任意のエネルギー分布関数に基づく
3DCG のためのエネルギー波表現に関する研究

2016 年 9 月

岩手大学大学院工学研究科

デザイン・メディア工学専攻

渡辺 大地

博士学位論文

任意のエネルギー分布関数に基づく 3DCG のためのエネルギー波表現に関する研究

岩手大学大学院工学研究科

デザイン・メディア工学専攻

渡辺 大地

2016 年 9 月

概要

アニメーションや漫画などの創作コンテンツの中で、エネルギーの塊が強く発光し、移動していくという表現がある。これは特撮映像において光線を表した表現が、特に日本のアニメーションの世界で大きく発展したものである。近年、日本国内のアニメーション愛好家の間では、このような創作表現を「エネルギー波」と呼ぶことが多い。日本以外ではこのような呼称は一般的ではないが、アニメーションやコンピューターグラフィックス (以下「CG」) における映像効果としては一般的なものとして認識されている。本論文では、エネルギーの塊が強く発光し、移動する表現を日本での表現にならい「エネルギー波」と呼称する。エネルギー波は、「空間中のエネルギー密度が高い箇所が強く発光する」という架空の物理現象と定義する。また、エネルギー波はエネルギー分布の移流により発光体形状も移動や変化が起こるものとする。

エネルギー波を用いた特殊効果は、実写映像、CG 映像、アニメーションに関わらず様々なコンテンツで利用されている。また、近年ではリアルタイムグラフィックス技術の高速化により、ゲームなどのインタラクティブコンテンツでも利用されるようになった。しかしながら、リアルタイムグラフィックスは一般的に平面や曲面に関しての描画機能は充実しているが、空間中の分布を描画することには適した構成をしていない。ゲーム制作現場においてよく用いられる手法は、アニメーションテクスチャによる表現や、多数のパーティクルアニメーションによる方法である。アニメーションテクスチャ表現は、描画の様子をアーティストが画像として入力できるという利点を持つが、エネルギー波と外側の境界が明確になりすぎてしまうことや、透明度を利用して境界を曖昧にしたときに適切な視点の位置と姿勢が限定されてしまうという問題がある。また、パーティクルを用いた手法は一般的に、視点の位置や姿勢に対する制限はないが、描画した印象は粒状光源の集合というものであり、エネルギーが空間全体に稠密な状態の表現が困難であることや、

広い領域に渡る表現に向いていない。

稠密なエネルギー分布による光源効果を実現する従来手法として、フォトンマッピングアルゴリズムを利用したボリュームライトを生成する手法がある。しかし、フォトンマッピングを利用する手法は映画作品のような高精細なアニメーションを作成することを目的としており、リアルタイムな処理の実現には向いていない。他にも稠密な空間領域に対する光学現象を実現する研究は数多くある。ボクセルデータ構造を用いた方法は最も一般的に普及しているものの一つである。近年、GPU(Graphics Processing Unit) によるボリュームレンダリングに関する多くの手法が提案されている。GPU によるボリュームレンダリングは非常に高速な描画を可能とするが、分布データの更新に時間がかかることや、広い領域を表現することには向いていないといった問題がある。

本研究は、エネルギー波の表現に特化した、エネルギーが稠密な空間に対するエネルギー波表現のリアルタイムレンダリング手法を提案する。基本的な方針は、次の 2 つの理論に基づいている。

(1) 3 次元空間におけるスカラー場となるエネルギー分布関数を定義し、任意の点におけるエネルギー値の算出を可能とする。
(2) 視点から各画素を通る直線上のエネルギー量を、線積分によって求め、その量を画素輝度値としてレンダリングを行う。

エネルギー分布関数については、従来手法にて一点を中心とした分布関数や線分を中心に円柱形状となる分布関数は既に提案されていたが、本研究ではそれに加えトーラス型の分布関数と 2 次曲線による分布関数を提案する。2 次曲線は、自由曲線としてよく知られている 2 次 Bézier 曲線や放物線を含むものであり、従来と比べて分布状態を自由に作成することが可能となる。また、2 次曲線に基づく分布関数においては、任意の部位の強弱を調整する手法についても提案する。

これにより、曲線上に沿った高速なエネルギー波の移動を簡易なパラメータ調整によって実現できる。

画素輝度値を求めるための線積分処理は、画素ごとに個別に行われるものであるため、膨大な計算量が必要となる。従来の逐次処理による実装では一回当たりの出力画像作成に数秒から数十秒を要するものであり、リアルタイムグラフィックスへの適用は現実的ではない。そこで本研究では、GPGPU(General Purpose on GPU) 技術を用いて画素ごとの演算を並列に行うことで高速な演算を実現し、リアルタイムグラフィックスでの実用に適応できる処理速度を実現した。

以上の理論と実装に基づき、本研究では形状変形を伴うエネルギー波のリアルタイムレンダリングを可能とした。検証においては、解像度や各種パラメータ値による実行速度や描画結果に関して比較を行い、本手法の有効性を示した。

目 次

図目次

表目次

第 1 章

序論

1.1 エネルギー波について

アニメーションや漫画といった創作コンテンツの中で、エネルギーの塊が強く発光し、移動していくという表現がある。これは特撮映像において光線を表した表現が、特に日本のアニメーションの世界で大きく発展したものである。近年の日本国内において、このような創作表現はアニメーションの愛好家の間で「エネルギー波」と呼ばれるようになった。日本以外ではこのような呼称は一般的ではないようであるが、アニメーションやコンピュータグラフィックス (以下「CG」) の中でのエフェクトとしては一般的なものとして認識されている。一方で、Nowrouzezahrai[1] は「Artistic Volumetric Light」と呼称している。

本論文では、エネルギーの塊が強く発光し、移動していく表現を日本での表現にならい「エネルギー波」と呼称する。エネルギー波の定義は、阿部ら [2][3] と同様に「空間中のエネルギー密度が高い箇所が強く発光する」という架空の物理現象とし、エネルギー分布の移流により発光体形状も移動や変化が起こるものとする。

エネルギー波表現は、現実世界におけるレーザー光線や炎などの物理現象の特徴をある程度は踏襲しているものの、実際にはそのような現象は現実世界では生じないものであり、あくまでコンテンツ制作者の想像上の産物である。しかしながら、エネルギー波表現は数十年に渡って多くの作品で用いられてきており、作品ごとに細かな差異はあるものの、ある程度共通の特性を持っている。

エネルギー波の外観は、実世界での光の道筋が観測できる現象である薄明光線やレーザーと類似している。しかしエネルギー波は光とは異なり、光速で移動するわけではなく、多くの場合目視可能な速度で移動する。また、エネルギー波は他の物体と衝突し、衝突により形状が大きく変化する。そして衝突後のエネルギー波は間もなく消えていく特徴を持つ。これらの点が実世界における薄明光線やレーザー光線とは異なる。

実写特撮映像やアニメーションにおいて、いつからエネルギー波と同様の表現が用いられるようになったかは定かではないが、1941 年にアメリカの Fleischer Studios 社が作成したアニメーション作品「Superman」において、既に現在の「エネルギー波」表現に近い効果が用いられていることから、かなり古い歴史を持っていることがうかがえる。図 1.1 は「Superman」においてエネルギー波表現が用いられている場面である。

図 1.1　アニメ「Superman」の場面

©Fleischer Studios, Inc.

実写特撮映像においても、古くは「ゴジラ」[4] や「ウルトラマン」[5] といった作品の中でエネルギー波表現が用いられており、1960 年代には既に特撮技法として定着していたと考えられる。当時は主に「光線」という用語が用いられていたため、レーザー光線現象を参考にして制作されていたものと思われる。また、1977 年の映画「スターウォーズ」[6] においては、先述した光線銃だけでなく、刀の刀身にあたる部分が尖形状の光によって構成されている「ライトセーバー」という武器が登場する。ライトセーバーなどは先述した本研究におけるエネルギー波の定義である「空間中のエネルギー密度が高い箇所が強く発光する」という性質が前提となっており、その後の様々な SF 作品の映像表現に大きな影響を与えている。

日本のアニメーションにおいても、エネルギー波表現は多くの SF 作品で頻繁に用いられている。代表的なものとして、1974 年のアニメ「宇宙戦艦ヤマト」[7] の「波動砲」、1979 年のアニメ「機動戦士ガンダム」[8] の「ビームサーベル」などがある。この時期になると、エネルギー波表現は単純に光線を模したものではなく、曲線を描くものや途中で分離するものなど、多様な表現が扱われるようになった。

これらの表現効果に対し、「エネルギー波」という呼称が最初に用いられたのは漫画「ドラゴンボール」の作品中においてである [9]。作品中では、体内エネルギーを光線状に放出する攻撃を「エネルギー波」、弾状に放出する攻撃を「エネルギー弾」と呼称した。作品の爆発的人気に伴い、他の創作コンテンツにおいても同様の現象が「エネルギー波」と呼ばれることが多くなった。

エネルギー波を用いた特殊効果は、現在においても実写映像、CG 映像、アニメーションに関わらず様々なコンテンツで利用されている。また、近年ではリアルタイムグラフィックス技術の高速化により、ゲームなどのインタラクティブコンテンツでも利用されるようになった。

しかしながら、リアルタイムグラフィックスは一般的に平面や曲面に関しての描画機能は充実しているが、空間中の分布を描画することには適した構成をしていない。ゲーム制作現場においてよく利用されている手法は、アニメーションテクスチャを用いるものや、パーティクルアニメーションによる描画手法である。アニメーションテクスチャは、描画の様子をアーティストが画像として入力できるという利点を持つが、エネルギー波と外側の境界が明確になりすぎてしまうことや、透明度を利用して境界を曖昧にしたときに適切な視点の位置と姿勢が限定されてしまうという問題がある。

1.2 本研究の概要

本研究は、阿部ら [2][3] の研究と同様にエネルギー波の表現に特化した、空間で稠密な分布関数およびそのリアルタイムレンダリング手法を提案する。一点を中心としたものや線分を中心に円柱形状となるものに加え、本研究ではトーラス型の分布と 2 次曲線による分布を実現した。2 次曲線は、自由曲線である 2 次 Bézier 曲線や放物線を含むものであり、従来と比べて分布状態の作成の自由度が大きく向上した。また、先行手法の多くは分布状態の動的変化に向いていないのに対し、本手法は毎描画フレームごとにレンダリングを行うため、非常に高速な変形処理を実現できることが特長である。

1.3 論文構成

2 章では、本研究における関連研究と先行研究を紹介すると共に、本研究との位置づけを明確にする。2.1 節では、リアルタイムグラフィックスにおけるエネルギー波表現を実現する既存手法とその問題点について述べる。2.2 節では、本手法の基礎理論となっている阿部ら [2] の手法について詳しく述べる。

3 章にて手法理論について述べる。まず、3.2 節にてエネルギー分布関数について定義を行う。特に、阿部手法にはなかったトーラス型と自由曲線型について詳しく述べる。3.3 節では自由曲線型における部位強弱調整手法について述べる。3.4 節で複数エネルギー波の加算方法を、3.5 節で数値積分による画素輝度値算出方法を述べる。

4 章で GPGPU を用いた高速なレンダリング手法について述べる。まず GPGPU についての概略と、本研究のおける GPGPU 環境の解説を述べる。次に CPU と GPU 間のデータ転送に用いる配列の定義を行った後に、数値積分手法に基づき線積分値を算出する手法について述べる。

また、GPGPU による一連の処理をフローチャートとして図示する。

5 章で本研究での検証結果について述べる。5.2 節で実行環境を述べたのち、5.3 節では実行速度、5.4 節では出力結果の分析を述べる。また、5.5 節では、近似積分の際の分割数が出力にどのように影響を及ぼすかについての分析を述べる。5.6 節で、本手法の問題に対する改善点の考察について述べる。

最後に 6 章で本研究の成果をまとめ、今後の課題と展開について考察を行う。

第 2 章

関連研究

本章では、本研究における関連研究と先行研究を紹介すると共に、本研究との位置づけを明確にする。

2.1 リアルタイムグラフィックスにおけるエネルギー波表現

リアルタイムグラフィックスにおいてエネルギー波を表現する際には、

- アニメーションテクスチャ
- パーティクルアニメーション

の 2 つの手法がよく用いられる。本節では、まずこれらの手法について解説する。次に、エネルギー波表現として 2 つの手法を用いたときの問題点を述べる。また、2.1.3 項以降では、エネルギー波表現と関連する各種手法について述べ、合わせてエネルギー波として表現する際の問題点について述べる。

2.1.1 アニメーションテクスチャを主体とした手法

ゲームなどのリアルタイム 3D コンテンツ上でエネルギー波を表現する手法として主流なものにアニメーションテクスチャによる表現がある。これは、透過テクスチャを平面あるいは曲面として空間中に配置し、そのテクスチャに対しエネルギー波を描画したアニメーション画像を表示するというものである。アニメーションテクスチャを用いた作品のスナップショットを図 2.1 に示す。

図 2.1　アニメーションテクスチャの実行例

アニメーションテクスチャを用いる手法は、画像を動的に作成することによってデータ容量を削減することができ、描画処理速度も高速であるという利点がある。また、エネルギー波の描画状態をデザイナーが事前にデザインすることができるため、コンテンツ制作現場において分業が容易である。そのため、エネルギー波表現のようなエフェクト制作の方法として広く受け入れられている。この方法を採用した制作ツールも多く、代表的なものとして BISHAMON [10], Effekseer [11] などがある。

しかしながら、この手法は特定の視点からの表現を前提としているため、その他の視点からの描画では不適切な表現となるという問題がある。例えば、エネルギー波の内部に視点がある場合にエネルギー波が描画されなくなる。また、大抵の場合は複数画像のアニメーションとしてデータを格納しているため、状態の動的な変化には対応しづらいという問題もある。

2.1.2　パーティクルを主体とした手法

近年ではシェーダー技術の発達により、前述したアニメーションテクスチャに加えて「パーティクルアニメーション」によるエネルギー波表現もよく用いられるようになった。パーティクルアニメーションとは、空間上に分布する多数の粒子に対し、各粒子の位置に円型や線型のエネルギー

波を描画したテクスチャ画像を配置し、多数のテクスチャ画像の集合によりエフェクト表現を実現する手法である [12][13]。各パーティクルは粒子法等の流体力学に基づいた流れの表現が可能であるため、アニメーションテクスチャよりも動的な形状変化表現が可能となっている。また、視点の制約を受けないという利点もある。パーティクルエフェクトを用いたエネルギー波の表現は、仁藤らによる手法 [14][15] などがある。図 2.2 に仁藤らの手法の実行の様子を示す。

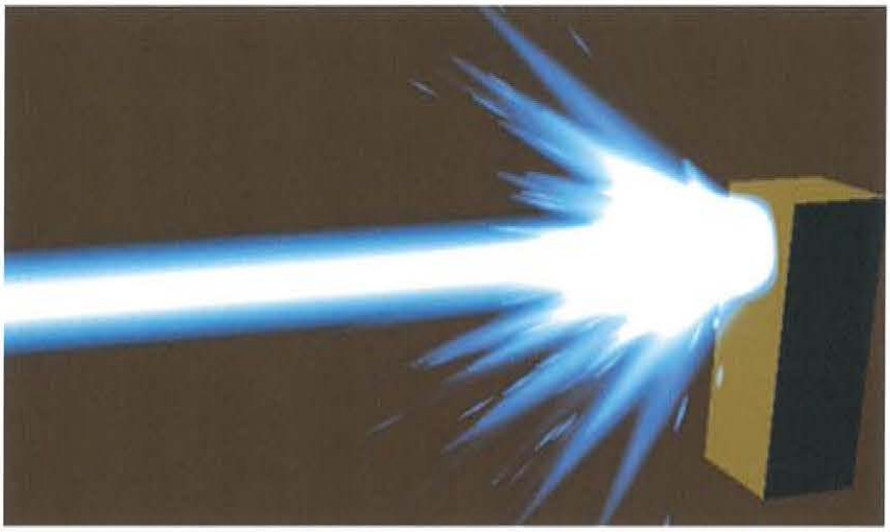


図 2.2　仁藤らの手法の実行例
(文献 [14] 図 24 より引用)

しかしながら、パーティクルアニメーションを用いて描画した印象は粒状光源の集合というものであり、エネルギーが空間全体に稠密な状態の表現が困難であることや、広い領域に渡る表現に向いていないという問題がある。

2.1.3　空間中の連続分布関数を主体とした手法

稠密なエネルギー分布による光源効果を実現した手法として、Nowrouzezahrai らが提案するフォトンマッピングアルゴリズムを利用したボリュームライトを生成する研究 [1] がある。図 2.3 にその実行例を示す。

図 2.3　Nowrouzezahrai らの手法の実行例
(文献 [1] Fig.1 より引用)

文献 [1] による手法では、ボリュームライトの形状だけでなく、それを光源とした他物体への陰影効果についてもフォトンマッピングにより実現しているのが特徴である。

しかしながら、この手法は映画作品のような高精細なアニメーションを作成することを目的としており、オフラインレンダリングを前提とした手法設計となっている。そのため、この手法はかなり大規模な演算を必要とするものであり、現在のコンピュータの処理速度ではリアルタイムな描画には適用できない。

2.1.4　ボリュームレンダリングを主体とした手法

稠密な空間領域に対する光学現象を実現する研究は数多くある。Drebin らの手法 [16] を起源とするボクセルデータ構造を用いた方法は最も一般的に普及しているものの一つである。ボクセルデータは、体内構造の可視化 [17] や、濃淡、煙等の不定形自然現象を表現 [18] する場合に適したデータ構造である。また、竹内らの研究 [19][20][21] にあるように、モデリングへの適用も近年多く提案されており、ZBrush[22] など多くのモデリングソフトウェアがリリースされている。

一般的に、ボリュームレンダリングは高精細な表示が可能である反面、データ容量が肥大化す

る傾向にあり、計算コストも高い [23]。

近年、GPU によるボリュームレンダリングに関する手法 [24][25][26][27] も多く提案がある。図 2.4 は、GPU のボリュームレンダリング機能を用いた描画の様子を示す。これはリアルタイムで視点変更が可能である。

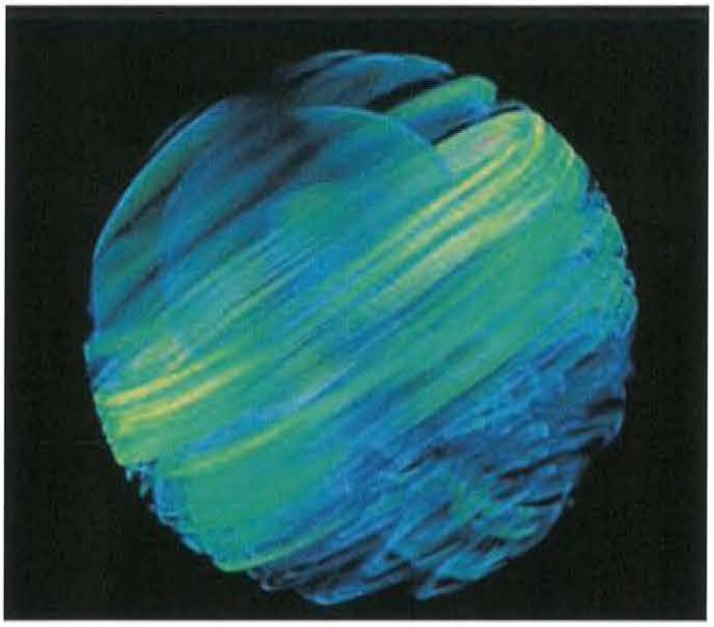


図 2.4　GPU のボリュームレンダリング機能の実行例
(文献 [27] の発表資料 P.28 より引用)

しかしながら、ボリュームレンダリングはそのデータ構造の特性上、大量の記憶領域を必要とするものである。近年の GPU はかなり大きなメモリを搭載する傾向にあるので表現自体は可能であるが、ボクセル値をボリュームデータ全体にわたって更新する処理はやや時間がかかるものとなる。そのため、エネルギー波形状を固定した上で視点を変更することは高速に行えるが、形状自体を変形することにはリアルタイムな処理が行えないという問題がある。

また、ボリュームデータの解像度は 512^3 程度のものであり、狭い領域を表現するには適しているが、広い領域の表現には向いていないという問題もある。

2.1.5 陰関数を主体とした手法

空間の分布表現の手法としては、ボクセルデータの他にも陰関数によるものが多くあり、Blinn の「BlobbyModel」[28] や Nishimura による「Meta-Ball」[29] など、古くから多用されている。近年でも、Kanamori ら [30] や Kanai ら [31] らによる GPU を用いた高速な描画手法がある。この多くは形状表面の描画を目的としており、エネルギー波の描画には直接利用することはできない。透過や濃淡といった表現が可能な手法としては、FRep オブジェクト [32] を用いた Pasko らの手法 [33]、Schmitt らの手法 [34][35] や、Børlum らの手法 [36] などがある。図 2.5 に Børlum らの手法の実行例を示す。この図から、半透明部位の描画が実現できていることがわかる。

図 2.5 Børlum らの手法による実行例
(文献 [36] Fig.1 より引用)

しかしながら、これらの手法は外部からの光線による形状内部での光学現象を再現するものであり、エネルギー波のように空間自体が光るような現象を記述することはできない。

2.1.6　リアルタイムシャドー技術の応用を主体とした手法

リアルタイムシャドーの理論を応用し、チンダル現象による薄明光線を表現する手法として、Billeter らによる手法 [37][38] がある。チンダル現象とは、微小粒子が散乱する大気や不透明な液体中の光が、ミー散乱によって散乱し、光の通路が空間上に光って見える現象のことである [39]。これも空間に稠密な光学的現象のリアルタイムグラフィックスによる表現として有効な手段である。図 2.6 にその実行例を示す。

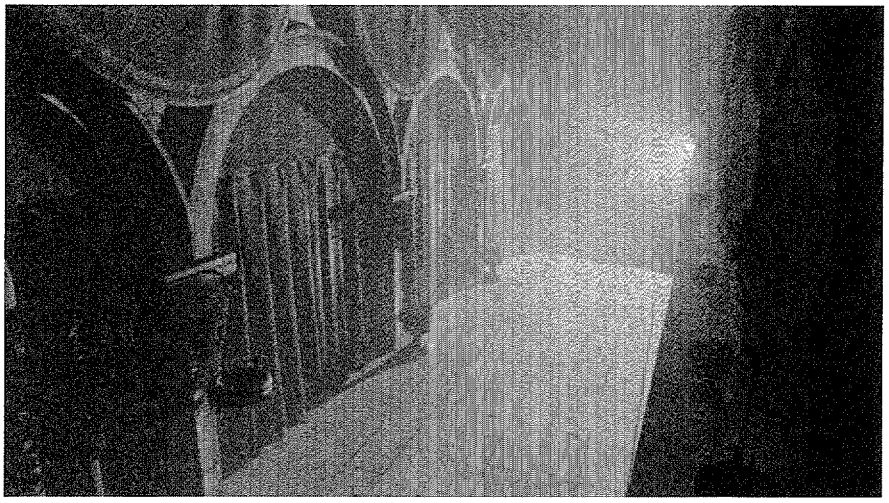


図 2.6　Billeter らの手法による実行例
(文献 [38] Fig.1 より引用)

しかし、この手法は点光源、平行光源、面光源によって生じる薄明光線は表現できるものの、エネルギー波のように光源自体が空間中に分布する表現に適用することができない。

2.1.7　流体表現を主体とした手法

煙、雲、爆発などの自然現象をシミュレーションする研究は多数存在する。多くの手法はシミュレーションによるものであるため、最終的な状態をユーザーがデザインすることは困難である。これに対し、流体の自然な動きを保ちつつ動作や最終状態を制御することを目的とした手法が提

案されている。

Treuille ら [40] や Fattal ら [41] は、ユーザーが指定した形状へ煙が形状変化するアニメーション生成手法を提案した。Dobashi ら [42] は、ユーザーが指定した形状にほぼ一致するように動く積乱雲のシミュレーション手法を提案した。佐藤ら [43] は、複雑なパラメータを必要とせず、ユーザーが指定したステップ数で指定した形状に変化する爆発シミュレーションを提案した。

これらの手法は、実世界における煙、雲、爆発などの現象をもとにシミュレーションを行っており、エネルギー波表現の特質とは異なった面が多く、そのままエネルギー波表現に適用することは困難である。

2.2 本研究の基本となる先行研究

本研究は、阿部らの手法 [2][3] を基本としている。本節では、阿部らの手法のうち本研究を論じるのに必要な事項について紹介する。以降、この手法を「阿部手法」と呼称するものとする。

2.2.1 概要

阿部手法では、創作コンテンツ上でこれまで述べてきたようなエネルギー波表現について、「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」現象と定義した。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している。

阿部手法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している。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、処理コストが低いという利点がある。また、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ている。阿部手法で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。

図 2.7 に、阿部手法によってリアルタイムにレンダリングされたエネルギー波描画のスナップショットを 3 種類示す。視点やエネルギー波の進行方向が変化しても、適切に描画されていることがわかる。

図 2.7　阿部手法での実行の様子
(文献 [2] 付録動画より引用)

2.2.2　エネルギー波形状の生成

阿部手法では、エネルギー波形状を規定するエネルギーの 3 次元分布関数として、2 種類の関数を提案している。

まず第 1 に、任意の一点 $\mathbf{M}$ より放射状に広がる分布関数 $S(\mathbf{P})$ を以下の式 (2.1) で定義した。

$$S(\mathbf{P}) = \frac{a}{|\mathbf{P} - \mathbf{M}|} \tag{2.1}$$

ただし、a は任意の実数パラメータであり、これにより密度の度合いを操作できる。

第 2 に、半直線を中心に放射する分布関数 $C(\mathbf{P})$ を以下の式 (2.2) で定義した。

$$C(\mathbf{P}) = \frac{b}{\sqrt{|\mathbf{P} - \mathbf{M}|^2 - ((\mathbf{P} - \mathbf{M}) \cdot \mathbf{D})^2}} \tag{2.2}$$

ただし、$\mathbf{M}$ は半直線の起点、$\mathbf{D}$ は半直線と平行な単位ベクトルである。また、b は密度調整用の

実数パラメータである。

阿部手法では、この 2 つの関数を組み合わせ、エネルギー波形状を表す分布関数 $E(\mathbf{P})$ を以下の式 (2.3) として定義した。

$$E(\mathbf{P}) = S(\mathbf{P}) + C(\mathbf{P}) \tag{2.3}$$

2.2.3 レンダリング手法

阿部手法におけるレンダリング手法は、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。

また、阿部手法では 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に対し線積分を用いることで最終的な描画結果を得る。図 2.8 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グラフとして比較したものである。左側がサンプリングによるもので、右側が線積分による厳密解の場合である。

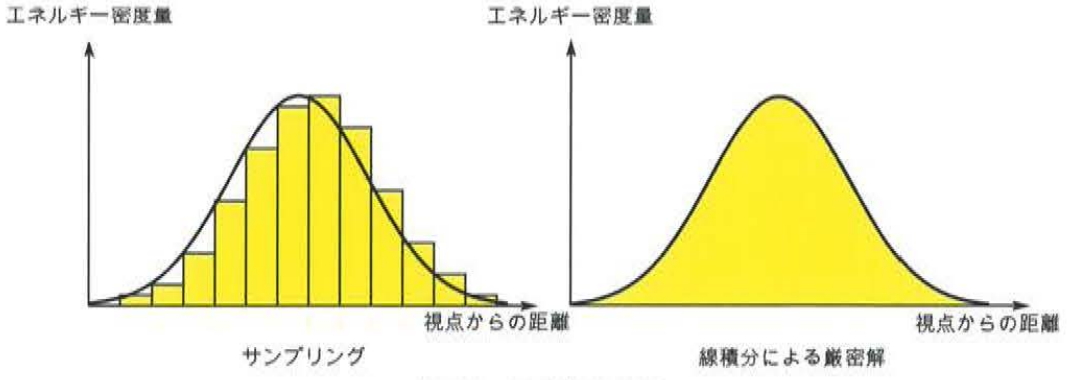


図 2.8　積分値の比較

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 $\mathbf{R}$ を実

数媒介変数 t を用いて式 (2.4) のように表す。

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

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

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

この $\mathbf{R}(t)$ を式 (2.1)、式 (2.2) に代入し、それぞれ式 (2.6)、式 (2.7) を得る。

$$S(t) = \frac{a}{\sqrt{U(t)}}, \qquad U(t) = |\mathbf{R}(t) - \mathbf{M}|^2 \tag{2.6}$$

$$C(t) = \frac{b}{\sqrt{V(t)}}, \qquad V(t) = U(t) - ((\mathbf{R}(t) - \mathbf{M}) \cdot \mathbf{D})^2 \tag{2.7}$$

式 (2.6)、式 (2.7) のうち、$U(t)$、$V(t)$ は t の 2 次式となる。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。

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

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

ここで、ベクトル $\mathbf{S}, \mathbf{M}, \mathbf{D}, \mathbf{Q}$ の各成分を式 (2.10) で示すように表記するものとする。

$$\begin{aligned} \mathbf{S} &= (s_x, s_y, s_z), \\ \mathbf{M} &= (m_x, m_y, m_z), \\ \mathbf{D} &= (d_x, d_y, d_z), \\ \mathbf{Q} &= (q_x, q_y, q_z) \end{aligned} \tag{2.10}$$

このとき、式 (2.8) の A, B, C はそれぞれ式 (2.11) で算出できる。

$$A = q_x^2 + q_y^2 + q_z^2 \tag{2.11}$$

$$B = q_x s_x + q_y s_y + q_z s_z - q_x m_x - q_y m_y - q_z m_z \tag{2.12}$$

$$C = (s_x - m_x)^2 + (s_y - m_y)^2 + (s_z - m_z)^2 \tag{2.13}$$

また、式 (2.9) の D, E, F はそれぞれ式 (2.14)、式 (2.15)、式 (2.16) で算出できる。

$$\begin{aligned} D = {} & (1 - d_x^2)q_x^2 + (1 - d_y^2)q_y^2 + (1 - d_z^2)q_z^2 \\ & - 2(d_x d_y q_x q_y + d_x d_z q_x q_z + d_y d_z q_y q_z) \end{aligned} \tag{2.14}$$

$$\begin{aligned} E = {} & m_x d_x d_x q_x + m_x d_x d_y q_y + m_x d_x d_z q_z \\ & + m_y d_x d_y q_x + m_y d_y d_y q_y + m_y d_y d_z q_z \\ & + m_z d_x d_z q_x + m_z d_y d_z q_y + m_z d_z d_z q_z \\ & - d_x d_x q_x s_x - d_x d_y q_y s_x - d_x d_z q_z s_x \\ & - d_x d_y q_x s_y - d_y d_y q_y s_y - d_y d_z q_z s_y \\ & - d_x d_z q_x s_z - d_y d_z q_y s_z - d_z d_z q_z s_z \\ & - m_x q_x - m_y q_y - m_z q_z \\ & + q_x s_x + q_y s_y + q_z s_z \end{aligned} \tag{2.15}$$

$$\begin{aligned} F = {} & 2(m_x d_x d_x s_x + m_x d_x d_y s_y + m_x d_x d_z s_z \\ & + m_y d_x d_y s_x + m_y d_y d_y s_y + m_y d_y d_z s_z \\ & + m_z d_x d_z s_x + m_z d_y d_z s_y + m_z d_z d_z s_z \\ & - d_x d_y s_x s_y - d_y d_z s_y s_z - d_x d_z s_x s_z \\ & - m_x m_y d_x d_y - m_y m_z d_y d_z - m_x m_z d_x d_z \\ & - m_x s_x - m_y s_y - m_z s_z) \\ & + m_x^2 + m_y^2 + m_z^2 + s_x^2 + s_y^2 + s_z^2 \\ & - m_x^2 d_x^2 - m_y^2 d_y^2 - m_z^2 d_z^2 - d_x^2 s_x^2 - d_y^2 s_y^2 - d_z^2 s_z^2 \end{aligned} \tag{2.16}$$

以上により、**S** から **E** までのエネルギー分布の線積分を算出することが可能となる。**S** を画面上の任意の画素の空間中の位置に設定し、**E** を **S** から視線方向に十分離れた箇所に設定することで、その線上でのエネルギー値の合計を得ることができる。これをその画素の輝度値とすることで、エネルギー波形状のレンダリングを実現している。

第 3 章

エネルギー波生成手法

3.1 本章の概要

本章では、本研究におけるエネルギー波表現理論について述べる。理論の基本的な考え方は 2.2 節で述べた阿部手法とほぼ同様であり、まず、3 次元空間中にエネルギー分布関数を規定し、その上で視点から各画素を通る直線上のエネルギー量を線積分によって求めるというものである。

本研究は、阿部手法と比較して大きく以下のような差異がある。

- エネルギー分布関数として、阿部手法による点型と線型に加え、トーラス型と自由曲線型を実現した。
- 自由曲線型において、B-Spline 基底関数を用いて任意の部位の強弱を調整する機能を持つ。
- トーラス型と自由曲線型は解析的積分が非常に複雑になるため、数値積分による手法を用いる。

本章では、まず 3.2 節にてエネルギー分布関数について定義を行う。特に、阿部手法にはなかったトーラス型と自由曲線型について詳しく述べる。次に 3.3 節で自由曲線型における部位強弱調整手法について述べる。次に、3.4 節にて複数エネルギー波の加算方法について述べ、3.5 節にて数値積分による画素輝度値算出方法を述べる。

3.2 エネルギー分布関数

本手法では、3 次元空間中の任意の点におけるエネルギー値を、実数体 (the set of real number) $\mathbb{R}$ に対し式 (3.1) のスカラー場関数 ψ として表す。

$$\psi(\mathbf{P}) : \mathbf{P} \in \mathbb{R}^3,\ \psi(\mathbf{P}) \in \mathbb{R} \tag{3.1}$$

$\psi(\mathbf{P})$ の値が高いほど、その地点での発光輝度が高いという想定となるが、具体的な輝度値算出は 3.5 節で述べる。

本節では以降、まず阿部手法によって定義された分布関数に基づき、3.2.1 項にて点型分布関数、3.2.2 項にて線型分布関数の再定義を行う。次に、3.2.3 項においてトーラス型の分布関数の定義を、3.2.4 項にて 2 次曲線による分布関数の定義を述べる。

3.2.1 点型分布関数

阿部手法において、任意の一点 $\mathbf{M}$ より放射状に広がる分布関数は式 (2.1) によって定義されている。この式は、点 $\mathbf{P}$ が点 $\mathbf{M}$ に近くなるほどエネルギー値が高くなる関数である。ただし、$\mathbf{P}$ と $\mathbf{M}$ の距離 $|\mathbf{P}-\mathbf{M}|$ をそのまま分母としてしまうと値が無限大に発散してしまうという問題がある。

阿部手法においては、式 (2.1) のままであっても、この式のまま直接計算することはなく、プログラムによる演算を行う式は (2.8) による積分式のみであるため、問題は生じない。しかしながら、本研究ではこの式を直接プログラムで求める場面があるため、このままでは $\mathbf{P}$ と $\mathbf{M}$ が一致した場合にゼロ除算のエラーが生じてしまう。

そこで本手法では、同様の分布関数 $S(\mathbf{P})$ を微小値 ϵ を用いて以下の式 (3.2) で再定義した。

$$S(\mathbf{P}) = \frac{1}{\max\left(\epsilon,\ |\mathbf{P}-\mathbf{M}|\right)} \tag{3.2}$$

図 3.1 は、式 (3.2) によるエネルギー分布の模式図で、色が濃い部分ほど高いエネルギー値であることを表す。

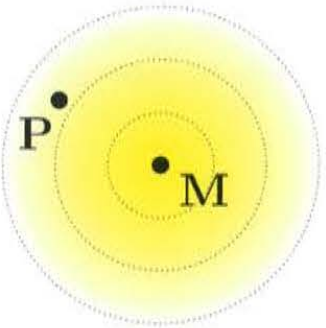


図 3.1　点型分布関数によるエネルギー分布

3.2.2　線型分布関数

線光源型の分布関数についても、阿部手法では式 (2.2) と定義されている。この式もこのままではゼロ除算の可能性が生じるため、本研究では微小値 ϵ を用いて以下のように再定義した。

$$C(\mathbf{P}) = \frac{1}{\max\left(\epsilon,\ \sqrt{|\mathbf{P}-\mathbf{M}|^2 - ((\mathbf{P}-\mathbf{M})\cdot\mathbf{D})^2}\right)} \tag{3.3}$$

ただし、$\mathbf{M}$ は直線上の任意の点、$\mathbf{D}$ は直線と平行な単位ベクトルである。

図 3.2 は、式 (3.3) によるエネルギー分布の模式図で、色が濃い部分ほど高いエネルギー値であることを表す。

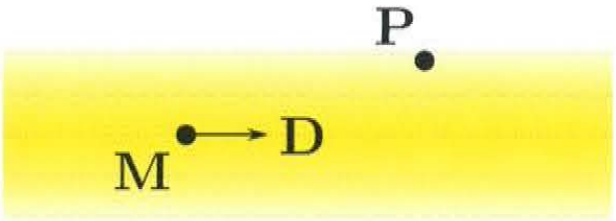


図 3.2　線型分布関数によるエネルギー分布

3.2.3　トーラス型分布関数

トーラス体とは、円柱形状の両底面を丸めて結合したような形状であり、日常的には「ドーナツ型」と呼ばれるような形状を指す。日本語では「輪環体」や「円環体」と呼ばれることもある。厳密には、3 次元空間中の xz 平面上に z 軸に交わらないように円を配置し、それを z 軸を中心とした回転してできる回転体として定義することができる [44]。

トーラス体の断面にあたる円を「小円」と呼ぶ。小円の半径を「小半径」と呼ぶ。また、小円の中心を z 軸中心に回転することによってえられる円を「大円」と呼び、大円の半径を「大半径」と呼ぶ。

図 3.3 は、トーラス体の模式図である。図中でトーラス体の断面図となっている青色の円が小円であり、r が小半径である。また、図中でトーラス内部にある赤色の円が大円であり、R が大半径である。

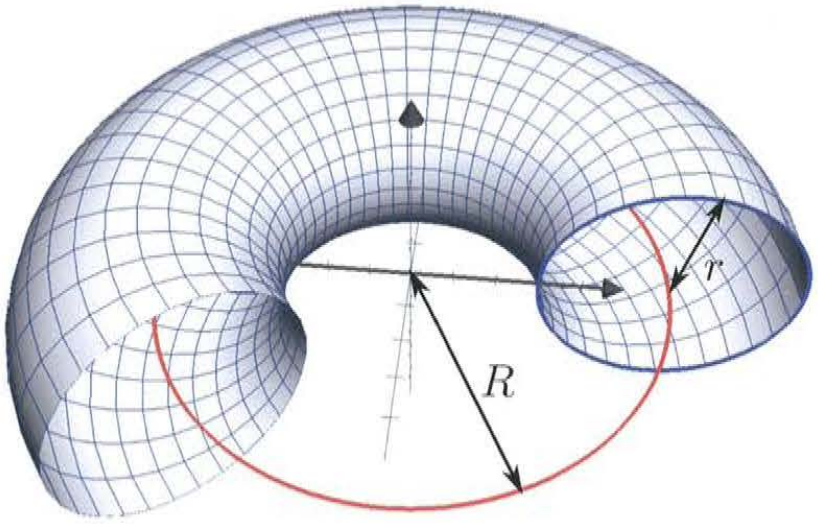


図 3.3　トーラス体の模式図

トーラス体の面の一般的な陰関数表現は、xz 平面上の半径 r の円の z 軸を中心とする回転体として定義することができる。これを式 (3.4) として定義する。

$$C : (x - R)^2 + z^2 = r^2 \quad (0 < r < R) \tag{3.4}$$

式 (3.4) の回転体の陰関数表現は以下の通りとなる。

$$T : \left(\sqrt{x^2 + y^2} - R\right)^2 + z^2 = r^2 \tag{3.5}$$

式 (3.5) を r について解くと、式 (3.6) となる。

$$r = \sqrt{\left(\sqrt{x^2 + y^2} - R\right)^2 + z^2} \tag{3.6}$$

この r の値は、トーラス型の中心曲線となる大円と 点 (x, y, z) との距離を意味する。これを利用し、トーラス型の分布関数を以下の式 (3.7) のように定めた。

$$T(\mathbf{P}, R) = \frac{1}{\max\left(\epsilon,\ \sqrt{\left(\sqrt{{P_x}^2 + {P_y}^2} - R\right)^2 + {P_z}^2}\right)} \tag{3.7}$$

式 (3.2) や式 (3.3) と同様、微小値 ϵ によって発散が生じないよう補正を行った。式 (3.7) の R の値を調整することにより、トーラス形状の大半径を制御できる。

この手法では、任意の点におけるエネルギーの影響は光源である円弧の最近点からの距離のみによって決定されることになる。これに対し、ある点におけるエネルギーの算出に円弧上の複数の点から影響を受けるような手法も考えられる。そのような手法では、大半径が大きくないトーラス型の中心付近では、円弧全体からエネルギー増加の影響を受けることになり、結果的にトーラス型特有の空洞が形成されずに楕円体形状をなしてしまうという問題が生じる。このような理由から、本手法では最近点からの距離のみを考慮することとした。図 3.4 にて、円弧状の最近点のみからの影響による場合の出力結果を左側に、複数箇所からの影響による場合の出力結果を右側に示す。

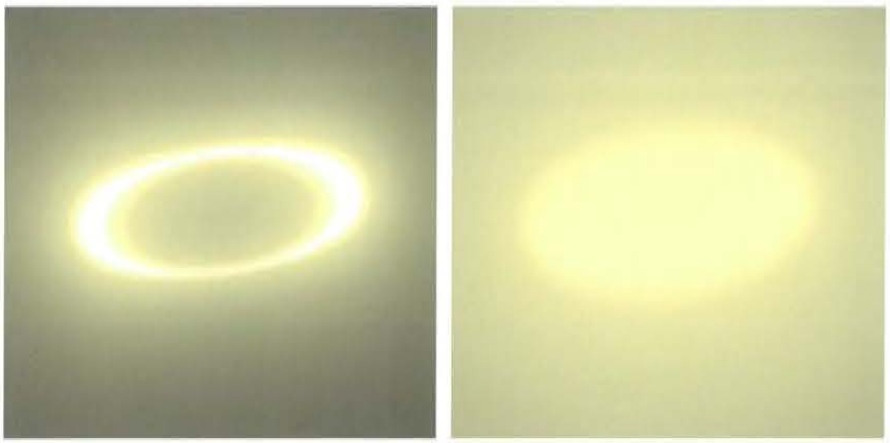


図 3.4　最近点のみの場合 (左) と複数点からの場合 (右)

3.2.4　2 次曲線による分布関数

本節では、2 次曲線を中心とした分布関数について述べる。これは、3 次元ベクトル列 $\{\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2\}$ とパラメータ t ($t \in \mathbb{R}$) を用いて、以下のような形式で表現できる曲線 $\mathcal{C}$ のことである。

$$\mathcal{C} : \mathbf{C}(t) = \mathbf{C}_2 t^2 + \mathbf{C}_1 t + \mathbf{C}_0 \tag{3.8}$$

一般的に「2 次曲線」と呼ばれる曲線のうち、円弧や双曲線などはこの形式では表現できないが、放物線の表現が可能である。また、自由曲線としては 2 次 Bézier 曲線がこの条件に当てはまる。制御点列 $\{\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2\}$ による 2 次 Bézier 曲線式は式 (3.9) となる。

$$\mathbf{C}(t) = (1-t)^2 \mathbf{B}_0 + 2t(1-t)\mathbf{B}_1 + t^2 \mathbf{B}_2 \tag{3.9}$$

この式を、式 (3.8) に当てはめると式 (3.10) となる。

$$\begin{cases} \mathbf{C}_2 = \mathbf{B}_0 - 2\mathbf{B}_1 + \mathbf{B}_2 \\ \mathbf{C}_1 = -2\mathbf{B}_0 + 2\mathbf{B}_1 \\ \mathbf{C}_0 = \mathbf{B}_0 \end{cases} \tag{3.10}$$

C^1 連続な曲線 $\mathcal{C}$ 上の点のうち、点 $\mathbf{P}$ と最近点となる点を $\mathbf{C}(\alpha)$ としたとき、点 $\mathbf{P}$ と $\mathbf{C}(\alpha)$ を通る直線と曲線 $\mathcal{C}$ は垂直に交わる。図 3.5 にその様子を示す。

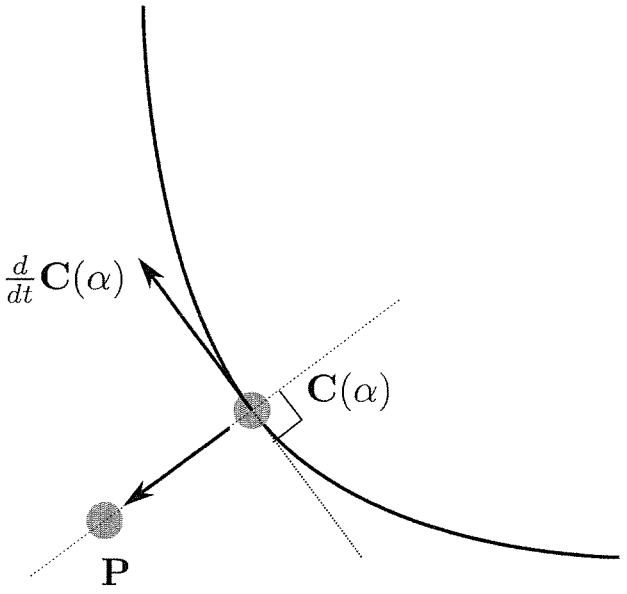


図 3.5　最近点の算出

これは、式 (3.11) が成り立つことを意味する。

$$\frac{d}{dt}\mathbf{C}(\alpha)\cdot(\mathbf{P}-\mathbf{C}(\alpha))=0 \tag{3.11}$$

従って、式 (3.11) を満たす実数解を求めることができれば、最近点の候補を算出することが可能となる。

式 (3.11) に式 (3.8) を代入すると、式 (3.12) のような 3 次方程式を得ることができる。

$$\begin{aligned}&\frac{d}{dt}\mathbf{C}(t)\cdot(\mathbf{P}-\mathbf{C}(t))=\\&\quad 2|\mathbf{C}_2|^2\,t^3+3(\mathbf{C}_1\cdot\mathbf{C}_2)\,t^2+\\&\quad \left(2(\mathbf{C}_0\cdot\mathbf{C}_2)+|\mathbf{C}_1|^2\right)\,t+(\mathbf{C}_1\cdot(\mathbf{C}_0-\mathbf{P}))\end{aligned} \tag{3.12}$$

3 次方程式の解は一般的に代数的に求めることができることが知られており、数値解析に頼ることなく厳密解を求めることが可能である。

本手法では、以下のステップにより任意点 $\mathbf{P}$ と曲線 $\mathcal{C}$ の距離を算出した。

1. 式 (3.12) を解き、実数解を得る。得られた実数解を $\{t_1,t_2,t_3\}$ とする。ただし、t_2,t_3

は存在しない場合もある。これらの実数解のうち、曲線の定義域 (Bézier 曲線であれば $0 \leq t \leq 1$) にあるもののみを抽出する。

2. ステップ 2 によって得られた各 t_i $(i = 1, 2, 3)$ に対し、曲線上の点 $\mathbf{C}(t_i)$ を得る。
3. 曲線の両端点 (Bézier 曲線であれば $\mathbf{C}(0), \mathbf{C}(1)$) を得る。
4. ステップ 3 とステップ 4 で得られた各点と点 $\mathbf{P}$ との距離を算出する。
5. ステップ 5 で得た各距離のうち最小値を点 $\mathbf{P}$ と曲線 C の距離とする。

なお、本手法では 3 次方程式の解を求めるのにカルダノの公式 [45] を用いた。以下に、カルダノの公式によって 3 次方程式 $x^3 + ax^2 + bx + c = 0$ の実数解を求める方法を述べる。ここではあくまで求解に必要な最低限の処理のみを述べるものである。詳細については参考文献 [45] を参照のこと。

まず、以下のようにして実数 p, q, r を求める。

$$p = -\frac{a^2}{9} + \frac{b}{3} \tag{3.13}$$

$$q = \frac{a^3}{27} - \frac{ab}{6} + \frac{c}{2} \tag{3.14}$$

$$r = p^3 + q^2 \tag{3.15}$$

次に、以下のようにして複素数 u, v を求める。

$$u = \sqrt[3]{-q + \sqrt{r}} \tag{3.16}$$

$$v = \sqrt[3]{-q - \sqrt{r}} \tag{3.17}$$

複素数 u, v をプログラムで求める場合、r の正負で処理がかわる。

- r が正の場合: u, v いずれも実数となる。
  1. $s = -q + \sqrt{r}$ を求め、$s \geq 0$ の場合 $u = \sqrt[3]{s}$、$s < 0$ の場合は $u = -\sqrt[3]{-s}$ とする。
  2. $t = -q - \sqrt{r}$ を求め、$t \geq 0$ の場合 $v = \sqrt[3]{t}$、$t < 0$ の場合は $v = -\sqrt[3]{-t}$ とする。

- r が負の場合: u, v いずれも複素数となる。

  1. 複素数 z を $z = -q + \sqrt{-r}\, i$ とする。
  2. z の偏角 $\theta = \arctan\left(\frac{q}{r}\right)$ を求めておく。
  3. $u = \sqrt[3]{z} = \sqrt[3]{|z|}(\cos\frac{\theta}{3} + i\sin\frac{\theta}{3})$ を求める。
  4. $v = \overline{u}$ を求める。

次に、複素数 $\omega_1, \omega_2, \omega_3$ をそれぞれ以下のように定義する。

$$\omega_1 = e^0 = 1 \tag{3.18}$$

$$\omega_2 = e^{\frac{2\pi i}{3}} = -\frac{1}{2} + \frac{\sqrt{3}}{2}\, i \tag{3.19}$$

$$\omega_3 = e^{\frac{4\pi i}{3}} = -\frac{1}{2} - \frac{\sqrt{3}}{2}\, i \tag{3.20}$$

これらから複素数 $u_1, u_2, u_3, v_1, v_2, v_3$ を以下の式で求める。

$$\begin{array}{lll} u_1 = \omega_1 u, & u_2 = \omega_2 u, & u_3 = \omega_3 u, \\ v_1 = \omega_1 v, & v_2 = \omega_2 v, & v_3 = \omega_3 v \end{array} \tag{3.21}$$

最終的な複素数も含めた 3 次方程式の解は、以下の式 (3.22) で得られる 9 個の複素数のうち、いずれか 3 個となる。

$$\begin{array}{lll} u_1 + v_1, & u_1 + v_2, & u_1 + v_3, \\ u_2 + v_1, & u_2 + v_2, & u_2 + v_3, \\ u_3 + v_1, & u_3 + v_2, & u_3 + v_3 \end{array} \tag{3.22}$$

ただし、本手法においては複素数解は必要ではなく、実数解だけを求めれば良いという条件がある。3 次方程式が実数解を持つ場合、それらの解は上記 9 個のうち実数であるものと完全に一致するので、虚部が 0 であるものだけを抽出することで、目的を達成することができる。

この処理による距離算出関数を $L(\mathbf{P}, \mathcal{C})$ とし、2 次曲線によるエネルギー分布関数 $Q(\mathbf{P}, \mathcal{C})$ を以下のように定義した。

$$Q(\mathbf{P}, \mathcal{C}) = \frac{1}{\max\left(\epsilon,\ L(\mathbf{P}, \mathcal{C})\right)} \tag{3.23}$$

3.2.3 節で述べたトーラス型分布関数と同様に、本手法では自由曲線においても最近点からの影

響のみを考慮している。なぜならば、曲線上の複数点からエネルギーの影響を受けるような分布にした場合、曲線の端の近辺ではエネルギー値が低くなってしまうことや、曲率が高い曲部の内側ではエネルギー値が高くなるという特性が生じるためである。すなわち、エネルギー波の形状があまり曲線に沿ったものではなくなることから、元となる曲線形状と入力パラメータ値による描画結果の予測が難しくなるという問題が生じるためである。図 3.6 にて、同じ曲線形状に対し、最近点のみからの影響による場合の出力結果を左側に、曲線全体からの影響による場合の出力結果を右側に示す。

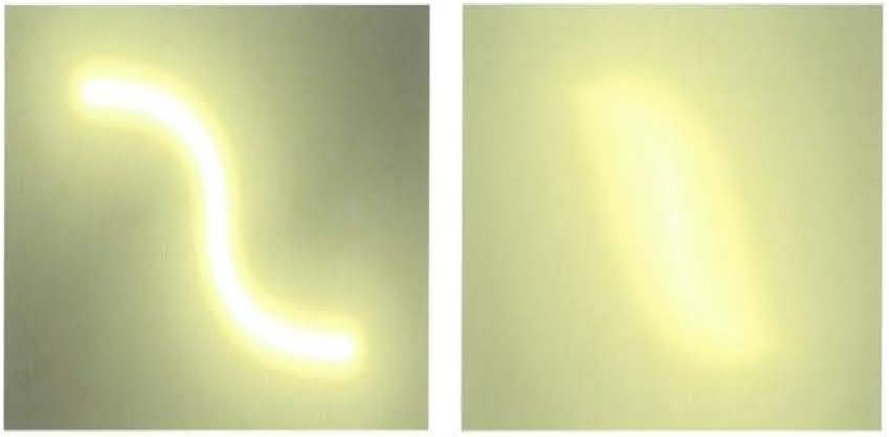


図 3.6　最近点のみの場合 (左) と複数点からの場合 (右)

図 3.6 の左側は、エネルギー波形状がほぼ元曲線に沿った形状となっているのに対し、右側は元曲線の両端点部分でエネルギー波形状がしぼんでしまっていることや、元曲線において曲がっている部分の内側にあたる部位でエネルギー波が膨らんだ状態となっている。本手法では、左側の状態の方がエネルギー波形状のデザインを容易に行えるものと考え、最近点のみを考慮することとした。

3.3 B-Spline 基底関数による部位の強弱調整

3.2.4 節で述べた分布関数 $Q(\mathbf{P},\mathcal{C})$ を用いた分布は、曲線全体に対し一様に分布した状態となる。これに対し、曲線の部位毎にエネルギーの強弱を調整できる機能があると、より自由度の高い表現が可能となる。例えば、エネルギー波が曲線に沿って移動するといった表現が容易に実現できるようになる。図 3.7 は、部位毎に調整した様子の模式図である。

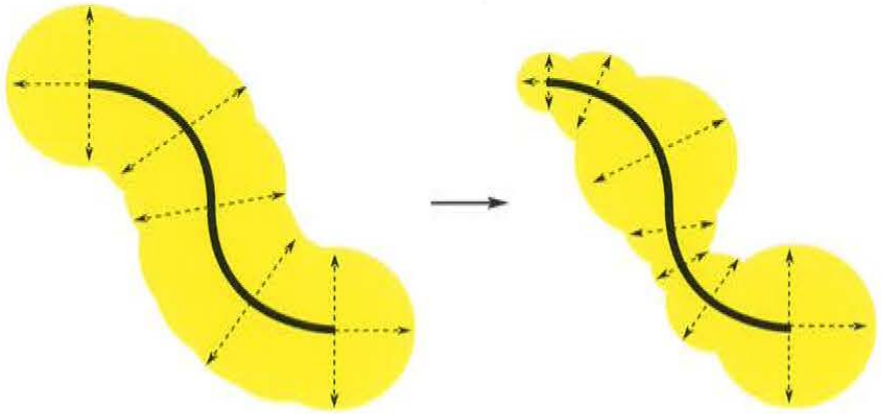


図 3.7　曲線部位毎の調整

これは先行研究 [2] の手法でも、球状や円柱状の分布関数を組み合わせることで理論的に表現は可能であるが、その場合多数の分布関数を組み合わせる必要があり、実行速度が低下するという問題が生じる。

本節では、分布関数 $Q(\mathbf{P},\mathcal{C})$ に調整用の基底関数を組み合わせる手法を述べる。この手法を用いると、単一の分布関数内で調整が完結するため、実行速度に影響することがほとんどなく、高い表現能力を高速な処理で実現することを可能とする利点がある。

部位毎のエネルギー値増減調整を行う方法として、本手法では B-Spline 基底関数 [46] を利用した。B-Spline 基底関数を用いる利点は、

1. 関数連続性について既に十分研究がなされており、微分連続性を保ったエネルギー分布関

数を実現できる。

2. 実数の四則演算のみで算出が可能であり、計算時間が高速である。

3. Bézier 基底関数では不可能な局所的な制御ができ、柔軟な表現が可能となる。

といった点が挙げられる。

B-Spline 基底関数は、単調増加数列 $U = [u_0, u_1, \ldots, u_{m-1}]$ と 次数 k により、以下の式 (3.24) によって定義される。

$$
\begin{aligned}
N_i^0(t) &= \begin{cases} 1 & if\ u_i \leq t < u_{i+1} \\ 0 & else \end{cases}, \quad i = 0, \ldots, m-2 \\
N_i^k(t) &= \frac{t - u_i}{u_{i+k} - u_i} N_i^{k-1}(t) - \frac{t - u_{i+k+1}}{u_{i+k+1} - u_{i+1}} N_{i+1}^{k-1}(t), \quad i = 0, \ldots, m-k-2
\end{aligned}
\tag{3.24}
$$

U を「ノットベクトル」と呼ぶ。加えて、曲線の部位ごとの強弱度合いを表す数列 $W = [w_0, w_1, \ldots, w_{n-1}]$ により、調整用関数 $S(t)$ を以下のように定義する。

$$
S(t) = \sum_{i=0}^{n-1} w_i N_i^k(t) \tag{3.25}
$$

本手法では、計算速度をできるだけ高速にすることと、各 w_i の影響範囲を狭め、局所的な調整をやりやすくするため、次数 k を 2 とした。

調整用数列 W は可変長であり、要素数 n によって曲線のパラメータ領域 $[0, 1]$ を等分割し、w_i は曲線パラメータが $\frac{i}{n-1}$ となる部分のエネルギー強弱を調整することになる。その際、ノットベクトルは式 (3.26) となるように設定する。

$$
U = \left[0,\ 0,\ 0,\ \frac{1}{n-2},\ \frac{2}{n-2},\ \ldots,\ \frac{n-3}{n-2},\ 1,\ 1,\ 1\right] \tag{3.26}
$$

図 3.8 は、$W = [1.0, 0.6, 0.2, 0.4, 0.8, 0.6, 1.0]$ としたときの $S(t)$ の変化をグラフで表したものである。

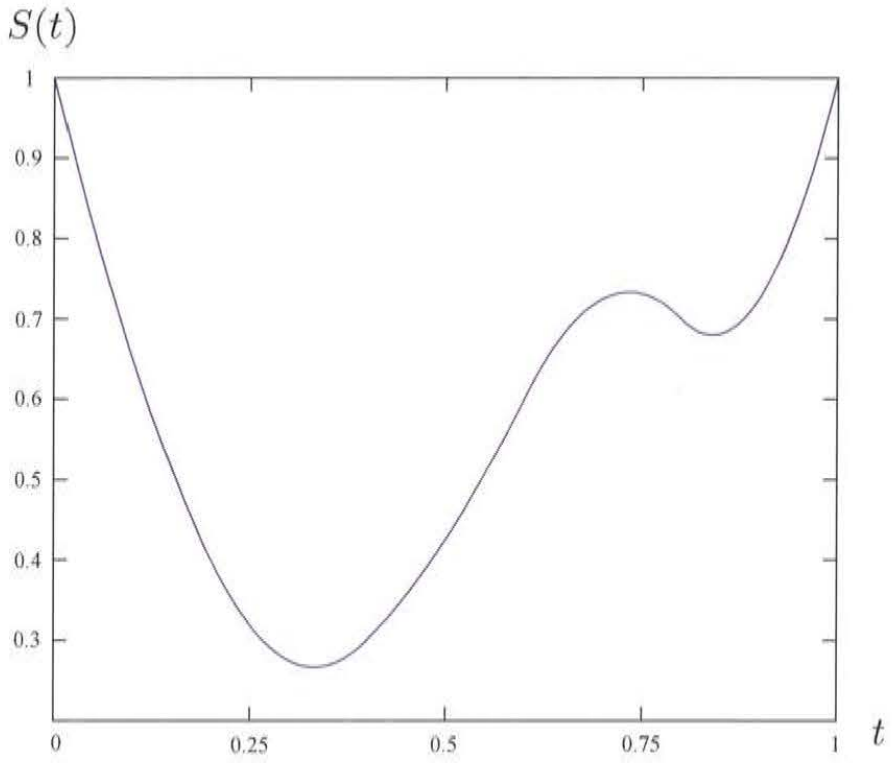


図 3.8　$S(t)$ のグラフ

最後に、式 (3.23) と組み合わせ、部位毎の強弱調整を可能とする分布関数 $B(\mathbf{P},\mathcal{C})$ を式 (3.27) として定義する。

$$B(\mathbf{P},\mathcal{C}) = S(t) \cdot Q(\mathbf{P},\mathcal{C}) \tag{3.27}$$

ここで、t は $\mathcal{C}$ 上の $\mathbf{P}$ からの最近点での曲線パラメータである。

3.4　複数エネルギー波形状の加算

最終的なスカラー場関数 $\psi(\mathbf{P})$ は、これまで述べてきた (3.2), (3.3), (3.7), (3.23) の各式による分布関数の総和で求められる。

$$\psi(\mathbf{P}) = \sum_i s_i S_i(\mathbf{P}) + \sum_i c_i C_i(\mathbf{P}) + \sum_i t_i T_i(\mathbf{P}) + \sum_i b_i B_i(\mathbf{P}) \tag{3.28}$$

s_i, c_i, t_i, b_i はそれぞれの分布関数に対する実数係数で、これらを以降「エネルギー係数」と呼称するものとする。この値を変更することによって分布関数の影響力を調整することができる。

3.5　テクスチャ画像の生成

本節では、これまで述べてきたエネルギー分布関数を利用し、出力するテクスチャ画像の各画素の輝度値を決定する手法について述べる。まず 3.5.1 項にて画素輝度値算出の概要を述べる。3.5.2 項では輝度値算出の際に用いる積分区間の決定方法について述べる。続く 3.5.3 項では、数値積分による輝度値算出手法について述べる。

3.5.1　画素輝度値算出の概要

本手法のレンダリングは、基本的にはボリュームレンダリング手法として一般的なレイキャスティング法を用い、空間中のビルボードテクスチャに対しデータを動的に生成することでレンダリングを実現する。視点位置から投影面に対して各画素ごとに視線を飛ばし、その視線方向に沿ってエネルギーの分布量を算出し、各線でのエネルギー値の合計を輝度値として設定する。

エネルギー分布関数は全て連続なスカラー場関数として定義されているため、視線直線上のエネルギー分布総量の算出は線積分を用いることで算出が可能である。図 3.9 にその概念図を示す。図 3.9 の右側のグラフは、左側の視線直線に沿ってエネルギー値をグラフ化したものであり、視線直線上のエネルギー値の総量は、グラフ内の黄色部分の面積となり、積分によって算出することが可能である。

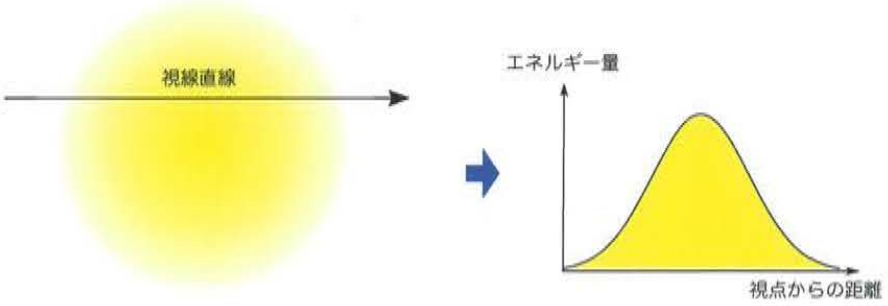


図 3.9　線積分によるエネルギー総量算出の概念図

ただし、エネルギー値をそのまま積分した場合、光の距離による減衰が表現できないため、本研究では視線からの距離による減衰を補正する関数をエネルギー分布関数と併用して各画素の輝度値算出を行う。

3.5.2 積分区間の決定

投影面の任意の画素の空間上での位置ベクトルを $\mathbf{S}$、視点から $\mathbf{S}$ に対し視線を飛ばし、視点から見てある程度離れた距離にある視線上の点を $\mathbf{E}$ とする。$\mathbf{S}$ と $\mathbf{E}$ を結ぶ半直線の範囲内が積分区間となる。この半直線 $\mathcal{L}$ を、パラメータ s によって以下の式 (3.29) のように表す。

$$\mathcal{L} : \mathbf{L}(s) = (1-s)\mathbf{S} + s\mathbf{E} \qquad (0 \leq s \leq 1) \tag{3.29}$$

$\mathbf{E}$ は、$\mathbf{S}$ から視線上にあり一定距離離れた箇所に定める。$\mathbf{E}$ の位置はエネルギー波形状よりも十分離れた位置に置く必要があるが、$\mathbf{E}$ の位置が $\mathbf{S}$ から離れるほど計算精度は落ちることになり、不適切な描画がなされる可能性がある。実験の結果、投影面からエネルギー波形状の中心位置に対し、その 4 倍程度の距離に $\mathbf{E}$ を置くことで、概ねの場合で良好な出力結果が得られた。

3.5.3 輝度値計算のための積分式

輝度値の計算は、以下の式 (3.30) によって求められる。

$$\int_0^1 \psi\left(\mathbf{L}(s)\right) \zeta(s) ds \tag{3.30}$$

ここで、ψ は式 (3.28) で述べたエネルギー分布を表すスカラー場関数で、ζ は距離による減衰を実現するための補正関数である。

一般的に、視点に到達する光の強さは光源からの距離の 2 乗に反比例する。

ζ によってその効果を実現する。今回は減衰関数として、式 (3.31) を用いた。

$$\zeta(s) = \frac{\alpha}{(s+\beta)^2} \tag{3.31}$$

α, β はそれぞれ調整用の実数定数である。本論文中の評価実験では、全ての場合で $\alpha = 1, \beta = 1$ を用いている。

式 (3.30) で表される関数式を解析的に求めることは、式 (3.30) は初等関数の範疇で求めることができない関数が出現することや、数百項に渡るような複雑な式となってしまうため、演算時間を考えるとあまり現実的ではない。そのため、本手法においては関数 $f(s)$ を以下のようにおく。

$$f(s) = \psi\left(\mathbf{L}(s)\right)\zeta(s) \tag{3.32}$$

この $f(s)$ に対し、積分値を近似的に求めるシンプソンの公式を適用した。

$$\int_0^1 f(s)ds \approx \frac{1}{3n}\left(f(0) + 2\sum_{j=1}^{\frac{n}{2}-1} f\left(\frac{2j}{n}\right) + 4\sum_{j=1}^{\frac{n}{2}} f\left(\frac{2j-1}{n}\right) + f(1)\right) \tag{3.33}$$

n は分割数を表し、大きいほど精度がよくなる。本手法においては、200 程度で描画に十分な精度が得られることを確認した。図 3.10 に上記処理の概念図を示す。

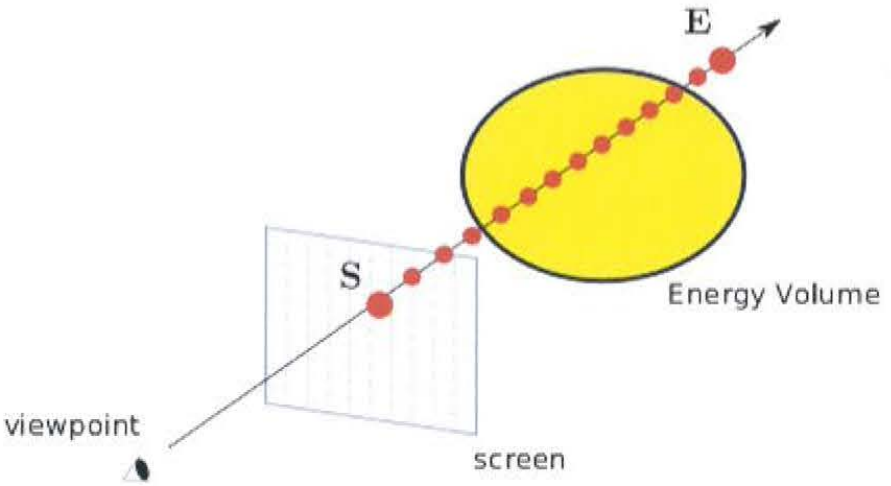


図 3.10　輝度値算出の概念図

第 4 章

GPGPU を用いた手法実装

4.1 本章の概要

本章では、3 章で述べた理論を元に、実装手法について述べる。

3 章における理論は画素ごとに個別に行われるものであるため、膨大な計算量が必要となる。従来の逐次処理による実装では一回当たりの出力画像作成に数秒から数十秒を要するものであり、リアルタイムグラフィックスへの適用は現実的ではない。そこで本研究では、GPGPU (General Purpose computing on GPU) 技術を用い、画素ごとの演算を並列に行うことで高速な演算を実現し、リアルタイムグラフィックスでの実用に適応できる処理速度を実現した。本章では、特に並列演算を念頭においた実装設計についてを述べていく。

本章では、まず 4.2 節で GPGPU についての概略と本研究のおける GPGPU 環境の解説を述べる。4.3 節では CPU と GPU 間のデータ転送に用いる配列の定義を行う。4.4 説では各画素ごとに線積分を行う際の、線分の始点と終点の算出手法について述べる。4.5 節では、数値積分を行う際の分割点算出方法について述べる。続く 4.6 節では、3.5 節で述べた数値積分手法に基づき、線積分値を算出する手法について述べる。最後に 4.7 節で GPGPU による一連の処理の流れをフローチャートとして図式化する。

4.2 GPGPU について

本手法では、GPU (Graphics Processing Unit) 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 3 次元座標変換や描画処理を専門に行うユニットであるが、近年では CPU の代わりに汎用的な演算装置として利用することも多い。GPU が持つ演算コアは、CPU のコアと比較すると単体では性能的に劣り、特に分岐処理や再帰処理には不向きである。しかしながら、近年の GPU は演算コアを数百個から数千個程度搭載しており、単純な処理を並列に行う場合には劇的に速度が向上する。

本手法においては、描画処理の際に 1 画素ごとに式 (3.30) を実行することになるが、CPU 演算のみによる処理でのリアルタイムレンダリングは現時点では不可能である。そこで、本手法は式 (3.30) を 1 つの処理とみなし、同時に複数の画素輝度計算を GPGPU を用いて並列処理を行うことで処理速度を向上した。

GPGPU 上での処理を実現する技術としては、有名なものに HLSL や GLSL といったシェーダー言語によるものや、NVIDIA の CUDA、そして Khronos Group による OpenCL がある。シェーダーは描画処理に特化されている言語であるため、データをテクスチャデータとして変換するなどの工夫が必要となる。CUDA は GPGPU として最も有名なシステムであるが、NVIDIA 製の GPU でしか実行できないという問題がある。これらの事情から、今回我々は OpenCL を採用した。

4.3 配列の準備

本手法では、まず空間中のエネルギー値を格納する 3 次元配列 $A^{i,j,k}$ と、画素ごとの輝度値を格納する 2 次元配列 $B^{i,j}$ を用意する。配列の大きさは A, B ともに第 1 成分が出力画像の横幅画素数、第 2 成分が縦幅画素数である。本手法では計算を簡略化するため、出力画像の縦幅数と横幅数は共に同一とした。この幅数を以降 W とする。

また、A の第 3 成分の大きさは、積分計算を近似的に求める際の分割数となる。これを以降 d とする。

4.4 各画素の空間座標算出

最初に、各画素の空間座標を算出する必要がある。まずテクスチャ画像の各画素に対し、番号を (i, j) で割り振る。次に、テクスチャ平面内での正規化 2 次元座標 $\mathbf{T}^{i,j}$ を式 (4.1) による算出

する。

$$\mathbf{T}^{i,j} = \left(\frac{i}{W}, \frac{j}{W} \right) \tag{4.1}$$

次に、各画素の 3 次元空間中での位置ベクトル $\mathbf{S}^{i,j}$ を式 (4.2) により算出する。

$$\mathbf{S}^{i,j} = \mathbf{M} \begin{pmatrix} -\frac{l}{2} + l \cdot {T^{i,j}}_x \\ -\frac{l}{2} + l \cdot {T^{i,j}}_y \\ 0 \\ 0 \end{pmatrix} + \mathbf{C} \tag{4.2}$$

ここで、$\mathbf{M}$ はテクスチャ画像のモデリング変換行列、l はテクスチャ画像の空間中での辺の長さ、$\mathbf{C}$ はテクスチャ画像の空間中での中心位置ベクトルである。この $\mathbf{S}^{i,j}$ が、式 (3.29) の $\mathbf{S}$ に相当することになる。図 4.1 に、$\mathbf{T}^{i,j}$ から $\mathbf{S}^{i,j}$ への変換の模式図を示す。

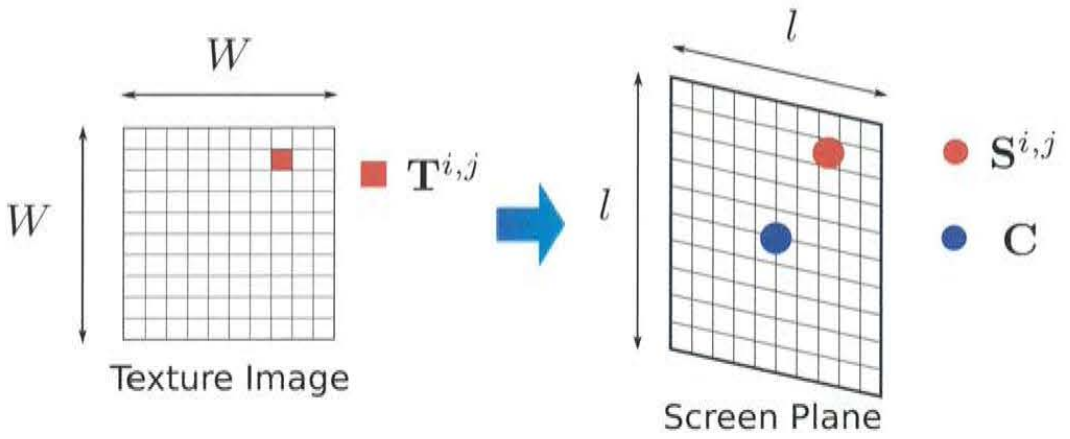


図 4.1　テクスチャ座標系から空間座標系への変換

次に、式 (4.2) で求めた $\mathbf{S}^{i,j}$ と視点位置ベクトル $\mathbf{V}$ から、式 (3.29) における $\mathbf{E}$ を求める。これを $\mathbf{E}^{i,j}$ とすると、以下の式 (4.3) によって求められる。

$$\mathbf{E}^{i,j} = \nu \mathbf{S}^{i,j} - \mathbf{V} \tag{4.3}$$

図 4.2 に、$\mathbf{E}^{i,j}$ 算出の模式図を示す。

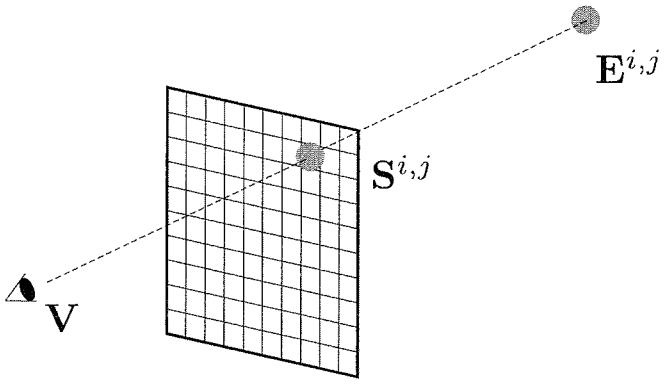


図 4.2　線積分範囲の算出

ν が大きくなるほど $\mathbf{E}$ は遠くなりより広い範囲を描画に反映することができるが、積分近似精度が悪くなるという問題があり、状況により適切な値を設定する必要がある。

なお、5 章での検証結果では、全て ν の値を 2 として出力を行った。

4.5　数値積分に用いる分割点座標の算出と保存

式 (4.2) と式 (4.3) を式 (3.29) に代入することにより、視線のベクトル方程式を求めることができる。視線は各画素ごとに生じることになり、これを式 (4.4) として定義する。

$$\mathbf{L}^{i,j}(s) = (1-s)\mathbf{S}^{i,j} + s\mathbf{E}^{i,j} \tag{4.4}$$

式 (3.28) の ψ 関数と式 (3.31) の ζ 関数、および $\mathbf{L}^{i,j}(s)$ を用いて、輝度値配列 A を以下の式 (4.5) によって求める。

$$A^{i,j,k} = \psi\left(\mathbf{L}^{i,j}(s)\right)\zeta(s) \qquad \left(s = \frac{k}{d},\ k = 0, 1, \ldots, d-1\right) \tag{4.5}$$

4.6　画素輝度値算出

テクスチャ画像の各画素輝度 $B^{i,j}$ は、式 (3.30) を適用することにより、式 (4.6) によって求められる。

$$B^{i,j} = \int_0^1 \psi\left(\mathbf{L}^{i,j}(s)\right)\zeta(s)ds \tag{4.6}$$

式 (4.6) に対し、式 (3.33) で紹介したシンプソンの公式に当てはめ、式 (4.7) によって近似値の算出を行う。

$$B^{i,j} = \frac{1}{3d}\left(A^{i,j,0} + 2\sum_{k=1}^{\frac{d}{2}-1} A^{i,j,2k} + 4\sum_{k=1}^{\frac{d}{2}} A^{i,j,2k-1} + A^{i,j,d-1}\right) \tag{4.7}$$

最終的なテクスチャ画像の画素色 $C^{i,j}$ は、エネルギー波の色を C_e, 背景色を C_b としたときに、式 (4.8) を用いて算出した。

$$C^{i,j} = (1 - B^{i,j})C_b + B^{i,j}C_e \tag{4.8}$$

ここで、$B^{i,j}$ は 1 以上の値を示す場合もありえるものとし、もし画素の RGB 要素のいずれかが最大値を超えてしまう場合は、最大値に丸めるものとした。各画素は $B^{i,j}$ が 1 を超えた場合に C_e よりもより白に近い色になることになり、いわゆる「白飛び」の効果が生じ、エネルギー波が眩しく輝く効果を実現できる。

4.7 GPGPU による処理の流れ

本手法では、これまで述べた一連の処理に対し、CPU と GPU での処理分担を以下のように実装した。

1. 各種パラメータの設定を CPU 側のプログラム (C++) で取得する。
2. パラメータデータを GPU に渡す。
3. GPU の各演算ユニットは CPU からパラメータデータを受け取り、各画素ごとに式 (4.2), (4.3), (4.5) を実行し (OpenCL)、配列 A に演算結果を格納する。

4. GPU の各演算ユニット各画素ごとに式 (4.7) を実行し (OpenCL)、配列 B に演算結果を格納する。

5. 配列 B データを CPU に渡す。

6. CPU 内で式 (4.8) を実行し (C++)、テクスチャデータを作成する。

これを描画するごとに行うことで、エネルギー波描画のリアルタイム処理を実現した。

本研究では、上記手順 3 番にあたる処理と 4 番にあたる処理は別々の実行関数として実装を行った。理論的には、この 2 つの処理は一続きの実行関数として実装することは可能である。しかしながら、今回検証を行った環境においては GPU の実行関数一つあたりの処理ステップ数には限りがあり、一続きで実装した場合には正常な動作が行われないという問題が生じた。そのため、本研究では上記 2 つの手順を別々の関数として実装し、別個に処理を行うことで一連の処理を実現した。

これらの処理をフローチャートとして表した図を図 4.3 に示す。

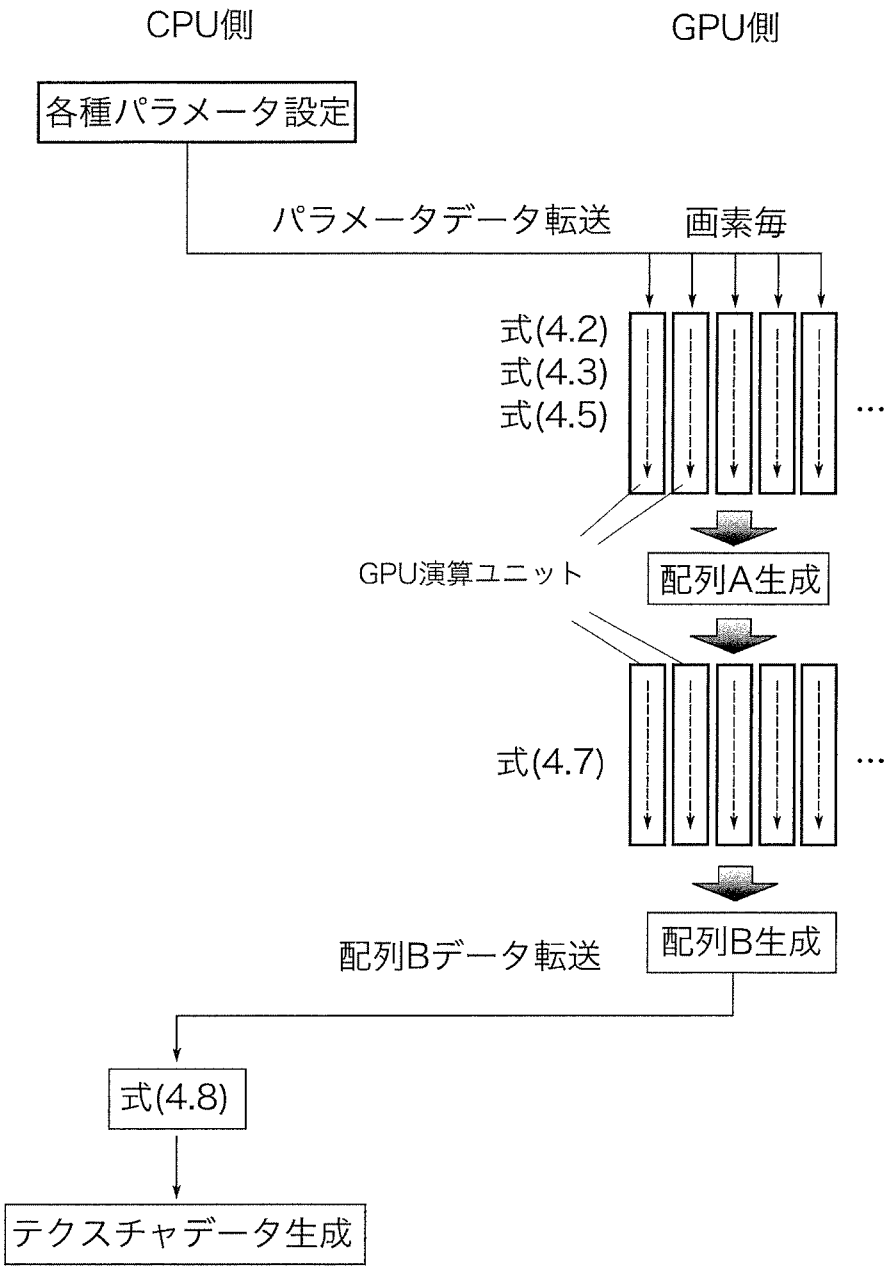


図 4.3　メインループ内のフローチャート

第 5 章

検証

5.1 本章の概要

本章では、本手法での検証結果について述べる。5.2 節では検証に用いた実行環境について述べる。5.3 節では実行速度に関する検証結果について述べる。5.4 節では本手法による出力結果の分析を述べる。5.5 節では、近似積分の際の分割数が出力にどのように影響を及ぼすかについての分析を述べる。最後に 5.6 節で、本手法の問題に対する改善点の考察について述べる。

5.2 検証環境

これまで述べた手法に基づき、以下の表 5.1 に挙げたスペックの PC 上で描画および速度の検証を行った。

表 5.1 検証用 PC のスペック

Machine	MacPro (Late 2013)
CPU	3.7Ghz Quad-Core Intel Xeon E5
Memory	16GB 1866Mhz DDR3 ECC
GPU	AMD FirePro D500 3GB Memory

FirePro D500 のストリーム・プロセッサー数は 1526 個である。

本研究では、基本的な開発や検証は AMD 製 GPU を搭載した MacOSX 上で行ったが、本手法による実装は NVIDIA 製 GPU を搭載した Windows7 上でほぼ同速度での実行が可能であることを確認している。

5.3 実行速度

投影面とサンプリング分割数が異なる状況での速度検証を行った。投影面は 128x128, 256x256, 512x512 の 3 種類の解像度を想定し、サンプリング分割数は 64、128、256 の 3 種類を想定した。これらの条件の元、1 秒あたりの再描画回数 (frame per second) を計測した。以下の表 5.2, 5.3

にその結果を示す。

表 5.2　速度検証結果 (トーラス型)

解像度	64 分割	128 分割	256 分割
128x128	384.61	168.63	63.25
256x256	116.69	46.49	16.85
512x512	31.23	13.49	5.21

表 5.3　速度検証結果 (Bézier 曲線)

解像度	64 分割	128 分割	256 分割
128x128	116.69	60.17	28.94
256x256	34.64	17.16	8.31
512x512	9.00	4.40	2.15

一般的に、リアルタイムレンダリングでは 60 FPS 以上の速度を必要とする。それを踏まえると、トーラスについては 256x256、Bézier 曲線については 128x128 の解像度にて現実的に利用できる処理速度であると言える。近年の PC 用モニタの画素数と比較するとこれらの解像度はかなり低いと言えるが、低解像度のテクスチャを補間して高精細な出力を可能とする 3D 技術 [47] も発展しており、特に隣接する画素同士の色変化が大きくない画像においては高い効果を発揮する。本手法における出力はその条件に当てはまるため、画像補間技術を併用することにより実用上での利用は可能であると考えられる。

5.4　出力結果

本節にて描画出力結果を示す。描画解像度は全て 256x256 とした。

図 5.1, 図 5.2 は、トーラス型エネルギー分布描画結果を 3 種類の角度で並べた物である。キャプション内の数値は式 (3.7) の R と、式 (3.28) でトーラス型のエネルギー係数にあたる t_i の値である。

図 5.1　トーラス型エネルギー分布の描画 $(R = 0.9, t_i = 5.1)$

図 5.2　トーラス型エネルギー分布の描画 $(R = 3.0, t_i = 7.4)$

図 5.3, 図 5.4 は、2 本の Bézier 曲線を連続に接続したエネルギー分布描画結果を、3 種類の角度で並べた物である。キャプション内の数値は式 (3.28) で 2 次曲線型のエネルギー係数にあたる b_i の値である。

図 5.3　Bézier 曲線型エネルギー分布の描画 $(b_i = 5.0)$

図 5.4　Bézier 曲線型エネルギー分布の描画 ($b_i = 7.5$)

図 5.5 は、図 5.3 の状態から動的に制御点を移動によって形状を変更したものである。

図 5.5　エネルギー分布の形状変更

本手法では、毎回の描画時に視点情報やパラメータから画像を再構築しているため、視点移動やパラメータ変更によって処理時間が長くなることはないことが特長である。

3.3 節で述べた機能を用いて、エネルギー分布を部位毎に強弱を調整した様子を示す。図 5.6 は、エネルギー係数 b_i が 8.0 である自由曲線型エネルギー波において、左側が全部位で最大強度であるものに対し、右側は調整数列を $W = [1.0, 0.2, 0.6, 1.0, 0.5, 0.8, 1.0]$ として描画したものである。

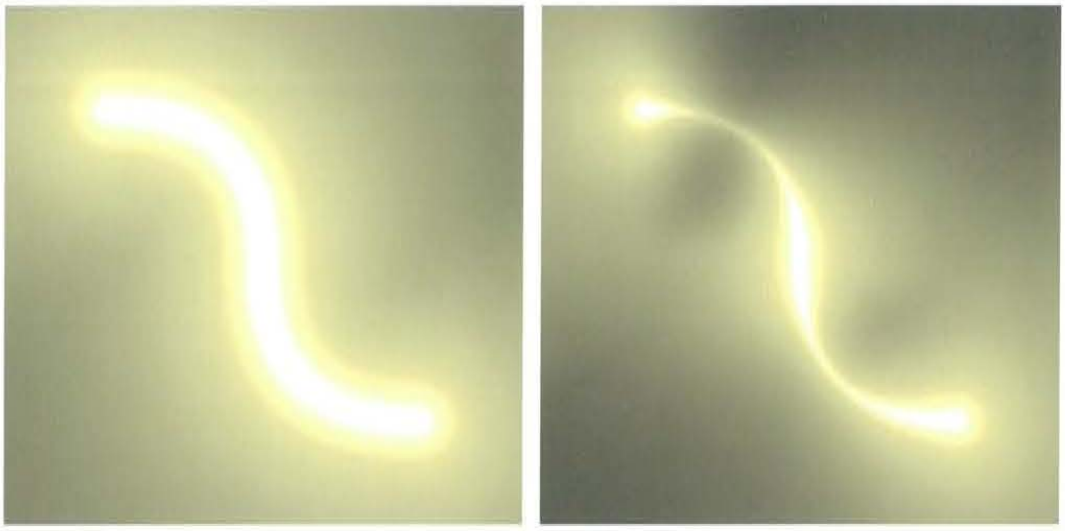
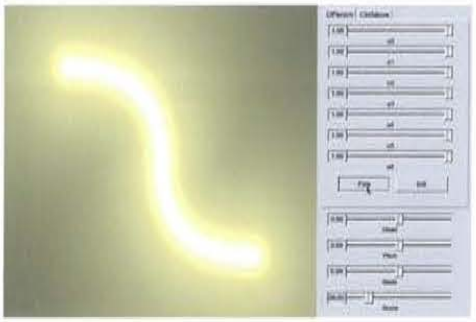


図 5.6　調整前と調整後の様子

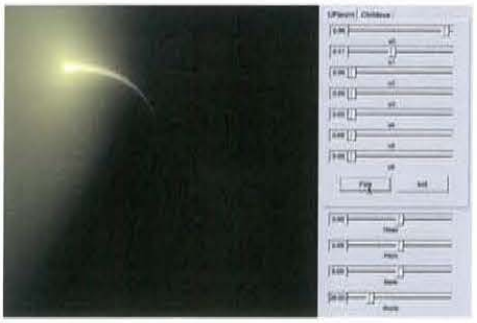
調整数列の各項に対し、タイミングをずらして周期的に変化させることにより、エネルギー波が曲線に沿って高速に移動する表現を実現することができる。図 5.7 でその段階別の描画状態を示す。描画フレーム $t = 0, 1, \ldots$ に対し、調整数列の各項 w_i を以下の式 (5.1) として定めている。

$$\begin{aligned} \theta_i &= \frac{(t - 7i)}{25}\pi \\ \theta_i{}' &= \begin{cases} \theta_i & if\ 0 \leqq \theta_i \leqq 2\pi \\ 0 & else \end{cases} \\ w_i &= \min\left(1, \max\left(0, \frac{-\cos\theta_i{}' + 0.8}{1.5}\right)\right) \end{aligned} \tag{5.1}$$

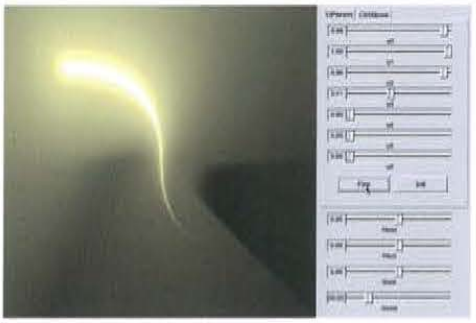
初期状態が全ての調整数列項で 1 であるのに対し、段階的に調整数列を式 (5.1) に従って定め、段階的に変化する様子を (a) から (e) までで示している。



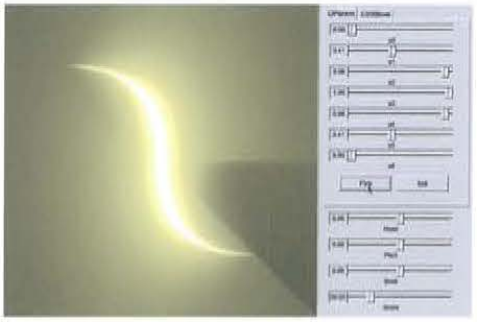
初期状態



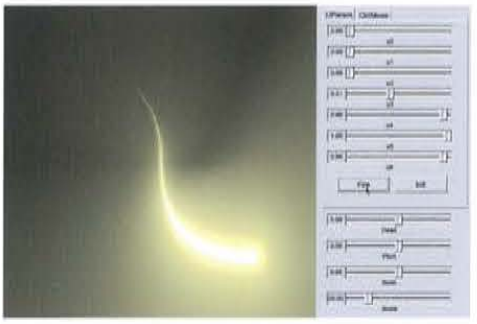
(a)



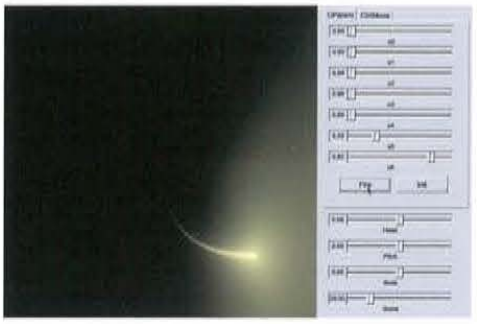
(b)



(c)



(d)



(e)

図 5.7　調整数列によるエネルギー波移動表現

図 5.8 は、本手法を用いた実践的な利用例を示したものである。

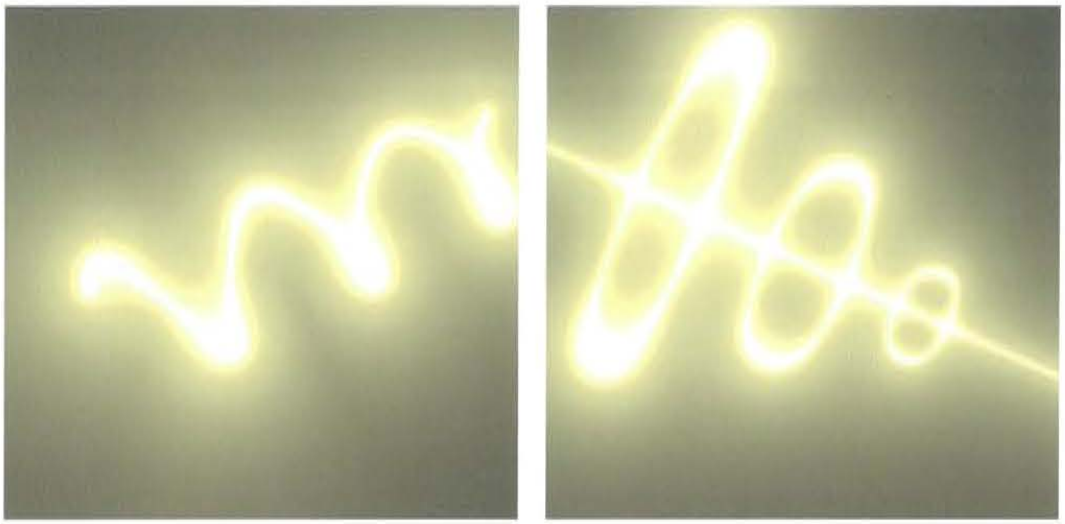


図 5.8　実践利用を想定した出力例

5.5　近似積分分割数の評価

本手法では、数値積分によって近似的に輝度値を求めているため、その際の分割数が低い場合は結果に影響が出ることがある。式 (3.28) における各エネルギー係数が低く、かつ分割数が低い場合、結果にムラが生じてしまうことがある。

図 5.9 - 5.12 は分割数を変えて描画を行った様子である。図 5.9 はトーラス型に対しエネルギー係数 $t_i = 8.0$、図 5.10 はトーラス型に対しエネルギー係数 $t_i = 4.0$、図 5.11 は自由曲線型に対しエネルギー係数 $b_i = 8.0$、図 5.12 は自由曲線型に対しエネルギー係数 $b_i = 4.0$ で描画を行っている。各図の下部が近似積分の際の分割数を示している。

64　　128　　256

図 5.9　トーラス型、$t_i = 8.0$

64　　128　　256

図 5.10　トーラス型、$t_i = 4.0$

64　　128　　256

図 5.11　自由曲線型、$b_i = 8.0$

64　　128　　256

図 5.12　自由曲線型、$b_i = 4.0$

図 5.9 と図 5.11 で示しているように、エネルギー係数が高い場合は分割数を変更しても目視ではほとんど違いはない。しかし、図 5.10 と図 5.12 のようにエネルギー係数が低い場合、分割数が低くなるに従ってムラが生じていることがわかる。分割数と描画速度はトレードオフの関係にあり、低いエネルギー係数による描画においては分割数の調整に注意が必要であるといえる。

5.6　改善点の考察

5.5 節で述べたように、近似積分分割数を低めに設定した場合には、エネルギー係数が低いエネルギー波描画の際にムラが生じてしまうという問題がある。このような周期性のあるムラに対応する手法として、確率的サンプリング (Stochastic Sampling)[48] が知られており、本手法に適用することで改善できることが考えられる。

本手法では「非常に高い分布値を持つが単一」な点よりも、「それほど高い分布値ではないが、視線経路上に満遍なく分布している」という画素の方が、より高い輝度値を持つという傾向にあり、視点の回転において違和感を感じることがある。その場合、式 (3.30) の $\zeta(s)$ に対し、単なる距離以外の減衰関数を用いることを検討する必要がある。

第 6 章

結論

6.1 研究のまとめと成果

本研究では、先行研究 [2] における分布関数の種類を拡張し、エネルギー波表現における新しい表現手法を提案したものである。

1 章では、アニメーション・漫画や特撮映像などの創作コンテンツにおいて、エネルギー波表現が用いられてきた歴史的経緯を説明すると共に、本研究における「エネルギー波」の定義を行った。近年では、アニメーションや特撮映像のみならず、ゲームコンテンツでも頻繁に用いられる映像効果となっており、表現技術の需要はますます高くなっていると言える。

2 章では、まず 2.1 節にてエネルギー波表現を実際にリアルタイムグラフィックス中で用いる際の現状の問題点について述べた。本来リアルタイムグラフィックスは、テクスチャマッピングを伴う面を描画することに最適化された仕組みを持っており、エネルギー波のような空間中に稠密な状態を表現するための技術はあまり向いているとは言えない。現状でエネルギー波を表現するための技術の多くは空間中に稠密な状態を表現することに向いていないことと、リアルタイムグラフィックス中でボリュームデータを扱う様々な手法はエネルギー波表現を行うには問題があることについて解説を行った。

2.2 節では、本研究の先行研究となる阿部らの手法 [2][3] の概要を紹介し、本研究で提案する理論の元となっている部分について詳しく述べた。

3 章は、本研究の理論基盤について述べた。阿部らにより既に点型分布関数、線型分布関数については提案がなされていたが、本論文ではそれに加えてトーラス型分布関数、2 次関数による分布関数を 3.2 節で提案した。特に、2 次関数についてはよく知られた自由曲線形式である 2 次 Bézier 曲線を用いることが可能であり、その適用方法について詳細に述べた。

また、3.3 節では 2 次曲線による分布関数に対し、曲線の部位毎にエネルギーの強弱を調整できる機能について述べた。曲線を等分割した際の各箇所に対し 0 から 1 までのパラメータ値を設

定することで、B-Spline 基底関数に基づいて曲線の任意部分についてのエネルギー強度が決定する。調整は容易であり、演算も高速であることから、本手法におけるエネルギー波の表現能力を大きく向上することに成功した。

4 章では、3 章の内容に基づき、本手法の実装について述べた。特に、本手法の特色の一つとして GPGPU を用いた高速化があり、並列に計算するための各種アルゴリズムの設計と、CPU と GPU でどのように処理を分担するかについて詳細に述べた。

5 章では、ここまで述べた成果を実際に実行し、描画結果や処理速度についての検証を行い、本研究の有用性を確認した。

本研究によって得られた成果をまとめた結果、以下のような結論が得られた。

1. 空間中に稠密なエネルギー波表現を、リアルタイムグラフィックス技術によって視点の位置や姿勢の制約を持たない状態で描画することを可能とした。
2. 従来手法に対し、トーラス型表現や自由曲線型の表現を可能とした。
3. 自由曲線型においては、部位毎の強弱を調整できる機能を可能とした。調整は曲線を等分割した際の各部位のパラメータ値を変更することで可能とし、リアルタイム性を損なわない実行速度を実現した。
4. GPGPU 技術を用いることにより、大量の演算処理を高速に実行することを実現し、本手法のリアルタイムグラフィックス中での適用を実用的なものとした。

以上の結果、本研究で目的としていた、エネルギー波表現のリアルタイムレンダリングに関する問題を解決し、これまでの手法では不可能であったエネルギー波表現を可能とした。

6.2 今後の課題と展開

現時点ではまだ実用するには多くの課題がある。まず、現在のエネルギー分布状態は数値パラメータによって決定するものであり、コンテンツ制作者にとって望ましい発光状態を作成するには直感的な入力ではない。これを容易とするようなツールの提供や、ゲームエンジンでの調整を可能とする GUI を提供するといった、作成を支援する方法の提供が必要である。また、先行研究において実現していた隠面処理については現時点でまだ実装および検証を行っていない。これについては早急に対応したい。また、発光体として捉えるのであればその他のオブジェクトを明るく照らす効果や、影の発生等も必要である。これらの問題点を克服することにより、本手法はゲーム等の創作コンテンツでの表現力を高めることができるであろう。

謝辞

本研究を遂行し学位論文をまとめるに当たり、大変多くの方々のご支援とご指導を賜りました。特に、本研究完遂にあたって欠かすことのできない方々への謝意をここに述べさせて頂きます。

博士後期課程において指導教員でありました岩手大学工学研究科の今野晃市教授に心より感謝申し上げます。20 年前に (株) リコーにてお会いしたときの縁と、学会での雑談を覚えていて頂き、博士後期課程の社会人入学をお薦め頂いたこと、遠方住まいである私に対しメールやネットワーク通話などを用いて丁寧にご指導頂いたことなど、大変ご恩を感じております。

副査を担当して頂いた岩手大学工学研究科の大塚尚寛教授と藤本忠博教授に感謝申し上げます。有益なご助言をありがとうございました。

当方の本務校である東京工科大学メディア学部の教員各位には、研究と業務の両面で大変お世話になりました。特に、近藤邦雄教授、柿本正憲教授、三上浩司教授に心より感謝申し上げます。近藤先生は私があまり学術活動を行っていなかった数年前より研究に対しいつも激励して頂き、今回の学位取得についてもいつも気に掛けて頂きました。今野先生に並ぶ学位取得での恩人であります。柿本先生は同じ研究分野の先達として活発にご議論させて頂くと共に、学位取得に際しては影日向なくご支援して頂きました。三上先生は十年に及ぶ共同研究者であり、本論文も三上先生とのコンテンツ研究に対する追求による賜と言えます。

今回の研究は、私が学生時代より開発を続けている「Fine Kernel Toolkit」というフレームワーク上でシステム構築を行いました。当システムの共同開発者である竹内亮太氏に感謝致します。彼はかつて私の指導学生でしたが、師である私より先に博士学位を取得しております。ようやく彼と肩を並べることができました。

本研究の基盤となる先行研究を行った阿部雅樹氏に感謝致します。彼の「ドラゴンボール」に対する情熱が、本研究の基礎を形作ったと言っても過言ではありません。また、GPGPU 関連の基礎部分も彼に構築してもらっており、彼の助力なくして本研究の完遂はありませんでした。

学生時代からの友人であり、現在はカーネギーメロン大学研究員である山川総司氏に感謝致します。研究に対する助言に加え、英文論文執筆の際に英語が苦手な当方の稚拙な英文を、素晴らしい表現に修正して頂きました。

その他、研究を進めるにあたり、ご支援、ご協力を頂きながら、ここにお名前を記すことかが出来なかった多くの方々に心より感謝申しあげます。

最後に、四十路の坂を越えてからの博士後期課程入学を快く承諾し、どのような状況においても応援してくれた妻の理代子と、いつも心の支えになってくれた娘の真凪に、心から感謝致します。

参考文献

[1] D.Nowrouzezahrai, J.Johnson, A.Selle, D.Lacewell, M.Kaschalk, and W.Jarosz. A programmable system for artistic volumetric lighting. *ACM Transactions on Graphics*, Vol. 30, No. 4, pp. 29:1–29:8, 2011.

[2] 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. 芸術科学会論文誌, Vol. 9, No. 3, pp. 93–101, 2010.

[3] 阿部雅樹. エネルギー波表現のリアルタイムレンダリング. 修士論文, 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, 2010.

[4] 野村宏平. ゴジラ大辞典. 笠倉出版社, 2004.

[5] 円谷プロダクション. ウルトラマン大図鑑. ポプラ社, 2011.

[6] ライダー・ウィンダム, ダニエル・ウォーレス, アダム・ブレイ, トリシア・バー. アルティメット・スター・ウォーズ. 学研マーケティング, 2015.

[7] スタジオハード MX. 宇宙戦艦ヤマト画報 – ロマン宇宙戦記二十五年の歩み. 竹書房, 2001.

[8] サンライズ. 機動戦士ガンダム MS 大図鑑 PART 1 一年戦争編. バンダイ, 1989.

[9] 鳥山明. DRAGON BALL 大全集 – 鳥山明ワールド (7). 集英社, 1996.

[10] MATCHLOCK Corporation. BISHAMON. `http://www.matchlock.co.jp/english/`. 参照: 2016.4.15.

[11] Effekseer 開発チーム. Effekseer. `http://effekseer.github.io/jp/index.html`. 参照: 2016.4.15.

[12] W.T.Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Transactions of Graphics*, Vol. 2, pp. 91–108, 1983.

[13] E.Haines and T.A.Moller. *Real-Time Rendering, Third Edition.* A K Peters/CRC Press, 2008.

[14] 仁藤将輝, 渡辺大地, 柿本正憲, 三上浩司. リアルタイム 3DCG における衝突を考慮したエネルギー波表現. 芸術科学会論文誌, Vol. 13, No. 3, pp. 144–153, 2014.

[15] 仁藤将輝. リアルタイム 3DCG における衝突を考慮したエネルギー波表現に関する研究. 修士論文, 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, 2014.

[16] R.Drebin, L.Carpenter, and P.Hanrahan. Volume rendering. *Computer Graphics*, Vol. 22, No. 4, pp. 65–74, 1988.

[17] M.Groher, F.Bender, R.T.Hoffmann, and N. Navab. Segmentation-driven 2D-3D registration for abdominal catheter interventions. *Proceedings of the 10th international confernce on MICCAI*, pp. 527–535, 2007.

[18] 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*, Vol. 27, No. 3, pp. 36:1–12, 2008.

[19] R.Takeuchi, T.Watanabe, and S.Yamakawa. Sketch-based solid prototype modeling system with dual data structure of point-set surfaces and voxels. *International Journal of CAD/CAM*, Vol. 11, No. 1, pp. 1–9, 2011.

[20] R.Takeuchi, T.Watanabe, M.Kakimoto, K.Mikami, and K.Kondo. Stroke history management for digital sculpting. 芸術科学会論文誌, Vol. 11, No. 4, pp. 108–117, 2012.

[21] 竹内亮太. ストローク履歴を活用した 3 次元形状モデリング手法の研究. 博士論文, 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, 2013.

[22] Inc Pixologic. Home of ZBrush. `http://pixologic.com`. 参照: 2016.5.8.

[23] H.Tuy and L.Tuy. Direct 2d display of 3d objects. *IEEE CGA*, Vol. 4, No. 10, pp. 29–34, 1984.

[24] J.Krufer and R.Westermann. Acceleration techniques for GPU-based volume rendering. *Proceedings of the 14th IEEE Visualization*, pp. 287–292, 2003.

[25] J.Stuart, C.Chen, K.Ma, and J.Owens. Multi-GPU volume rendering using mapreduce. *Proceedings of the 19th ACM HDPC'10*, pp. 841–848, 2010.

[26] 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グラフィクスと CAD, Vol. 2007, No. 13, pp. 67–72, 2007.

[27] S.Venkataraman. 4D volume rendering. In *GPU Technology Conference 2009*, 2009.

[28] J.Blinn. A generalization of algebraic surface drawing. *ACM Transactions of Computer Graphics*, Vol. 1, No. 3, pp. 235–256, 1982.

[29] 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*, Vol. 568, pp. 718–725, 1985.

[30] Y.Kanamori, Z.Szego, and T.Nishita. GPU-based fast ray casting for a large number of metaballs. *In Proc. Eurographics*, Vol. 27, No. 2, pp. 351–360, 2008.

[31] T.Kanai, Y.Ohtake, H.Kawai, and L.Kase. GPU-based rendering of sparse low-degree implicit surface. *In Proc. of GRAPHITE*, pp. 165–171, 2006.

[32] A.Pasko, V.Adzhiev, A.Sourin, and V. Savchenko. Function representation in geometric modelling: concept, implementation and applications. *The Visual Computer*, Vol. 11, No. 8, pp. 429–446, 1995.

[33] A.Pasko, V.Adzhiev, B.Schmitt, and C. Schlick. Constructive hypervolume modeling. *Graphical Models*, Vol. 63, No. 6, pp. 413–442, 2001.

[34] 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.

[35] B.Schmitt, A.Pasko, V.Adzhiev, and C. Schlick. Constructive texturing based on hypervolume modeling. *Journal of Visualization and Computer Animation*, Vol. 12, No. 5, pp. 297–310, 2002.

[36] J.Børlum, B.B.Christensen, T.K.Kjeldsen, P.T.Mikkelsen, K.Ø.Noe, J.Rimestad, and J.Mosegaard. SSLPV: subsurface light propagation volumes. *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, pp. 7–14, 2011.

[37] M.Billeter, E.Sintorn, and U.Assarsson. Real time volumetric shadows using polygonal light volumes. *Proceedings of the Conference on High Performance Graphics*, pp. 39–45, 2010.

[38] M.Billeter, E.Sintorn, and U.Assarsson. Real-time multiple scattering using light propagation volumes. *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 119–126, 2012.

[39] E.Hecht. ヘクト 光学 I – 基礎と幾何光学 – 第 4 版. 丸善, 2002.

[40] A.Treuille, A.McNamara, Z.Popović, and J.Stam. Keyframe control of smoke simulations. *ACM Transaction of Graphics*, Vol. 22, No. 3, pp. 716–723, 2003.

[41] R.Fattal and D.Lischinski. Target-driven smoke animation. *ACM Transaction of Graphics*, Vol. 23, No. 3, pp. 441–448, 2004.

[42] Y.Dobashi, K.Kusumoto, T.Nishita, and T.Yamamoto. Feedback control of cumuliform cloud formation based on computational fluid dynamics. *ACM Transaction of Graphics*, Vol. 27, No. 3, pp. 94:1–94:8, 2008.

[43] 佐藤周平, 土橋宜典, 山本強, 安生健一. 最適な初期強度分布の推定および予測制御による爆発シミュレーションの制御. 映像情報メディア学会誌, Vol. 65, No. 11, pp. 1446–1451, 2011.

[44] D.C. ベンソン. 数学へのいざない (上). 朝倉書店, 2006.

[45] W.H.Press, B.P.Flannery, S.A.Teukolsky, and W.T.Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[46] G.E.Farin. NURBS 射影幾何学から実務まで 第 2 版. 共立出版, 2001.

[47] M.Pharr, R.Fernando, and T.Sweeney. *GPU Gems 2.* Addison-Wesley Professional, 2005.

[48] Robert L.Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, Vol. 5, No. 1, pp. 51–72, 1986.

発表業績

学術論文

1. 渡辺大地, 千代倉弘明, 任意三角形メッシュからの特徴稜線抽出, 電子情報通信学会論文誌 D-IIJ83-D-II(5), pp.1344-1352, 2000.
2. 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, 芸術科学会論文誌第 9 巻 (第 3 号), 2010.
3. Koji Mikami, Taichi Watababe, others, Construction Trial of a Practical Education Curriculum for Game Development Through Industry-University Collaboration, Computer & Graphic Journal Vol.34, pp.791-799, 2010.
4. Ryota Takeuchi, Taichi Watanabe, Soji Yamakawa, Sketch-based Solid Prototype Modeling System with Dual Data Structure of Point-set Surfaces and Voxels, International Journal of CAD/CAM Vol.11, No.1, 2011.
5. 近藤邦雄, 伊藤彰教, 三上浩司, 渡辺大地, Example Based Programming に基づく CG 制作の入門教育, 図学研究 (図学会論文誌) Vol.45 No.3, pp3-10, 2011.
6. 松尾隆志, 三上浩司, 渡辺大地, 近藤邦雄, リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案, 芸術科学会論文誌 Vol.10 No.4, pp.251-262, 2011.
7. Ryota Takeuchi, Taichi Watanabe, Koji Mikami, Kunio Kondo, Proposal of digital sculpting with history management of strokes, 芸術科学会論文誌 Vol.11 No.4, pp.108-117, 2012.
8. 小島啓史, 竹内亮太, 渡辺大地, 三上浩司, 特徴的な動き方を考慮したオーロラのビジュアルシミュレーション, 芸術科学会論文誌 Vol.12 No.1, pp.24-35, 2013.
9. Pulung Nurtantio Andono, I Ketut Eddy Purnama, Mochamad Hariadi, Taichi Watanabe, Kunio Kondo, 3D Surfaces Reconstruction of Seafloor Images Using Multiview

Camera Based on Image Registration, Transaction of ICACT, 201339, pp.227-235, 2013.

10. 小島啓史, 竹内亮太, 石川知一, 三上浩司, 渡辺大地, 柿本正憲, 近藤邦雄, 分断と再統合現象を考慮したオーロラのビジュアルシミュレーション, 情報処理学会論文誌 Vol.55 No.8, pp.1886-1898, 2014.
11. 仁藤将輝, 渡辺大地, 柿本正憲, 三上浩司, リアルタイム 3DCG における衝突を考慮したエネルギー波表限, 芸術科学会論文誌 Vol.13 No.3, pp.144-153, 2014.
12. Tomokazu ISHIKAWA, Yonghao YUE, Taichi WATANABE, Kei IWASAKI, Yoshinori DOBASHI, Masanori KAKIMOTO, Kunio KONDO, Tomoyuki NISHITA, Visual Simulation of Glazed Frost Using Hybrid Heat Calculation, IIEEJ Transactions on Image Electronics and Visual Computing, Vol.3 No.2, pp.136-142, 2015.
13. Taichi Watanabe, Masaki Abe, Kouichi Konno, Real-Time Rendering Technique for Visual Expression of Arbitrary-Shaped Energy Wave, 芸術科学会論文誌 Vol.15, No.2, pp.99-111, 2016.

研究速報

1. 成田晃, 柿本正憲, 渡辺大地, 竹内亮太, 三上浩司, 破片飛散現象表現におけるカメラ回避制御, 映像情報メディア学会論文誌 No.67 Vol.4, pp.148-151, 2013.

国際会議口頭発表 (査読付)

1. Taichi Watanabe, Hiroaki Chiyokura, Toolkit for Making 3D Cyber World, IEEE Workshop on Networkeed Realities, 1995.
2. Koji Mikami, Taichi Watanabe, others, Construction trial of a practical education curriculum for game development by industry/university collaboration, SIGGRAPH Asia 2009, 2009.
3. Ryota Takeuchi, Taichi Watanabe, Koji Mikami, Kunio Kondo, Proposal of digital sculpting with history management of strokes, NICOGRAPH International 2012, 2012.
4. Taichi Watanabe, Takafumi Kojima, Ryota Takeuchi, Automatic Generation of Aurora Animation Using Complex Functions, IWAIT2013, 2013.
5. Taichi Watanabe, Masaki Abe, Kouichi Konno, A Shape-Controlling Method of Energy-Wave with Continuous Scalar Function for Real-Time Rendering, NICOGRAPH International 2015, 2015.

国際会議ポスター発表 (査読付)

1. Ryota Takeuchi, Taichi Watanabe, Illstration based sculpture modeling system by point set surface, VRCAI 2009, 2009.
2. Takashi Matsuo, Koji Mikami, Kunio Kondo, Taichi Watanabe, Shape Oriented Line Drawing in Real-Time 3DCG, SIGGRAPH ASIA 2011, 2011.
3. Takafumi Kojima, Ryota Takeuchi, Soji Yamakawa, Taichi Watanabe, Koji Mikami, Visual Simulation of Aurora Movement, SIGGRAPH ASIA 2011, 2011.

4. Wataru Yamada, Taichi Watanabe, Masanori Kakimoto, Koji Mikami, Ryota Takeuchi, Reproduction of the Behavior of the Wet Cloths Taking the Atmospheric Pressure Into Account, SIGGRAPH 2013, 2013.
5. Tomokazu Ishikawa, Yonghao Yue, Taichi Watanabe, Kei Iwasaki, Yoshinori Dobashi, Masanori Kakimoto, Tomoyuki Nishita, Visual Simulation of Glazed Frost, SIGGRAPH 2013, 2013.
6. Takafumi Kojima, Yuka Sato,Taichi Watanabe, Koji Mikami, Kunio Kondo, V-Shaped Highlight in 2D Animated Character's Hair for Real-Time 3DCG, NICOGRAPH International 2014, 2014.

著作

1. Kunio Kondo, Taichi Watanabe, Interactive Sketch Interpreter for Geometric Modeling, The Visual Language of Technique 2, pp.29-48, Springer, 2015.
2. 三上浩司, 渡辺大地, CG とゲームの技術, オーム社, 2016.

学術雑誌解説・総説

1. 近藤邦雄, 三上浩司, 柿本正憲, 渡辺大地, 菊池司, 石川知一, グループ紹介:東京工科大学メディア学部 デジタルコンテンツ制作技術研究グループ, 画像電子学会学会誌, Vol.43 No.3, pp.420-425, 2014.
2. 渡辺大地, 独自ツールキットを用いたゲーム制作教育への取り組み, 情報処理学会学会誌, Vol.56 No.12, pp.1206-1209, 2015.

解説記事

1. 渡辺大地, 独自ツールキットを用いたゲーム制作教育への取り組み, 日経テクノロジーオンライン, 2015.

受賞歴

1. 第 11 回 NICOGRAPH 論文コンテスト優秀賞, 論文題目「柔軟な形状制御機能を持った Cyber World ToolKit」, 1993.
2. 情報処理学会 グラフィクスと CAD 研究会 優秀発表賞, 論文題目「独自ツールキットによるスケーラブルな CG とゲーム開発の教育的実践」, 2012.
3. 関東工学教育協会 2012 年度 業績賞.
4. 情報処理学会 2013 年度 優秀教材賞.