多い。ボリュームレンダリングをエネルギー波表現に用いる場合、任意視点におけるエネルギーの光の強さを正確に表現できるが、ボリュームデータの再サンプリングを伴うエネルギー波の変形表現には不向きである。

そこで本研究は、リアルタイム 3DCG において、3D 空間上にある視点位置の変更や、エネルギー波の形状変形に対応したエネルギー波表現手法を実現することを目的とする。エネルギー波表現は様々な形状で創作コンテンツ内に登場し、演出効果の高い表現として元々需要が高い。柔軟な場面変化に対応できるエネルギー波表現手法の確立は、インタラクティブなコンテンツ内において、更なる演出効果が望める。本研究ではボリュームレンダリングの一種であるレイキャスティング法に着目し、基本的な概念を同様とする新たなレンダリング手法を提案する。エネルギーの密度分布規定には線積分可能な関数を用いることで、ボリュームデータ

生成処理を行うことなくエネルギー分布状況の 2 次元平面への投影を可能とした。計算処理には GPGPU を用いることで更なる処理速度の向上を図り、エネルギー波の形状変形を伴った、任意視点からの正確な光の強さを表現可能とした。関数に与えるパラメータを変更する事で、高速な描画処理を保ったままエネルギー波の移動や変形を可能とした。エネルギー波に隠面消去処理を施すことで、一般的なオブジェクトとの前後判定を表現し、3 次元ビデオゲーム等のコンテンツ内における表現の有用性を高めた。提案手法をプログラム上で実装し、描画速度検証を行い、提案手法の有用性を確認した。

1.2 論文構成

本論文の構成は以下の通りである。第 2 章では、本研究で提案するエネルギー波のレンダリング手法および変形・移動表現について述べる。第 3 章では、コンテンツ内におけるエネルギー波の隠面消去処理につていの提案手法を述べる。第 4 章では、本研究で開発したエネルギー波表現のプログラムにより、その結果の検証と考察を行う。第 5 章では、本研究の成果と意義をまとめ、今後の展望について述べる。

1.3 数式の定義

本論文で用いる数式を以下で定義する。

- $\mathbf{A}\cdot\mathbf{B}$ はベクトル $\mathbf{A}$,$\mathbf{B}$ の内積を表す。
- $|\mathbf{V}|$ はベクトル $\mathbf{V}$ のノルムを表す。
- $F'(x)$ は $F(x)$ の導関数を表す。

第 2 章

提案手法

本章では、本研究で提案するエネルギー波表現手法の手順について述べる。提案手法では初めにボリュームレンダリングと同様に、エネルギー波形状の生成を行った後、レンダリングを行う。提案するレンダリング手法はボリュームレンダリングの一種であるレイキャスティング法と基本的な概念が同等である為、レイキャスティング法の手順を同時に示す。2.1 節ではエネルギー波形状の生成について述べる。2.2 節ではレイキャスティングの基本的なプロセスを述べた後、提案手法のレンダリングについて述べる。2.3 節ではエネルギー波の移動と変形制御について述べる。

2.1 エネルギー波形状の生成

本研究で対象とするエネルギー波は架空の現象であるため、エネルギーの 3 次元密度分布データを計算によって求める。本手法ではエネルギー波形状を複数の関数を用いることで表現する。エネルギーの 3 次元密度分布データは連続関数を用いて定義する。連続関数には線積分可能な関数を用いることで、エネルギーの 3 次元分布状況をボリュームデータとしてサンプリングすることなく、高速且つ正確にエネルギー波のレンダリングが可能となる。アニメーション等の創作コンテンツ内における代表的なエネルギー波形状は、球体と円柱の組み合わせで表現しているため、本手法で用いる関数も球体と円柱形状を表すものとする。図 2.1 は創作コンテンツ内における代表的なエネルギー波形状を示す。

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。

$$E(\mathbf{P}) = S(\mathbf{P}) + C(\mathbf{P}) \tag{2.1}$$

$S(\mathbf{P})$ 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 (2.2) は球を表す関数式である。

図 2.1: 代表的なエネルギー波形状

$$S(\mathbf{P}) = \frac{a}{|\mathbf{P} - \mathbf{M}|} \tag{2.2}$$

$C(\mathbf{P})$ 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 (2.3) は円柱を表す関数式である。

$$C(\mathbf{P}) = \frac{b}{\sqrt{|\mathbf{P} - \mathbf{M}|^2 - ((\mathbf{P} - \mathbf{M}) \cdot \mathbf{D})^2}} \tag{2.3}$$

式 (2.3) で求まる円柱形状は、点 $\mathbf{M}$ を通り $\mathbf{D}$ 方向に延びる線分を中心線とする円柱形状である。図 2.2 は式 (2.3) で求まる円柱形状の模式図である。

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 (2.3) を、法線の逆方向区間では式 (2.2) 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 2.3 は平面 I において球体形状と円柱形状が繋がった状態の模式図である。

図 2.4 は提案手法で生成するエネルギー波の模式図である。

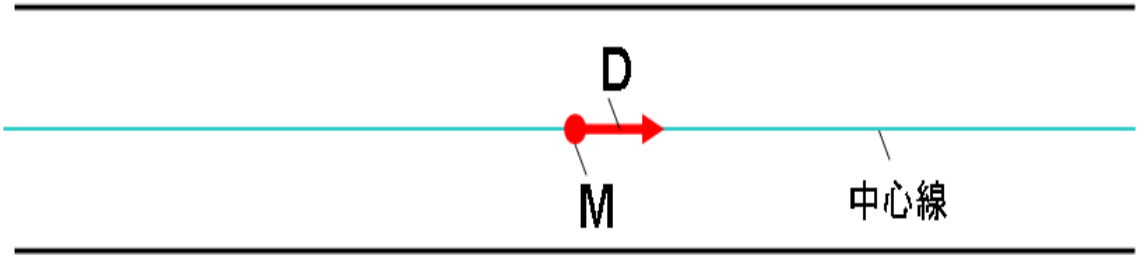


図 2.2: 式 (2.3) の円柱形状

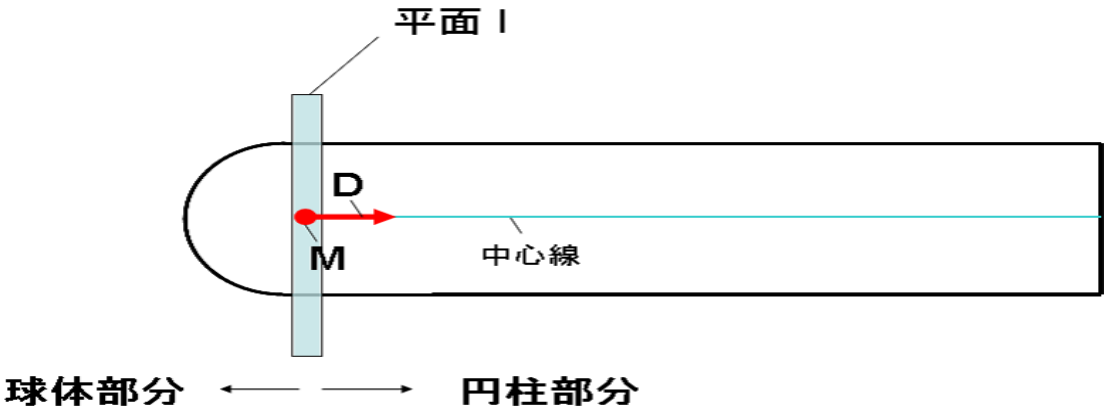


図 2.3: 平面 I で繋がる球体と円柱

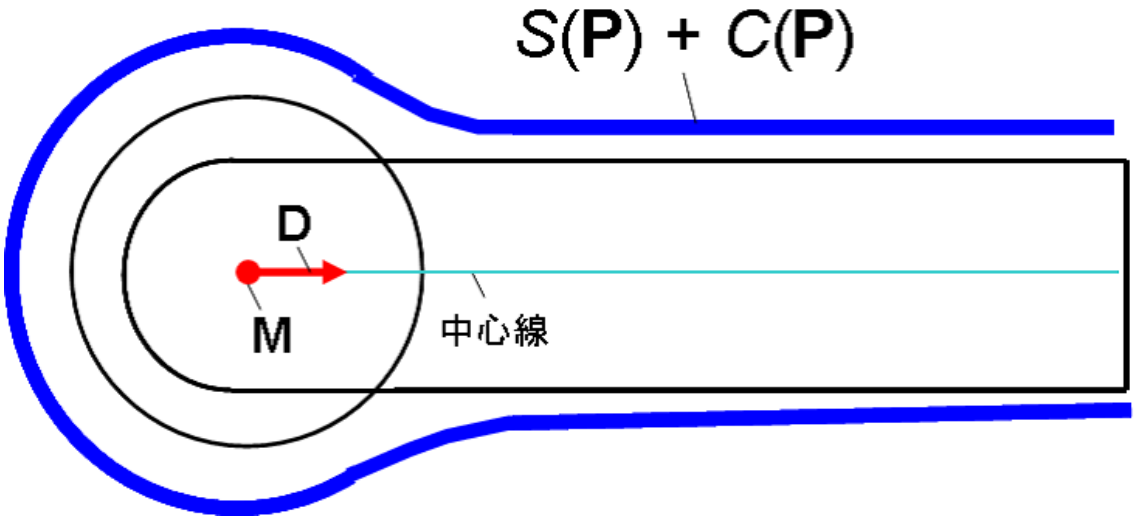


図 2.4: エネルギー波形状の模式図

2.2 レンダリング手法

本研究における、レンダリングの手法について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 3 次元分布状況の近似値であるボリュームデータに対して、視点の位置からボリュームデータを描画する 2 次元平面上の画素毎に視線を飛ばし、その視線に沿ってボリュームデータの分布関数を線積分する手法である。視線に沿った線積分の結果が 1 画素あたりの色値になる。描画面の画素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。この値は数値積分やサンプリング統計処理で求めることが多く、非常に計算コストの高い処理となる。視線上の分布関数におけるサンプリングポイントの数が多ければ高精細な描画結果になる反面、描画にかかる処理速度は低下する。図 2.5 はレイキャスティング法における視線と描画面、サンプリングポイントとボリュームデータの関係を図示したものである。

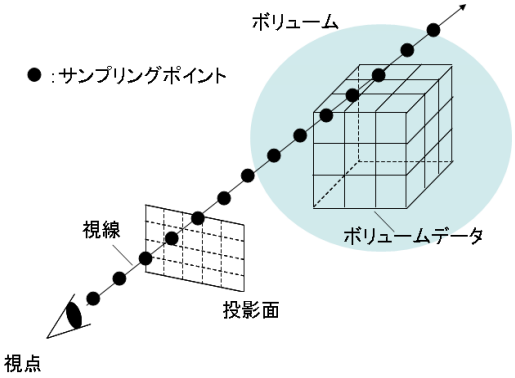


図 2.5: レイキャスティング法の模式図

提案手法では視線に沿ったエネルギーの分布関数を $f(x)$ としたとき、$F'(x) = f(x)$ となる関数 $F(x)$ を初等関数の組み合わせで表現することで、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最終的な描画結果を得る。図 2.6 は 1 本の視線における 3 次元分布状況の計算処理に必要な要素を、図 2.7 は計算結果の違いをそれぞれ比較している。

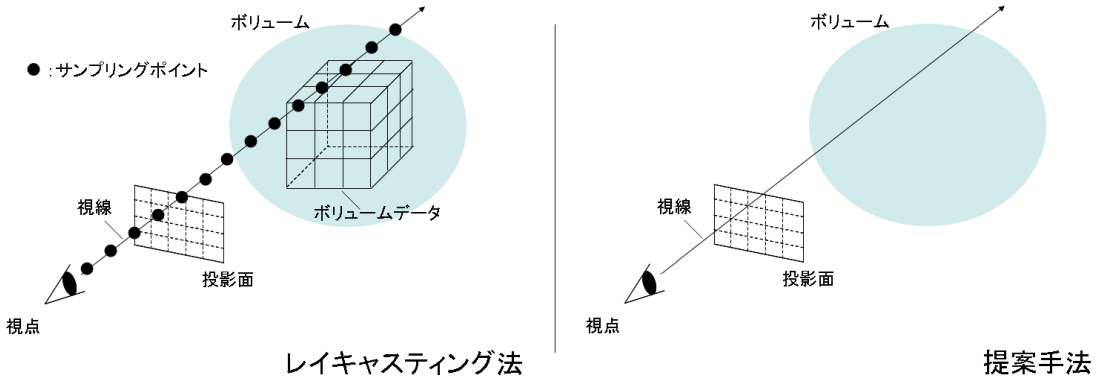


図 2.6: 積分に必要な要素の比較

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

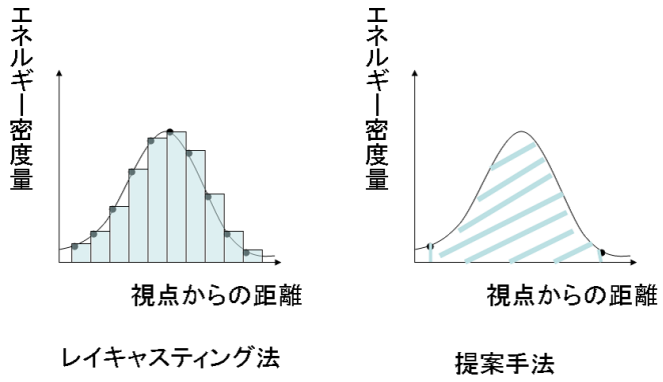


図 2.7: 積分値の比較

4. エネルギー波の表示

以下に個々の処理について述べる。

2.2.1 エネルギー分布状況の規定

エネルギーの分布は式 (2.1) を用いて規定する。

2.2.2 積分区間の決定

視点の位置ベクトルを **S**、視点から描画面に対して視線を飛ばし、視点から視線と描画面の交点までの距離の 2 倍の位置ベクトルを **E** とする。**S** と **E** の範囲内が積分区間となる。図 2.8 は積分区間と描画面の位置関係を表している。

2.2.3 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 **S**、**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 **R** を媒介変数 t を用いて式 (2.4) のように表す。t は実数とする。

$$\mathbf{R}(t) = (\mathbf{E} - \mathbf{S})t + \mathbf{S} \tag{2.4}$$

ただし　$0 \leq t \leq 1$

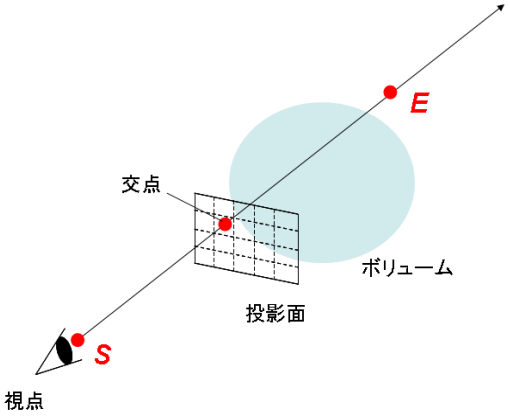


図 2.8: 積分区間と描画面の位置関係

このとき $\mathbf{E} - \mathbf{S}$ を $\mathbf{Q}$ とし、式 (2.5) のように示す。

$$\mathbf{R}(t) = \mathbf{Q}t + \mathbf{S} \tag{2.5}$$

$\mathbf{S}$、$\mathbf{M}$、$\mathbf{D}$、$\mathbf{Q}$ のベクトル成分はそれぞれ以下のものとする。

$$\mathbf{S} = (s_x, s_y, s_z)$$

$$\mathbf{M} = (m_x, m_y, m_z)$$

$$\mathbf{D} = (d_x, d_y, d_z)$$

$$\mathbf{Q} = (q_x, q_y, q_z)$$

$\mathbf{R}(t)$ の各成分を式 (2.2)、式 (2.3) に代入する。

$$S(t) = \frac{a}{\sqrt{U(t)}} \tag{2.6}$$

$$U(t) = |\mathbf{R}(t) - \mathbf{M}|^2$$

$$C(t) = \frac{b}{\sqrt{V(t)}} \tag{2.7}$$

$$V(t) = U(t) - ((\mathbf{R}(t) - \mathbf{M}) \cdot \mathbf{D})^2$$

式 (2.6)、式 (2.7) 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 (2.8)、式 (2.9) で求まる。

$$\int_0^1 S(t)dt = \left[\frac{a \sinh^{-1}\left(\dfrac{2(tA+B)}{4AC-(2B)^2} \right)}{\sqrt{A}} \right]_0^1 \tag{2.8}$$

ただし $A \neq 0$

$$4AC - (2B)^2 \neq 0$$

$$\int_0^1 C(t)dt = \left[\frac{b \sinh^{-1}\left(\dfrac{2(tD+E)}{4DF-(2E)^2} \right)}{\sqrt{D}} \right]_0^1 \tag{2.9}$$

ただし $D \neq 0$

$$4DF - (2E)^2 \neq 0$$

式 (2.8) 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 (2.9) の D、E、F はそれぞれ以下のように求まる。

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

$$
\begin{aligned}
E =&m_xd_xd_xq_x+m_xd_xd_yq_y+m_xd_xd_zq_z \\
&+m_yd_xd_yq_x+m_yd_yd_yq_y+m_yd_yd_zq_z \\
&+m_zd_xd_zq_x+m_zd_yd_zq_y+m_zd_zd_zq_z \\
&-d_xd_xq_xs_x-d_xd_yq_ys_x-d_xd_zq_zs_x \\
&-d_xd_yq_xs_y-d_yd_yq_ys_y-d_yd_zq_zs_y \\
&-d_xd_zq_xs_z-d_yd_zq_ys_z-d_zd_zq_zs_z \\
&-m_xq_x-m_yq_y-m_zq_z \\
&+q_xs_x+q_ys_y+q_zs_z
\end{aligned}
$$

$$
\begin{aligned}
F =&2(m_xd_xd_xs_x+m_xd_xd_ys_y+m_xd_xd_zs_z \\
&+m_yd_xd_ys_x+m_yd_yd_ys_y+m_yd_yd_zs_z \\
&+m_zd_xd_zs_x+m_zd_yd_zs_y+m_zd_zd_zs_z \\
&-d_xd_ys_xs_y-d_yd_zs_ys_z-d_xd_zs_xs_z \\
&-m_xm_yd_xd_y-m_ym_zd_yd_z-m_xm_zd_xd_z \\
&-m_xs_x-m_ys_y-m_zs_z) \\
&+m_x^2+m_y^2+m_z^2+s_x^2+s_y^2+s_z^2 \\
&-m_x^2d_x^2-m_y^2d_y^2-m_z^2d_z^2-d_x^2s_x^2-d_y^2s_y^2-d_z^2s_z^2
\end{aligned}
$$

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 (2.2) と式 (2.3) の合計値である。式 (2.2) を用いる区間と式 (2.3) 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 (2.10) で求める。

$$t = \frac{(\mathbf{D} \cdot \mathbf{M}) - (\mathbf{D} \cdot \mathbf{S})}{\mathbf{D} \cdot \mathbf{Q}} \tag{2.10}$$

t を積分区間の境目とし、式 (2.2) を用いた積分区間と式 (2.3) 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 (2.11) とする。

$$\int_l^k C(t)dt + \int_n^m S(t)dt \tag{2.11}$$

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 2.9 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。

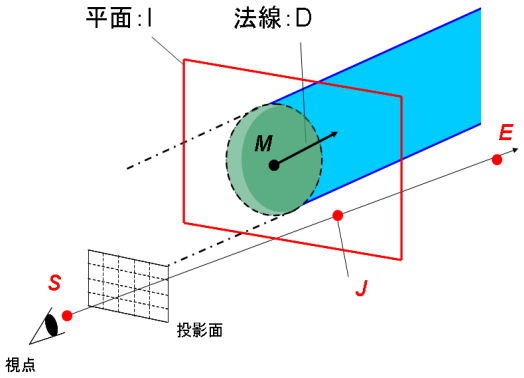


図 2.9: 積分区間の制御を行った円柱形状

最後に式 (2.8) と式 (2.11) を式 (2.12) のように足し合わせることで、1 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。

$$H = |\mathbf{Q}| \left(\int_0^1 S(t)dt + \int_l^k C(t)dt + \int_n^m S(t)dt \right) \tag{2.12}$$

2.2.4 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。

式 (2.12) を用いた密度分布計算を行う場合、球体部分と円柱部分が重複する箇所の密度値 H が極端に高くなり、不自然に輝度値が高くなる。本手法では任意のパラメータ c を閾値とし、H の結果が c を超過した場合には c の値を描画に用いる。輝度値が一定値以上に成らなくすることで、不自然な明るさを失くす。

2.3 移動・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。

第 3 章

エネルギー波の隠面消去処理

本章では、2 章で述べた提案手法に対して、3DCG 空間中におけるエネルギー波以外のモデルとの前後判定を考慮に入れた表現手法を述べる。本手法では、同一空間内におけるエネルギー波以外のモデルを他モデルと呼称する。他モデルはポリゴンモデルとする。2 章で述べた提案手法では、他モデルの存在を考慮していない。その為、エネルギー波と他モデルを同時に空間内に配置した場合、エネルギー波の任意中心座標 M やエネルギー波の投影面、他モデルの位置によっては、それぞれの前後関係を正確に描画することができない。視点から見て順に、任意中心座標 M、他モデル、エネルギー波の投影面という順番で並んでいた場合、エネルギー波は他モデルよりも視点に近い位置に存在することになるが、エネルギー波の投影面が他モデルよりも後ろに存在するため、最終的な描画結果では他モデルの後ろにエネルギー波を描画した状態になる。視点から順にエネルギー波の投影面、他モデル、M と並んだ場合も同様に矛盾が生じる。図 3.1 は M、エネルギー波の投影面、他モデルの前後関係に矛盾が生じている描画結果状態の 1 例を示す。

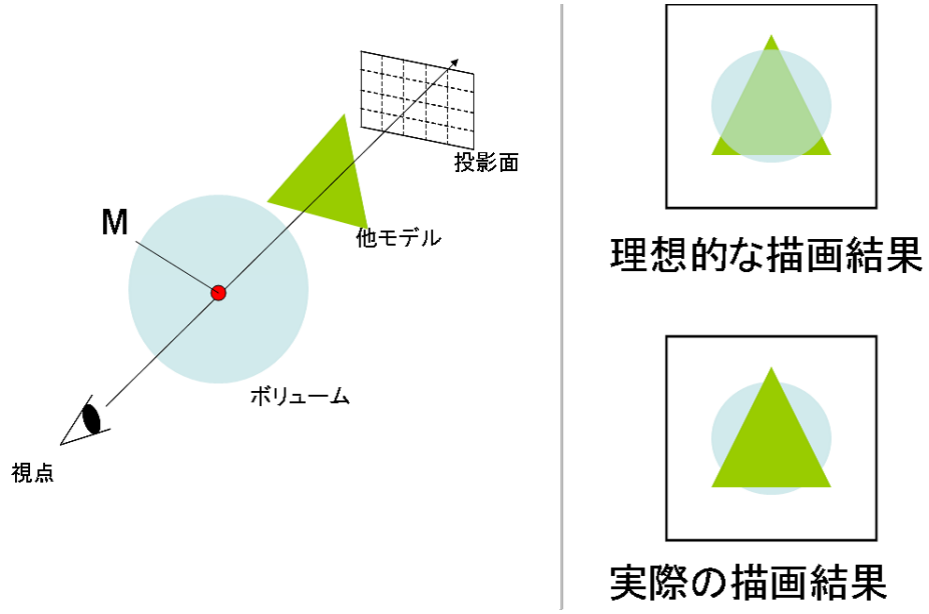


図 3.1: 配置順序と描画結果の矛盾

本手法では視点の位置やエネルギー波の位置は任意としているため、投影面の位置を固定するだけでは描画の際の矛盾を解決できない。本手法では一般的な 3 次

元ビデオゲームにおけるエネルギー波表現のビルボードテクスチャと同様に、視点に 1 番近い位置にエネルギー波の投影面を常に配置すると共に、2.2 節において隠面消去処理を加えることで、エネルギー波と他モデルの前後関係を矛盾することなく描画する。3.1 節では一般的な隠面消去処理で用いる、描画対象の奥行き情報であるデプス値について述べ、3.2 節ではエネルギー波の隠面消去処理について述べる。

3.1　隠面消去処理とデプス値

通常 3DCG では視点からは他の物体で影になっていたり、視点とは逆向きになっている面や線を描画しない事で立体感を高める。この手法を隠面消去法と呼び、z バッファ法、スキャンライン法、レイトレーシング法等が代表的隠面消去法となる。図 3.2 は隠面消去処理を施した描画結果の模式図である。

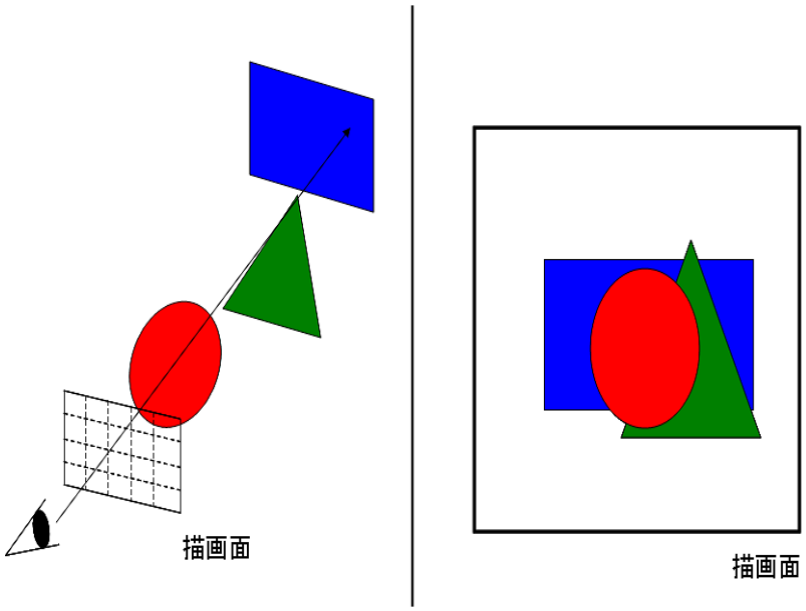


図 3.2: 隠面消去処理による立体感の効果

近年の 3DCG で用いる隠面消去法は z バッファ法が主流である。使用には多量のメモリ領域が必要になるが、ハードウェア性能の向上や、アルゴリズムがハードウェア化に向いている為に広く普及している。z バッファ法では、3DCG 空間内の物体を描画する際、描画面の画素は色値と同時に、その色値が視点からどれだけ離れた位置に存在するかという奥行き情報を持つ。視点から各画素に対して視線を飛ばし、視線と他モデルの交点が 1 箇所以上存在する場合、視点から 1 番近い位置ある交点位置を、その画素が持つ奥行き情報とする。奥行き情報のためのメモリ領域を z バッファ、デプスバッファと呼ぶ。本研究では奥行き情報のためのメモリをデプスバッファ、各画素が持つ奥行き情報をデプス値と呼称する。デプス値はハードウェア側で管理している値であり、任意の画素や状況に応じて取得が可能である [26]。

3.2 エネルギー波における隠面消去処理

エネルギー波に隠面消去処理を施す為には、視線毎におけるデプス値が必要となる。視線上に他モデルとの交点が存在する場合、視点から見て最初の交点よりも後ろの区間は隠面部分となるため、エネルギー波の密度分布計算には必要ない区間となる。2 章では視線毎にエネルギー密度分布を線積分計算することで画素値を決定したが、視線毎の積分区間の終了地点を視線と他モデルとの最初の交点、つまりデプス値とすることでエネルギー波の隠面消去処理が表現可能となる。図 3.3 は視線上に他モデルがあった場合、積分区間を変更する状況の模式図である。

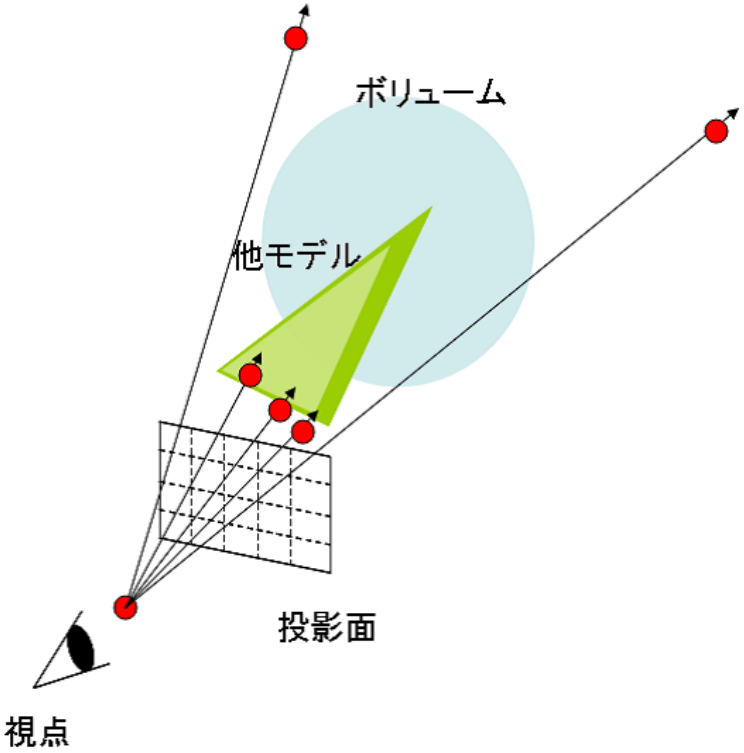


図 3.3: 視線上の他モデルの有無による積分区間の変更

提案手法ではエネルギー波以外のモデルを配置した状態から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 (3.1) で求める。値の正規化には 2 章の式 (2.4) で用いた位置ベクトル $\mathbf{S}$ および $\mathbf{E}$ を利用する。

$$g = \frac{d}{|\mathbf{E} - \mathbf{S}|} \tag{3.1}$$

$\mathbf{S}$ からみて、視線 SE と他モデルの最初の交点位置を $\mathbf{E}'$ とし、g を用いて式 (3.2) のように表す。

$$\mathbf{E}' = (\mathbf{E} - \mathbf{S})g + \mathbf{S} \tag{3.2}$$

積分区間の終了地点を $\mathbf{E}$ から $\mathbf{E}'$ に変更することで、視線毎の隠面部分を消去することが可能となる。最後に投影面を視点から視線方向へ 1 番近い位置に配置することで、エネルギー波と他モデルの前後関係を考慮した表現を実現する。

第 4 章

評価と検証

本章では、本研究で提案したエネルギー波表現手法に沿って実装したプログラムを使用し、その有用性を検証する。本研究で試作したプログラムは、グラフィクス API の OpenGL をベースとした 3 次元グラフィックスツールキットである「FK Kernel Tool Kit System」[27][28]、計算処理の高速化を図った GPGPU には、NVIDA 社の CUDA[29] を用いて実装した。

4.1 実行結果

本提案手法の実行結果を以下に示す。図 4.1 は球体形状のみ、図 4.2 円柱形状、図 4.3 は球体形状と円柱形状の組み合わせを示した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 $\mathbf{S}$ は $(0.0, 0.0, 300.0)$ とした。

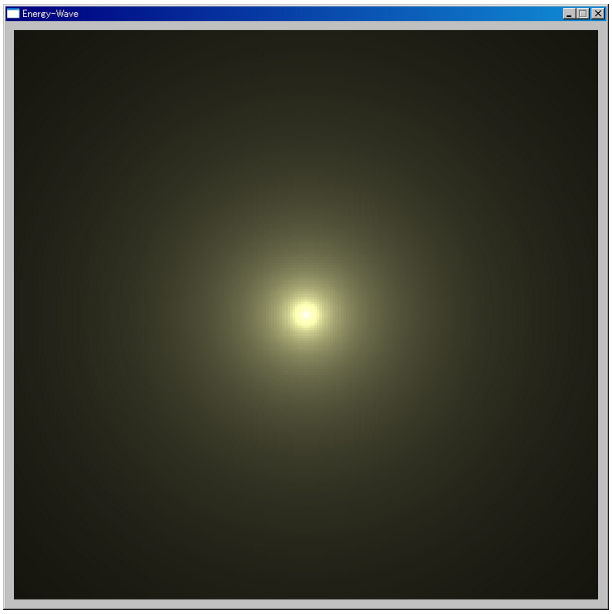


図 4.1: 実行結果：球体

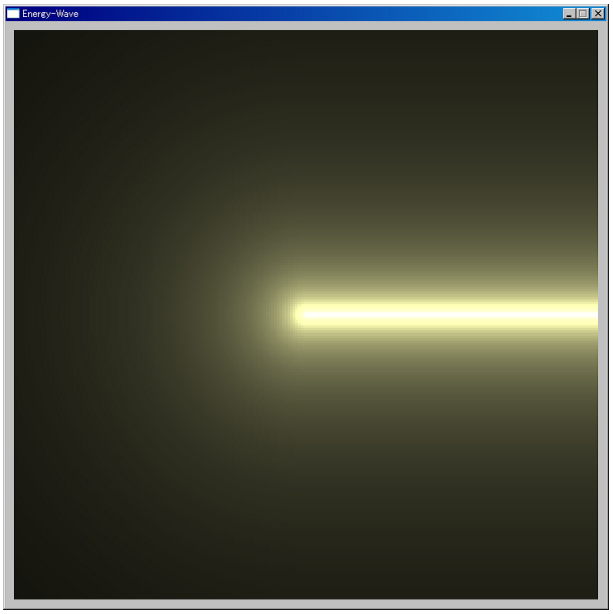


図 4.2: 実行結果：円柱

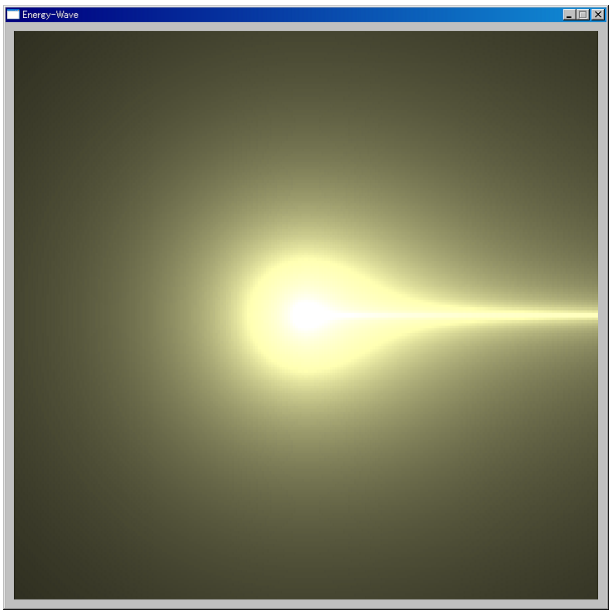
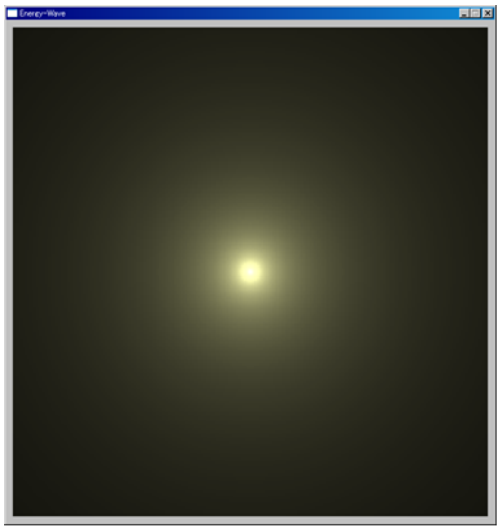
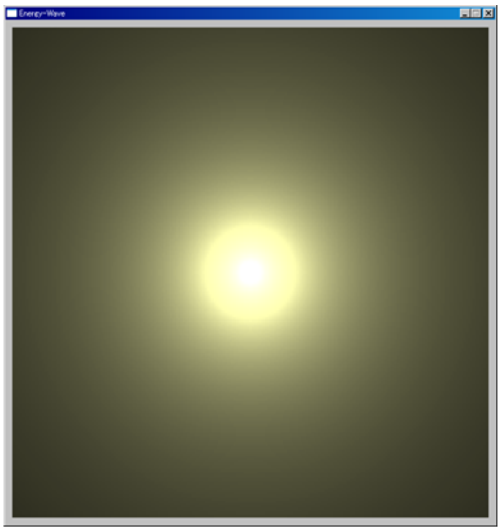


図 4.3: 実行結果：球体＋円柱

図 4.4 はパラメータ変更による球体形状変形、図 4.5 はパラメータ変更による円柱形状変形、図 4.6 はパラメータ変更による円柱形状と球体形状を組み合わせた形状変形を示したプログラムの実行結果である。

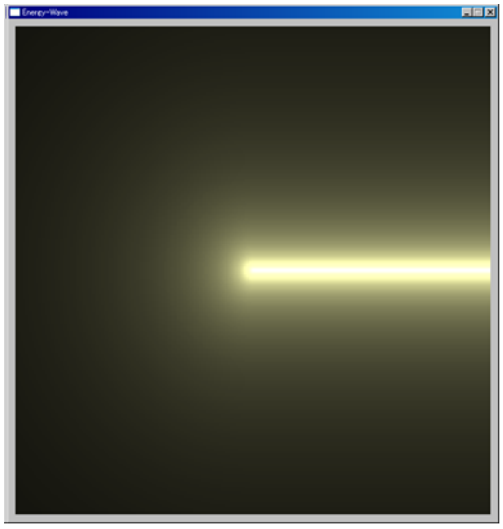


パラメータ a：60

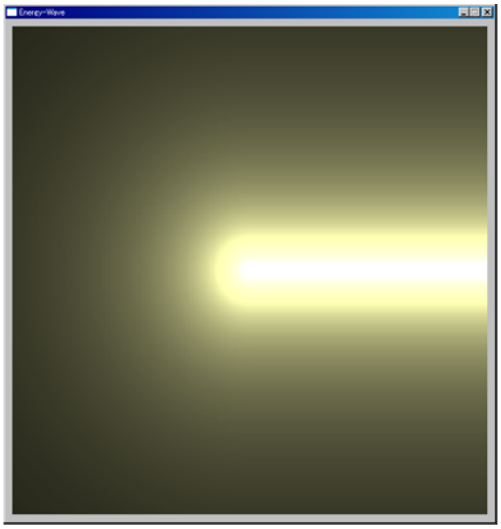


パラメータ a：90

図 4.4: 実行結果：球　パラメータ比較

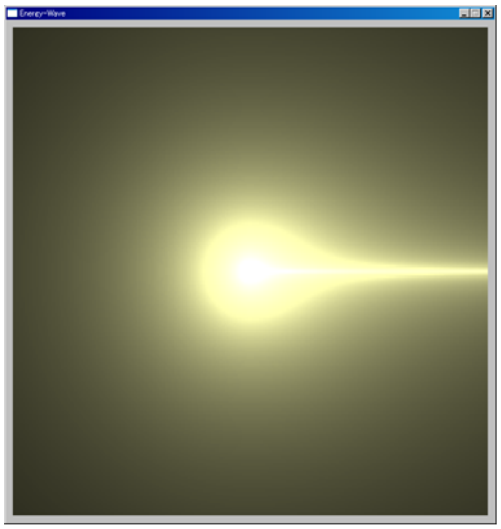


パラメータ b：60

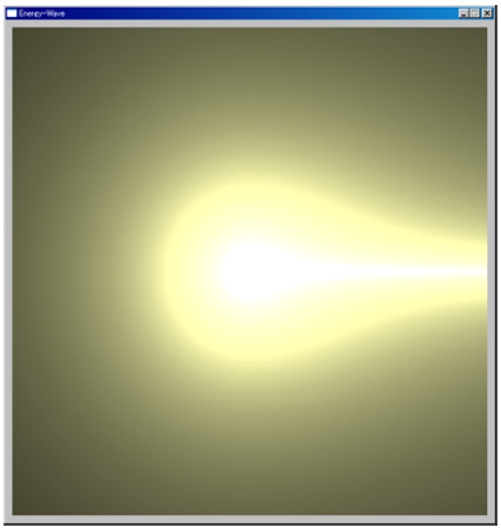


パラメータ b：90

図 4.5: 実行結果：円柱　パラメータ比較



パラメータ a：60
パラメータ b：30



パラメータ a：60
パラメータ b：50

図 4.6: 実行結果：変形制御

図 4.7 は任意地点 **M** を変更したエネルギー波の移動制御を示したプログラムの実行結果である。

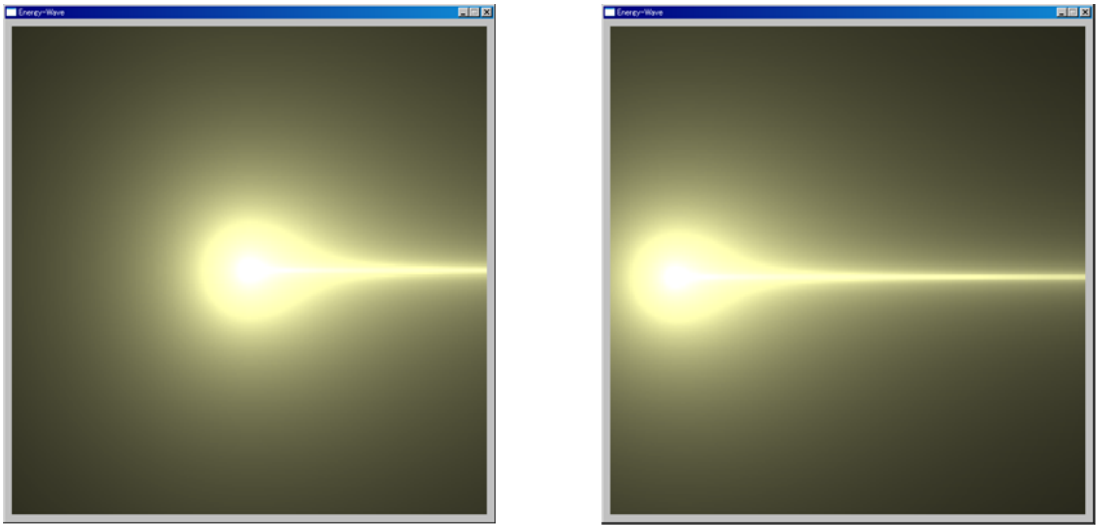


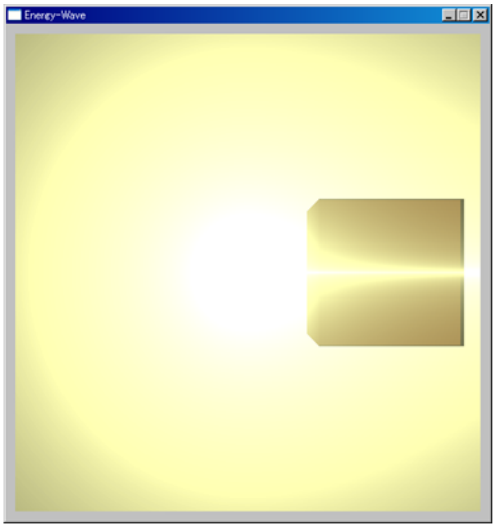
図 4.7: 実行結果：移動制御

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 $b = 30$ とし、任意点 **M** のパラメータ m_z を動的に変更した。他モデルとして 4 面体形状を配置した。配置したモデルの位置はそれぞれ $(0.0, 0.0, 0.0)$、$(10.0, 0.0, 250.0)$ であり、デプス値はそれぞれ、$z_1 = 0.5$ および $z_2 = 0.075$ となる。図 4.8 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 4.9 は $m_z = 160$ のときの実行結果である。図 4.10 は $m_z = 70$ のときの実行結果である。図 4.11 は $m_z = 10$ のときの実行結果である。図 4.12 は $m_z = -50$ のときの実行結果である。図 4.13 は $m_z = -140$ のときの実行結果である。

4.2 実行速度検証

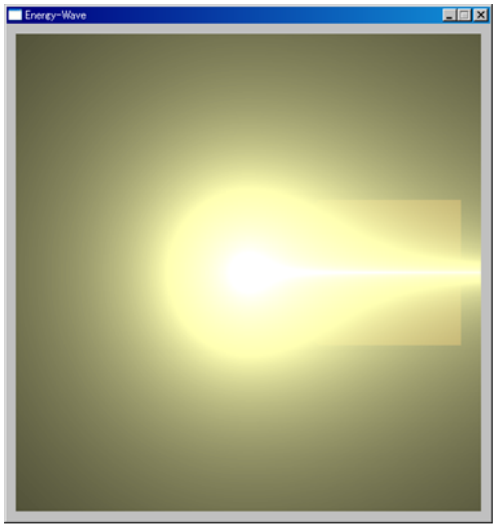
本手法で提案したエネルギー波表現手法を実装したプログラムの処理速度を検証した。処理速度の測定に用いた PC の構成は表 4.1 に示す。

隠面消去処理：無し

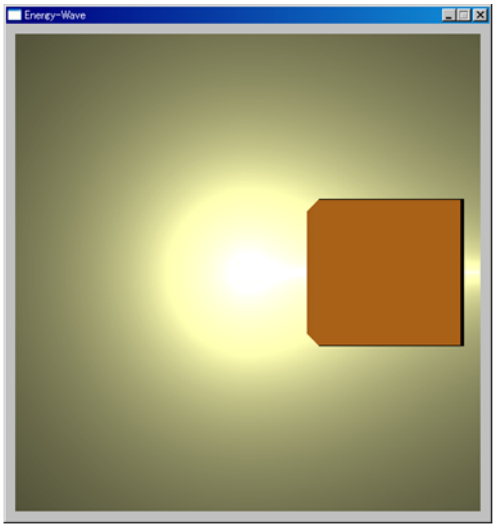


隠面消去処理：有り

図 4.8: 実行結果：$m_z = 280$ の実行結果

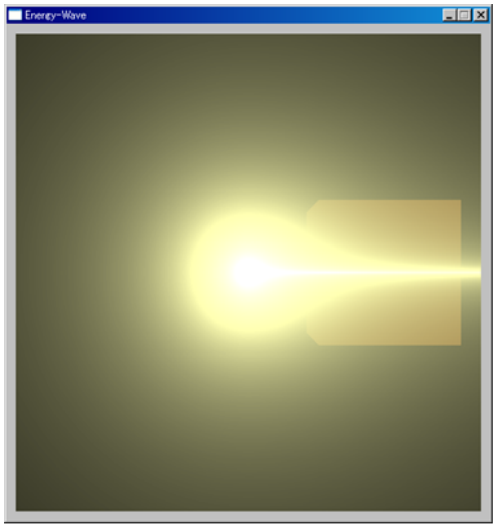


隠面消去処理：無し

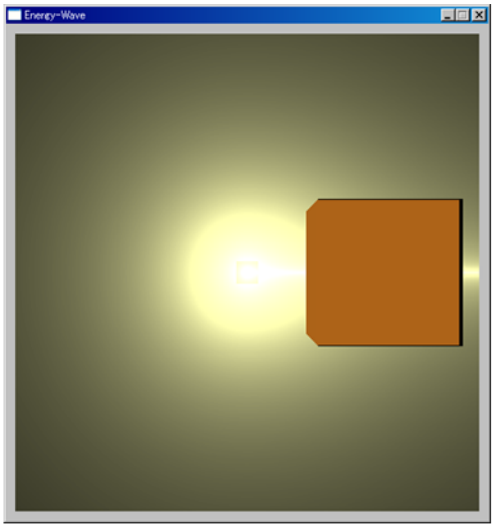


隠面消去処理：有り

図 4.9: 実行結果：$m_z = 160$ の実行結果

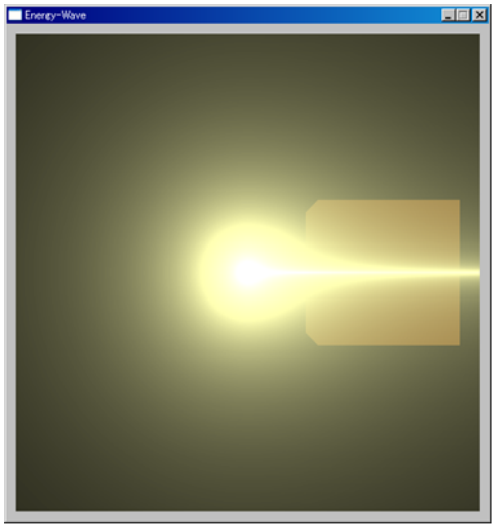


隠面消去処理：無し

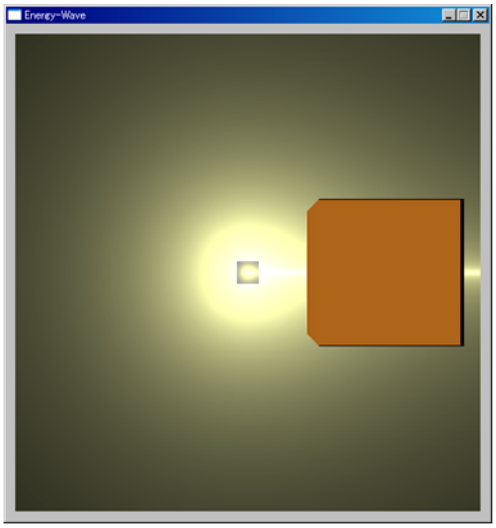


隠面消去処理：有り

図 4.10: 実行結果：$m_z = 70$ の実行結果

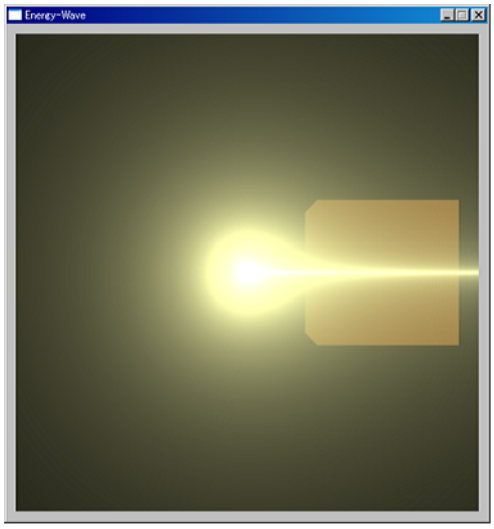


隠面消去処理：無し

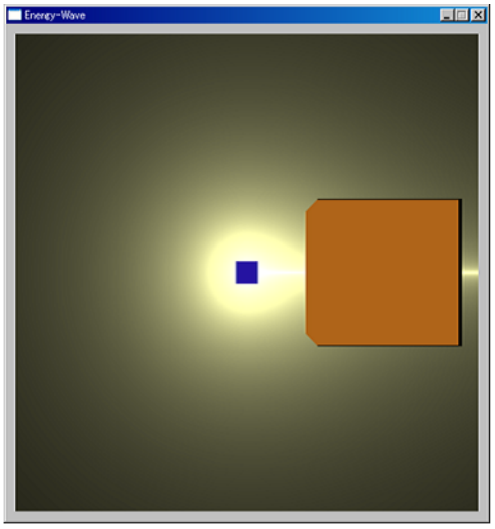


隠面消去処理：有り

図 4.11: 実行結果：$m_z = 10$ の実行結果

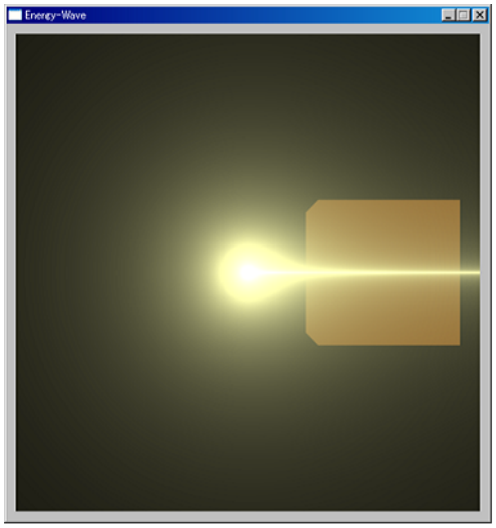


隠面消去処理：無し

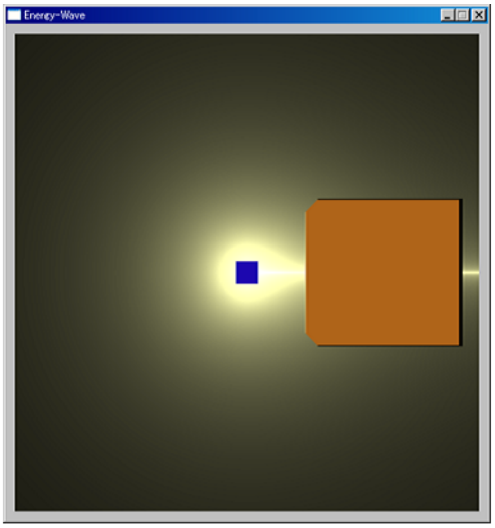


隠面消去処理：有り

図 4.12: 実行結果：$m_z = -50$ の実行結果



隠面消去処理：無し



隠面消去処理：有り

図 4.13: 実行結果：$m_z = -140$ の実行結果

表 4.1: 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

以上の環境下において、256 × 256 画素、512 × 512 画素の描画結果を生成した。球体および円柱形状を 1 つずつ配置した状態を 1 セットとし、複数のセット数で描画処理の検証を行った。処理速度の単位は FPS(Frame Per Second) であり、1 秒間に可能な描画処理の回数を表す。以下の表 4.2 にそれぞれの画素数における処理速度の結果を示す。

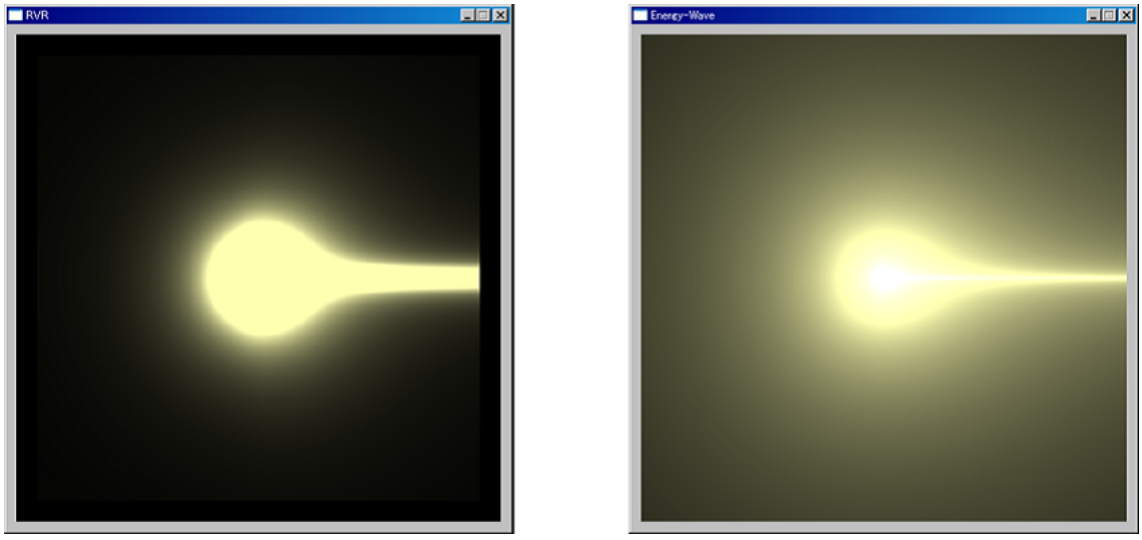
表 4.2: 提案手法 処理速度の測定

セット数	256 × 256 画素	512 × 512 画素
1	105 FPS	27 FPS
5	90 FPS	23 FPS
10	65 FPS	17 FPS

表 4.2 より、描画するエネルギー波形状の増加によって描画速度は低下する。しかし移動・変形制御ではほぼ処理速度の低下は見られなかった。このことより、本手法においては、リアルタイム性は実現できたと言える。

4.3 既存手法との比較

既存手法であるレイキャスティング法で生成したエネルギー波表現と、本手法で提案したエネルギー波表現との比較を行った。図 4.14 はそれぞれの手法で生成したエネルギー波の比較を示している。レイキャスティング法において、1 本の視線におけるサンプリング回数は 120 回である。サンプリングの回数は、描画面の解像度がそれぞれの手法において同じ場合、本手法とほぼ処理速度が等しくなる回数とした。レイキャスティング法では全体として色のグラデーションが少ないのに対し、本手法では分布関数の値を正確に反映しているため、細かな色のグラ



レイキャスティング法　　提案手法

図 4.14: 既存手法との表現の差異

デーションも表現可能となっている。また、レイキャスティング法は離散的なデータを扱うため、特に円柱形状を表現する場合、エネルギーが強まる箇所が飛び飛びになってしまうという問題点がある。図 4.15 はレイキャスティング法において、円柱のパラメータを増加している様子を示す。

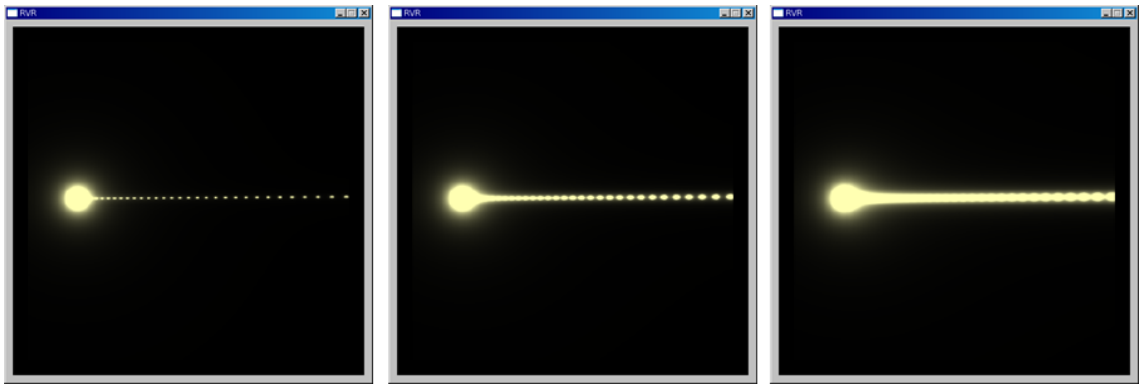


図 4.15: レイキャスティング法における円柱形状の拡大

以上のことから本手法は既存手法に比べ、エネルギー波の 3 次元分布状況を正確に反映可能であるといえる。

4.4 問題点

現在の問題点として、第 1 にエネルギー波形状の増加に伴う処理速度の低下が挙げられる。これは GPU 内で行う計算量が増加する為に、CPU と GUP 間でのデータ転送量やデータ転送頻度、GPU 内での計算方法がボトルネックとなっている部分が大きい。この問題は GPGPU のプログラミングコード改良により、処理速度の向上が望めると考える。

第 2 に、複雑な形状表現・変形が困難という点が挙げられる。現在の手法では、エネルギー波形状は複数の球体および円柱形状で生成するため、先述の問題点より複雑なエネルギー形状を生成することが困難である。また、エネルギー波形状を関数で規定しているため、大域的な形状変形は得意であるが、局所的な形状変形は不得意である。この問題を解決するためには先述の問題解決と同様に、GPGPU プログラミングの改善による処理速度の向上が挙げられる。複数の球体および円柱形状を高速に生成可能になれば、複雑なエネルギー波形状の生成も容易になる。また、球体および円柱以外の基本となるエネルギー波形状規定用関数を設定することが出来れば、更なるエネルギー波形状表現の向上が実現可能であると考える。

第 5 章

おわりに

本研究では、リアルタイムコンテンツにおける新たなエネルギー波表現を提案した。不定形な現象をレンダリングするための技術であるボリュームレンダリングの概念を踏襲しつつ、エネルギーの 3 次元分布状況を線積分式となる関数を用いることにより、ボリュームデータのサンプリングを必要とするくことなく、正確且つ高速に 3 次元分布状況を算出可能となった。その結果として、従来の手法では不可能であった任意視点からのエネルギー波の光の強さを正確に表現可能かつ、形状変形や移動制御を実現した。また、エネルギー波に隠面消去処理を施すことで、一般的なモデルとの前後関係を正確に表現し、創作コンテンツ内におけるエネルギー波表現の有用性を高めた。本手法で提案した手法を用いることで、インタラクティブゲームやアニメーションなどのインタラクティブコンテンツでのエネルギー波の表現の幅を広げるという目的において、大いに役立つだろう。

今後の展望として、4.4 節で問題点として挙げた更なるエネルギー波形状変形の手法、他モデルへの光源効果としての影響の考察が挙げられる。エネルギー波の大域的な形状変形は実現できたが、局所的な形状変形は不可能である。組み合わせる関数の数を増やすことで、複雑な形状を表現することは可能だが、描画にかかる計算処理が増加し、描画速度の低下に繋がる。また、他モデルへの影響として、光源としての効果が追加できれば、更なるエネルギー波表現の向上に繋がるだろう。

本研究は芸術科学会第 25 回 NICOGRAPH 論文コンテストにおいて “エネルギー波表現のリアルタイムレンダリング” [30] として発表した内容を含む。

謝辞

最初の感謝は渡辺大地先生へ。言葉になりません。

次の感謝は近藤邦雄先生、三上浩司先生へ。研究は外へ出します。

礎の感謝は坂井悠基君へ。君が植えたエネルギー波の種はここまで成長しました。

公私の感謝は望月順一君へ。今でも出会いは忘れません。

戦友の感謝は地神知哉君へ。君の戦線復帰はいつまでも待っています。

苦悩の感謝は渡辺賢悟さんへ。垣根を越えた苦悩談義は大いなる財産です。

影響の感謝は宮内優太さんへ。あなたがあって初めて私がいます。

目標の感謝は大魔王の息子へ。あなたの技が目標でした。

青春の感謝はメンバー達へ。ベランダとは、かくも美しいものです。

継続の感謝は自分へ。よくぞやり遂げました。

歓喜の感謝は未知へ。前人未踏とは発見することすら困難です。

そして最後の感謝はあなたへ。本論文が少しでも力になれることを祈っています。

参考文献

[1] 鳥山明, 「ドラゴンボール」, 集英社, 1985.

[2] 「機動戦士ガンダム」, 日本サンライズ, 1979.

[3] M. Levoy, “Display of surface from volume data”, IEEE computer Graphics and Applications, 8(3) pp.29-37, 1988.

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

[5] 3D 画像処理エンジン,「VolumePro100」, TERARECON.INC, <http://www.terarecon.co.jp/medical/oem1.html>

[6] H.Pfister, J.Hardenbergh, J.Knittel, H.Lauer, L.Seiler, “The VolumePro Real-Time Ray-Casting System”, In Proceedings of the ACM SIGGRAPH ’99 Conference, pp.131-138, 1998.

[7] 小林敏彦, 佐藤秀二, 藤井哲也, 江浩, “VolumePro を用いた 3 次元医療用画像ソフトウェア INTAGE RV の開発”, Proceedings of the Society Conference of IEICE, pp.321-322, 2000.

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

[9] W.E. Lorensen and H.E. Cline, “Marching cubes: a high resolution 3d surface reconstruction algorithm”, In Proceedings of the SIGGRAPH Annual Conference on Computer graphics, 1987.

[10] R. Westermann and T. Ertl, “Efficiently Using Graphics Hardware in Volume Rendering Applications”, In Computer Graphics Proceedings, Annual Conference Series, pp.169-177, 1998.

[11] 山崎俊太郎, 加藤究, 池内克史, “PC グラフィクスハードウェアを利用した高精度・高速ボリュームレンダリング手法”, 情報処理学会 CVIM-130-10, 2001.

[12] T. Gunther, C. Poliwoda, C. Reinhart, J. Hesser, R. Manner, H.-P. Meinzer, and H.-J. Baur, “Virim: A massively parallel processor for real-time volume visualization in medicine”, Computers & Graphics, 19, 5, pp.705-710, 1995.

[13] M. Ogata, H. Ohkami, H.C. Lauer and H. Pfister, “A Real-Time Volume Rendering Architecture with Resampling Scheme for Parallel and Perspective Projections”, Proceedings of the ACM/IEEE Symposium on Volume Visualization, pp.20-29, 1998.

[14] 原瀬史靖, 山内聡, 森眞一郎, 津邑公暁, 五島正裕, 中島康彦, 北村俊明, 富田眞治, “ReVolver/C40 を用いた時系列ボリュームデータの実時間可視化”, 情報処理学会研究報告 計算機アーキテクチャ研究会報告, 2002(37), pp.7-12, 2002.

[15] P.G. Lacroute and M. Levoy, “Fast Volume rendering using a shear-warp factorization of the viewing transformation”, In Proceedings of the ACM SIGGRAPH ’94 Conference, pp.451-457, 1994.

[16] Kwan-Liu Ma, “Parallel Volume Rendering Using Binary-Swap Compositing”, In IEEE Computer Graphics and Applications, 14, 4, pp.59-68, 1994.

[17] 丸山悠樹, 中田智史, 高山征大, 津邑公暁, 五島正裕, 森眞一郎, 中島康彦, 富田眞治, “汎用グラフィクスハードウェアを用いた並列ボリュームレンダリングの実装”, IPSJ SIG Notes, 2003.

[18] Matthias Hopf, Thomas Ertl, “Accelerating 3D Convolution using Graphics Hardware”, IEEE Visualization, 1999.

[19] C.Rezk-Salama, K. Engel, M. Bauer, G. Greiner, T. Ertl, “Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and

Multi-Stage Rasterization", In Proc. Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2000.

[20] 篠本雄基, "汎用 GPU を用いたボリュームレンダリングの高速化に関する研究", 京都大学大学院情報学研究科修士論文, 2006.

[21] 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

[22] J.Kruger, R.Westermann, "Acceleration techniques for gpu-based volume rendering", In Proc IEEE Visualization 2003, pp.287-292, 2003.

[23] C.Rezk-Salama, "GPU-Based Monte-Carlo Volume Raycasthin", In Proc. Pacific Graphics, 2007.

[24] Johanna Beyer, Markus Hadwiger, Torsten Möller, Laura Fritz, "Smooth Mixed-Resolution GPU Volume Rendering", In IEEE/EG International Symposium on Volume and Point-Based Graphics, pages 163-170, 2008.

[25] E.Gobbetti, F.Marton, J.A.I.Guitián, "GPU Ray Casting Framework for Interactive Out-of-Core Rendering of Massive Volumetric Datasets", The Visual Computer 24, 797-806, 2008.

[26] OpenGL.org, OpenGL, <http://www.opengl.org/>.

[27] 渡辺大地, "リアルタイムグラフィックスのためのツールキットに関する研究", 慶應義塾大学大学院政策・メディア研究科修士論文, 1996.

[28] 渡辺大地, FK Tool Kit System, <http://fktoolkit.sourceforge.jp/>.

[29] CUDA, <http://www.nvidia.co.jp/object/cuda_home_jp.html>

[30] 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.

発表論文

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.