2011年度
卒業論文

大規模シミュレーションデータの新しい可視化手法：固定視点動画の移動視点再生

神戸大学工学部情報知能工学科

山田知輝

指導教員　陰山聡

2012年2月22日

大規模シミュレーションデータの新しい可視化手法：固定視点動画の移動視点再生

山田知輝

要旨

スーパーコンピュータから出力されるデータは膨大であり、そのサイズは今後もさらに増大することが確実である。そのためポストプロセスとしての可視化手法には限界が来ている。そこで本研究では、ポストプロセスに代わる大規模シミュレーションデータの新しい可視化手法を提案する。この手法は実時間可視化法に基づくものであり、実時間で可視化を行うことによりデータ転送及び可視化処理にかかる時間をなくすことが可能となる。ただし、古典的な実時間可視化法とは異なり、ここで提案する手法では、非常に多くの視点で可視化を行う事により対話的な再生を可能とする。この手法を検証するために、Yin-Yang 格子上の 130 点に配置したカメラから可視化動画を作成した。その動画ファイル群から視線方向を自由に回転させつつ再生する機能を持つ対話的動画再生プログラムを開発した。実際のシミュレーションへの応用として、鳥取県西部地震の 3 次元地震波伝播シミュレーションにおける P 波および S 波の可視化を行った。検証の結果、本提案手法の有効性が確認された。今後は $O(10^3)$ 個の視点数に対応し、視点変換の自由度をさらに高めることができるよう動画プレーヤを改良し、さらに他のシミュレーションの実時間可視化に応用していく。

目 次

1 序論 **1**
1.1 研究の背景 1
1.2 提案する手法の概要 2

2 映像処理と可視化手法の基礎 **4**
2.1 レイキャスティング 4
2.2 Yin-Yang 格子 4
2.3 動画の圧縮 5
2.4 OpenCV 6

3 固定視点動画の移動視点再生プログラムの概要 **7**
3.1 目的 7
3.2 方法 7

4 地震波伝播シミュレーションへの応用 **10**
4.1 地震波伝播シミュレーションとは 10
4.2 地震波伝播シミュレーションの可視化 12
4.3 P 波と S 波の可視化手法 14
4.4 可視化結果 17

5 まとめ **23**

6 謝辞 **24**

参考文献 **24**

1 序論

1.1 研究の背景

スーパーコンピュータの進歩は近年、めざましいものがある。2010 年 11 月の TOP500[1] では中国の天河 1 号 A が 1 位となりその時の LINPACK 性能は 4.7PFlops であった。その 1 年後、理化学研究所の京が 1 位となり 11PFlops を記録した。スーパーコンピュータの性能の指数関数的な進歩は今後もしばらくは続くと予想される。

ハードウェアの進歩に伴い、計算機シミュレーションの大規模化と高速化も進んでいる。シミュレーション規模の大規模化はシミュレーションの出力データ量の増大をもたらす。そして大規模なデータはそれを解析するために不可欠な可視化の難しさを増大させる。大規模な計算機シミュレーションの結果を効果的に可視化する事ができるかどうかが、研究の成否を決める重要な要因となりつつある。

現在のシミュレーション研究においては可視化はポストプロセスとして行われるのが普通である。しかし、シミュレーション結果（データ量）の増大によって、ポストプロセスとしての可視化手法は限界を迎えつつある。

その理由の一つはシミュレーション結果のデータ転送に時間がかかるということである。例えば、3TB のサイズを持つシミュレーション結果を可視化しようとする場合、仮に 1Gb/s という高い実効転送速度が実現されたとしても、全てを転送するのに 24000 秒、即ち 6.6 時間程かかる。しかもこのような高い実効転送速度が実現される事は実際には期待する事はできない。仮に実効転送速度がこの理想的状況の 1/10 だとすれば、転送時間は数日間となる。

ポストプロセスとしての可視化が難しくなってきているもう一つの理由として、可視化処理自体もまた計算量が大きく、時間がかかるようになっているという事が挙げられる。少し以前の大規模シミュレーション研究では、可視化と言えば専用のグラフィックワークステーションがあれば十分であった。しかし今日、汎用の可視化ソフトとグラフィックワークステーションだけで可視化ができた時代は終わりつつある。シミュレーションの高度化と複雑化が進むにつれて可視化に必要な演算処理量も急激に増加し、また可視化の方法もそれぞれの問題に応じた専用の手法とスタイルが必要とされている。このような状況の下、大規模なシミュレーションデータの可視化は、その事自体が大規模な演算を必要とするようになりつつある。その計算規模はもはや PC やグラフィックワークステーションの能力を超える程の大きさになっており、スーパーコンピュータクラスの計算機が必要とされる事も珍しくない。明らかにポストプロセスとしての可視化は限界に来ていると言える。

上述のような背景の下、ポストプロセスとしての可視化手法の問題点を解決す

る新しい可視化手法の提案するのが本研究の目的である。

1.2 提案する手法の概要

本研究では、スーパーコンピュータを用いた大規模な計算機シミュレーションデータの新しい可視化手法を提案する。この可視化手法の基本的な土台は実時間可視化手法である。

実時間可視化手法とはシミュレーション計算と同時に可視化を行う手法である。通常、実時間可視化手法といえば可視化画像を作成するための視点が1つ、あるいは少数である事を想定している。従って別の視点から見た可視化画像が必要とされた場合には、もう一度実時間可視化の計算ジョブを計算機に投入する必要がある。この不便さがポストプロセス方式の可視化と比較すると全くと言っていい程実時間可視化が広まっていない原因である。実時間可視化手法がもつこの問題点を解決する方法を本研究では提案する。それは後で必要となる可能性のある全視点からの可視化画像を全て生成しておくというある意味で単純な方法である。例えば、あるシミュレーション領域を水平面内で360度回転しながら見る事がデータの解析上予想されるのであれば、可視化画像を生成する視点あるいはカメラをあらかじめ360台“設置”し、その全ての視点から可視化しておけばよい（Fig.1)。解析対象のシミュレーションが何かの現象の時間発展を追うものであれば、これら360台のカメラから実時間可視化によって作られる可視化結果としてのデータは360個の動画ファイルである。

これら360個の動画はそれぞれ1度ずつカメラ位置が異なるだけで全く同じシミュレーションを可視化している。360個の動画ファイルには（1度刻みではあるが）実質上どの方向から見た映像であっても自由に取り出せるだけの情報を保持している。従って、ある静止時刻における現象をぐるっと回転しながら見る事もできるはずだし、さらに時間発展している現象の可視化（即ち動画）を再生させながら、例えばマウス操作等を通じて自由自在に見る方向を変更する事が可能である。

可視化用カメラの位置は何も水平面の2次元的な配置に限る必要はない。シミュレーション領域を取り囲むように全方位、即ち立体角4πの球面上にカメラを分布させてやれば、3次元シミュレーションの実時間可視化の結果を好きな方向から見る事ができるだけでなく、解析対象を（通常のポストプロセス可視化ではマウスドラッグ等で実現している）自由な回転操作等も実現できる（Fig.2)。上に述べた方法をまとめると、「固定視点の可視化動画を実時間可視化法で大量に生成し、そうして生成された動画ファイル群から映像を適切に抜き出す事で対話的な可視化を実現する」方法と言える。本研究の目的はこのアイデアの有効性を実証する事である。

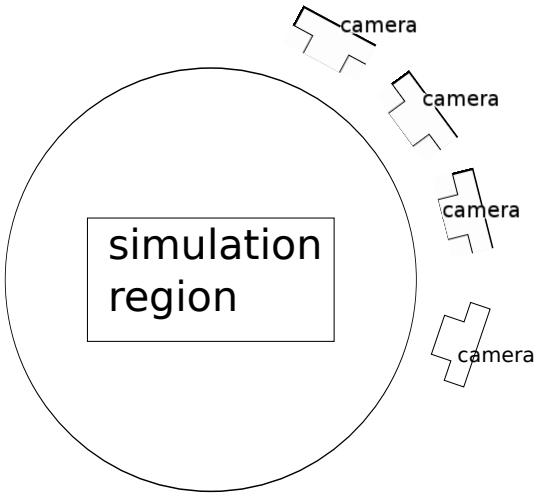


Fig. 1: Concept of real time visualization with multiple cameras

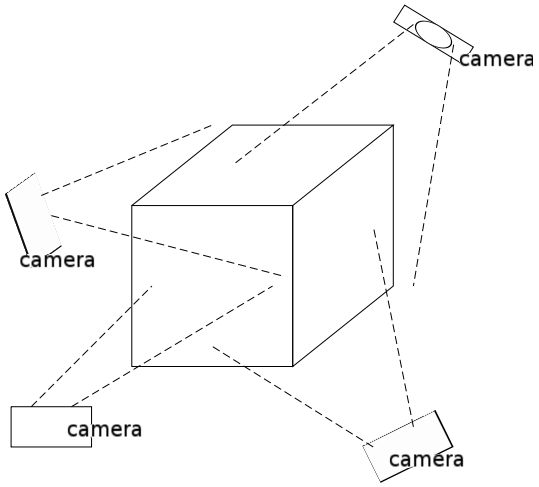


Fig. 2: Multiple cameras surrounding 3-D simulation region

2　映像処理と可視化手法の基礎

序章で述べた固定動画の対話的再生を実証する目的で実際にプログラムを実装し、可視化の実証実験を行った。この章ではその実証実験で用いた可視化やデータ処理に関わる基本手法やソフトウェアツールについて説明する。

2.1　レイキャスティング

ボリュームレンダリングとはスカラー場の 3 次元分布を可視化する基本的可視化手法の一つである。もう一つの基本的なスカラー場可視化手法である等値面表示手法では、可視化のために（数学的に構成した）仮想的な 3 次元物体を多角形（ポリゴン）を組み合わせて作るのに対し、ボリュームレンダリングではそのようなポリゴンを経由せずに画像を生成する。煙や霧のようにぼんやりとした分布をそのまま可視化画像で表現できる事がボリュームレンダリング法の特徴であり、利点でもある。

レイキャスティング法はボリュームレンダリングを実現する手法の一つである。まず、ある視点におけるスクリーンを設定する。スクリーン上の画素から光線、即ちレイを延ばしていく。レイの延長線上にボクセルがあるとする。そのボクセルが持つスカラー値を足し上げる。これを最大値を超えるまで、もしくはボリュームデータの境界に至るまで行う。この計算結果をそのスクリーンの画素の画素値に対応させる。これを全ての画素で行う事で画像を生成する。これがレイキャスティングである。

2.2　Yin-Yang 格子

球体面での数値計算では基本格子系として緯度経度格子がよく用いられる。緯度経度格子とは球座標の緯度と経度方向にそれぞれ等間隔に切った格子系である。格子の位置は余緯度 $\theta(0 \leq \theta \leq \pi/2)$ と経度 $\phi(-\pi < \phi \leq \pi)$ で表す事ができる。

この緯度経度格子は 極近傍では格子間隔が過度に密になってしまうという問題がある。この欠点を解消するため、Kageyama and Sato は Yin-Yang 格子を提案した [2] 。Yin-Yang 格子では、球座標における余緯度 θ において $\pi/4 \leq \theta \leq 3\pi/4$、経度 ϕ において $-3\pi/4 \leq \phi \leq 3\pi/4$ の格子部分を取り出す。これを Yin 格子とする。そして Yin 格子を余緯度方向に $\pi/2$、経度方向に π 回転させる。これを Yang 格子とする。これら 2 つの格子を接合した格子が Yin-Yang 格子である。Yin-Yang 格子はほぼ均一な格子間隔を持つ。(Fig.3 参照)

Yin-Yang 格子でも格子の位置を θ と ϕ で表す事ができる。ただし、Yin 座標系上のそれらと Yang 座標系上のとは定義が異なる。ここで Yin 座標系上における

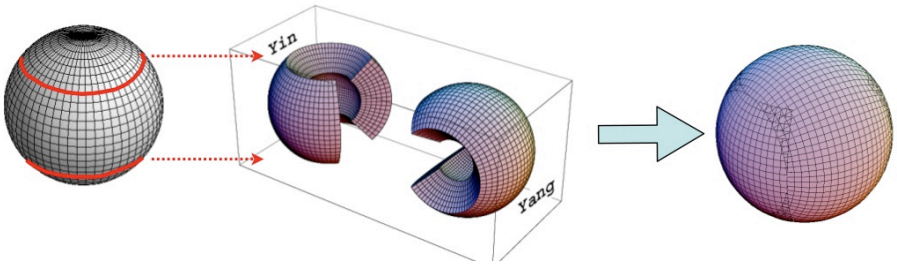


Fig. 3: Yin-Yang grid

余緯度と経度の値を Yang 座標系上におけるそれに変換する手順を示す。Yin 座標系上の θ と ϕ をそれぞれ θ^n と ϕ^n とする。同様に Yang 座標系上の θ と ϕ をそれぞれ θ^e と ϕ^e とする。まず、Yin 座標系上において、球体半径を r とした場合、カーテシアン座標 (x^n, y^n, z^n) への変換を行う。

$$(x^n, y^n, z^n) = (r\sin\theta^n\cos\phi^n, r\sin\theta^n\sin\phi^n, r\cos\theta^n) \tag{1}$$

となる。次に Yin 座標系上におけるカーテシアン座標を Yang 座標系上におけるカーテシアン座標へと変換する。

$$(x^e, y^e, z^e) = (-x^n, z^n, y^n) \tag{2}$$

となる。そして Yang 座標系上におけるカーテシアン座標を球座標に変換すればよい。

$$\theta^e = \cos^{-1}\left(\frac{z^e}{\sqrt{(x^e)^2+(y^e)^2+(z^e)^2}}\right), \quad \phi^e = \begin{cases} \cos^{-1}\left(\frac{x^e}{\sqrt{(x^e)^2+(y^e)^2}}\right) & (y^e \geq 0) \\ -\cos^{-1}\left(\frac{x^e}{\sqrt{(x^e)^2+(y^e)^2}}\right) & (y^e < 0) \end{cases} \tag{3}$$

となる。このようにして Yin 座標系上の余緯度と経度を Yang 座標系上のそれらに変換できる。Yin 格子と Yang 格子は合同の格子である。そのため、Yang 座標系から Yin 座標系への変換は Yin 座標系から Yang 座標系への変換と同等の手続きをとればよい。

2.3 動画の圧縮

本研究では動画ファイルフォーマットとして MPEG-4 を用いる。MPEG-4 は空間方向に離散コサイン変換、時間方向には動き補償を組み込んだフレーム間予測により高い圧縮率をもつのが特徴である。

本研究では動画の作成と変換にフリーソフトウェアの FFmpeg を利用した [4]。FFmpeg はマルチメディアフレームワークである。機能として、様々な画像や動画形式に対応したデコード、エンコード、マルチプレクサ、デマルチプレクサ、ストリーム、フィルタリング機能を持つ。コーデックライブラリには libavcodec が使われている。本研究では、同時可視化のポストプロセスとして CUI として配布されているものを使用した。特に、エンコード機能を利用して、PPM 形式の連番画像を MPEG-4 形式の動画へ変換する事に用いた。

2.4 OpenCV

OpenCV はオープンソースのコンピュータビジョンライブラリである。コンピュータビジョンとは静止カメラやビデオカメラからの静止画像もしくは動画像等を対象として計算機に画像認識させることである。2 次元画像の変換だけでなく、物体認識、対象検出、カメラキャリブレーションの機能等も持つ。

実際的な画像認識を実現するために OpenCV では高速性（リアルタイム処理）が重視されている。本研究ではその性質を利用して、対話型の動画プレーヤを開発した。

3　固定視点動画の移動視点再生プログラムの概要

3.1　目的

固定視点で撮影した多数の動画ファイル群から必要な情報を抜き出し、任意視点からの対話的な動画再生を実現するために OpenCV に基づいた独自の動画再生ソフトを開発した。

本ソフトは、Yin-Yang 格子上に配置された $O(10^2)$ 個の固定視点動画ファイルを読み込み、以下の 3 つの機能

- 再生、静止を自由に切り替える
- 再生視点の視線方向を自由かつ滑らかに 3 次元的に回転させる
- ズームイン、ズームアウトを自由にできる

を実装した。

3.2　方法

シミュレーション領域を 3 次元的に取り囲むように可視化用のカメラ位置（視点）を配置するために、本研究では Yin-Yang 格子上に視点を配置した。Yin-Yang 格子では球面上どの位置においても格子間隔がほぼ同じであるためである。

極座標系の格子点上に、等緯度等経度間隔に視点の位置をとった場合、極近くで視点が集中してしまうという問題がある。視点の数を極部分で間引きしたとしても、赤道上の視点から極方向へ視点を移動させたときに対応した等経度位置に視点がないという問題がある。球座標に基づくカメラ配置のこのような問題をさけるため本研究では Yin-Yang 格子を用いた。

本研究で開発した動画プレーヤは C++ によってクラス化されており、動画再生の行程はメンバ関数で整理されている。コンストラクタによる初期化処理、内部バッファにおける画像データの更新、画像データのウィンドウへの表示、キーボード操作による視点位置などのメンバ変数の値の更新、そしてデストラクタによる全メモリの解放などがある。このクラスをメイン関数上で呼び出す。終了処理のフラグが立つまで、画像データの更新、ウィンドウへの表示、メンバ変数の値の更新といったメンバ関数を繰り返し呼び出す事で動画の再生を行う。

初期化処理としては、動画ファイルの読み込みや視点情報の初期設定などを行う。Yin-Yang 格子上の $O(10^2)$ 点に配置した動画の読み込みには cvCreateFileCapture 関数を用いた。この関数は指定したファイル名の動画ファイルを読み込み、CvCapture 構造体へのポインタを返す。CvCapture 構造体は読み込んだ動画ファ

イルに関する情報全てを保持している。130 視点の場合、動画プレーヤのクラスはこの CvCaputure 構造体を $2 \times 13 \times 5$ の 3 次元配列としてメンバに持っている。2 は Yin と Yang の座標系、13 は経度方向へ分布された視点の数、5 は余緯度方向へ分布された視点の数を示す。ファイル名には 0 から 129 までの番号がついている。その値から視点位置に対応した添字を計算する。これらの処理はコンストラクタで行った。行列の CvMat 構造体のメモリ確保や各変数の初期化もこの段階で行っている。その後、cvNamedWindow 関数によって、画像を保持し表示する事ができるウィンドウを画像と同じサイズでスクリーン上にひらく。

コンストラクタで最初の動画像表示に必要な設定を行う。その情報を元に内部バッファへ画像の情報を格納する。動画プレーヤのクラスはメンバ変数として動画の再生フラグを持っており、再生フラグが ON である場合、現在の視点位置に該当する CvCapture 構造体から次に表示するフレームの画像を取り出す。これを cvQueryFrame 関数によって行う。この関数は動画データから画像をキャプチャし、そのデータを内部バッファにコピーし、そのバッファを指す IplImage*ポインタを返す。またその動画におけるフレームの位置を自動的に進めることができる。再生フラグが OFF であれば、内部バッファの画像の更新を行わない。これにより、動画の再生と中断を切り替えることができる。

視点の位置が切り替わった場合、フレームの位置を合わせる必要がある。全体におけるフレームの位置の値をメンバとして持っておく事で、動画の同期を行った。視点位置の変更のフラグがオンである場合、対応した動画のフレームの位置を最初の位置に設定する。そしてメンバ変数の値分 cvQueryFrame 関数を呼び出している。これによって滑らかな視点の切り替えを可能としている。

その次に、IplImage*ポインタの画像をウィンドウ上に表示させる。この際、いったん画像を拡大、縮小、回転の処理を行って、その画像データを内部バッファの別の領域へ格納してから表示を行う。この処理は cv2DRotationMatrix 関数で画像変換行列を計算する事で実現した。この拡大縮小変換によって、ズームインとズームアウトの機能を実現した。画像を回転させるのは、その視点における上方向を変えるためである。これにより、Yin 座標系上の視点と Yang 座標系上の視点の切り替えをどの視点位置からでも行えるようになった。

キー入力によって、対応したメンバ変数の値を更新することで、次の内部バッファの更新や画像のウィンドウ表示にその処理を反映させる。具体的な手順は以下の通りである。まず cvWaitKey 関数でキー入力を受け取る。z キーまたは x キーならばズームインまたはズームアウトを行う。動画プレーヤのクラスはメンバ変数として画像の倍率の値を保持している。ズームする場合はその値を調整し、次の画像のウィンドウ表示の際にそれを反映させる。スペースキーならば動画の再生フラグの ON と OFF を切り替える。カーソルキーならば視点の位置情報を更新する。

視点位置情報の更新は Fig.4 のような Yin-Yang 格子の展開図で考える。ここで Yin-Mesh において横方向に並ぶ数字は経度方向のカメラの場所、縦方向に並ぶ数字は緯度方向のカメラの場所である。CvCapture 構造体の添字をここで決定する。キー入力を受け取り、このメッシュの線に沿って視点位置を切り替えていくようにする。

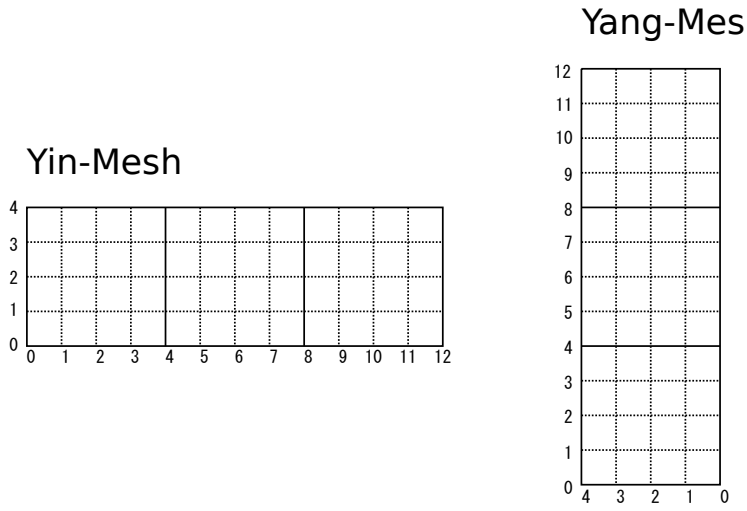


Fig. 4: 130 viewpoints on Yin-Yang grid

視点の位置の移動方向はその視点の上方向ベクトルとキー入力によって決定する。上方向ベクトルは上下左右の 4 方向をとる。上キーの入力なら上方向ベクトルの方向に視点を現在の位置から 1 マス移動させ、下方向ならその逆、左右も対応した方向に視点を 1 マス移動させる。

現在の視点位置が一方の格子メッシュの境界線上にあり、移動したあとの視点位置がその領域の外へ出るとき、現在の視点位置を表す整数値を別の系の格子メッシュの視点位置を表す整数値に変える。この計算を複数の段階に分けて行う。まず、その視点が属する格子の系列の極座標系へ変換し、カーテシアン座標へ変換する。その値を Yin-Yang の座標変換によって別の座標系におけるカーテシアン座標をもとめる。そして極座標変換を行い、メッシュ上の位置を表す整数値を求める。視点の上方向の値も別の系へ移動するときに値を変える。

キー入力で ESC キーが押された時、終了処理のフラグが立つ。メイン関数がそれによって更新処理の繰り返しを中断し、デストラクタを呼び出す。デストラクタでは全ての視点における cvCapture*構造体や行列、ウィンドウのメモリを解放する。

4　地震波伝播シミュレーションへの応用

前章で述べた動画再生ソフトを適用する実時間可視化の対象シミュレーションとして、地震波伝播シミュレーションを選んだ。本章ではこのシミュレーションについて述べる。

4.1　地震波伝播シミュレーションとは

地震波伝搬シミュレーションでは、地震波の伝播方程式の時間発展を計算する。3 次元直交座標における地震波の方程式は以下の通りである。

$$\rho \ddot{u}_p = \frac{\partial \sigma_{xp}}{\partial x} + \frac{\partial \sigma_{yp}}{\partial y} + \frac{\partial \sigma_{zp}}{\partial z} + f_p \quad (p = x, y, z) \tag{4}$$

ここで $\ddot{u}_p$ は加速度、σ_{pq} は応力テンソル、ρ は密度、f_x は外力を表す。等方完全弾性体の場合、応力と歪み e_{pq} は Lame の定数 λ, μ を用いて次の構成方程式で表現される。

$$\sigma_{pq} = \lambda(e_{xx} + e_{yy} + e_{zz})\delta_{pq} + 2\mu e_{pq} \quad (p, q = x, y, z) \tag{5}$$

$$e_{pq} = \frac{1}{2}\left(\frac{\partial u_p}{\partial q} + \frac{\partial u_q}{\partial p}\right) \quad (p, q = x, y, z) \tag{6}$$

ここで、δ_{pq} はクロネッカのデルタを表す。

今回のシミュレーション計算では、3 次元領域全体を格子間隔 $(\Delta x, \Delta y, \Delta z)$ で分割し、各格子点上に対応した物性値をそれぞれ定めて、上の方程式を解いている。物性値は既知とする。計算領域を z 軸方向に分割し、各プロセスがそれぞれの分割された領域の計算を行っている。

変数の空間微分はスタガード格子モデル上での FDM（有限差分法）に従って行われている。例として、$\sigma_{pq}(x, y, z)$ の x 微分の計算は図のように隣接した変数値、この場合は $\sigma_{pq}(x \pm \Delta x/2, y, z), \sigma_{pq}(x \pm 3\Delta x/2, y, z), \sigma_{pq}(x \pm 5\Delta x/2, y, z), ...$ から、次のように計算を行われている。

$$\frac{\partial}{\partial x}\sigma_{pq}(x, y, z) = \frac{1}{\Delta x}\sum_{m=1}^{M/2} c_m[\sigma_{pq}\{x + (m + \frac{1}{2})\Delta x, y, z\} - \sigma_{pq}\{x - (m - \frac{1}{2})\Delta x, y, z\}] \tag{7}$$

ここで c_m は中央差分の係数を表しており、2 次精度では $c_1 = 1$、4 次精度では (c_1, c_2)=(9/8,-1/24)、8 次精度では $(c_1, c_2, c_3, c_4) = (9/8, -1/24, 49/5120, -5/7168)$ としている。今回のシミュレーションでは、σ_{pq} の x,y 軸方向の微分を 8 次精度、z 軸方向の微分を 4 次精度で行っている。

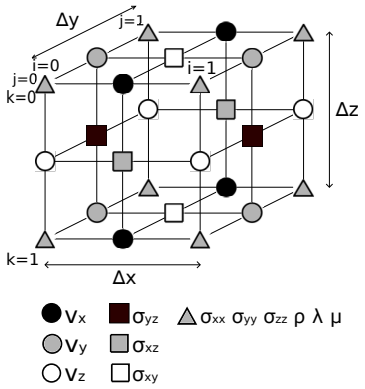


Fig. 5: model of staggerd-grid

時間方向への積分は以下のようにして行う。

$$\dot{u}_p^{n+\frac{1}{2}} = \dot{u}_p^{n-\frac{1}{2}} + \frac{1}{\rho}\left(\frac{\partial \sigma_{xp}^n}{\partial x} + \frac{\partial \sigma_{yp}^n}{\partial y} + \frac{\partial \sigma_{zp}^n}{\partial z} + f_p^n\right)\Delta t \quad (p = x, y, z) \tag{8}$$

$$\sigma_{pq}^{n+1} = \sigma_{pq}^n + \left[\lambda\left(\frac{\partial \dot{u}_x^{n+\frac{1}{2}}}{\partial x} + \frac{\partial \dot{u}_y^{n+\frac{1}{2}}}{\partial y} + \frac{\partial \dot{u}_z^{n+\frac{1}{2}}}{\partial z}\right)\delta_{pq} + \mu\left(\frac{\partial \dot{u}_p^{n+\frac{1}{2}}}{\partial q} + \frac{\partial \dot{u}_q^{n+\frac{1}{2}}}{\partial p}\right)\right]\Delta t$$
$$(p, q) = (x, y, z) \tag{9}$$

この式は現時刻 $(t = n\Delta t)$ の σ_{pq}^n から、時刻 $(t = (n + 1/2)\Delta t)$ の速度値 $\dot{u}_p^{n+1/2}$ を求め、次にこの値を用いて次の時刻 $t = (n + 1)\Delta t$ の σ_{pq}^{n+1} を求めることを意味している。即ち、速度変数 $\dot{u}_p^{n+1/2}$ と応力変数 σ_{pq}^n は $\Delta t/2$ だけずれた時刻に定義されている。

P 波及び S 波を可視化するために、P 波は $\phi_p = \nabla \cdot \dot{\mathbf{u}}$、S 波は $\phi_s = |\nabla \times \dot{\mathbf{u}}|$ を計算する。式として表すと以下の通りである。

$$\phi_p = \nabla \cdot \dot{\mathbf{u}} = \frac{\partial \dot{u}_x^{n+\frac{1}{2}}}{\partial x} + \frac{\partial \dot{u}_y^{n+\frac{1}{2}}}{\partial y} + \frac{\partial \dot{u}_z^{n+\frac{1}{2}}}{\partial z} \tag{10}$$

$$\phi_s = |\nabla \times \dot{\mathbf{u}}|$$
$$= \sqrt{\left(\frac{\partial \dot{u}_z^{n+\frac{1}{2}}}{\partial y} - \frac{\partial \dot{u}_y^{n+\frac{1}{2}}}{\partial z}\right)^2 + \left(\frac{\partial \dot{u}_x^{n+\frac{1}{2}}}{\partial z} - \frac{\partial \dot{u}_z^{n+\frac{1}{2}}}{\partial x}\right)^2 + \left(\frac{\partial \dot{u}_y^{n+\frac{1}{2}}}{\partial x} - \frac{\partial \dot{u}_x^{n+\frac{1}{2}}}{\partial y}\right)^2} \tag{11}$$

4.2 地震波伝播シミュレーションの可視化

シミュレーションコードで計算した ϕ_p 及び ϕ_s の値をボリュームデータとして 3 次元配列に格納した。その変数の可視化コードへの値渡しを行う事で実時間可視化を行った。この際、視点の位置設定を変えて多視点の可視化を実現する。視点はインヤン格子に従って座標の配置を行った。そして、生成された多視点の連番画像を視点ごとに動画に圧縮した。これにより、データ転送の時間を大幅に短縮した。以下ではこの並列ボリュームレンダリングと視点配置の設定、及び動画圧縮の手順を示す。

並列ボリュームレンダリングの流れは Fig.6 の通りである。最初に可視化コードが呼び出されたとき、設定ファイルを読み込む。その内容を元に、係数やボクセル数、地形データやカラーテーブル等の読み込みや必要なメモリの確保といったプリプロセスの処理を行う。また、ボリュームデータの値渡しがされるたびに、ボリュームデータの正規化を行う。そして正規化したボリュームデータから、ボリュームデータ間の接合面のデータを、MPI 通信で交換する。その後、視点の設定とそのレンダリング処理を視点ごとに繰り返し行う。視点の座標の代入は格子を Yin、Yang の順番で行う。今回はそれぞれ経度 $\pi/4$ から $7\pi/4$ までの $\pi/8$ 間隔、余緯度 $\pi/4$ から $3\pi/4$ までの $\pi/8$ 間隔で、$2 \times 13 \times 5$ 個の視点位置を設定した。

各視点ごとに投影画面の上方向のベクトル $\vec{u}$ を設定する必要がある。$\vec{u}$ はその視点から極座標系の中心へのベクトルとその視点が属する Yin もしくは Yang の座標系の緯線に垂直であり、余緯度 0 の極点の方をさす。レンダリングされる地形モデルの中心を原点とし、視点の座標を (x, y, z) とする。このとき Yin 格子上の視点の場合、$\vec{u}$ はカーテシアン座標で表すと、

$$\vec{u} = (u_1, u_2, u_3) = \begin{cases} (\frac{x}{x^2+y^2}, \frac{y}{x^2+y^2}, -\frac{x^2+y^2}{x^2+y^2+z^2}) & (z > 0) \\ (-\frac{x}{x^2+y^2}, -\frac{y}{x^2+y^2}, -\frac{x^2+y^2}{x^2+y^2+z^2}) & (otherwise) \end{cases} \tag{12}$$

となる（Fig.7）。Yang 格子上の視点の場合は、Yang の座標系上で同様に考え、それを Yin の座標系に変換する。即ち、

$$\vec{u} = (-u_1, u_3, u_2) \tag{13}$$

とすればよい（Fig.8）。

そして各プロセスは定められた位置から担当のボリュームデータをレイキャスティング法によって投影計算を行い、2 次元配列に格納する。このとき各プロセスは z 軸方向に分割された地下構造のボリュームデータのレンダリングデータの配列をそれぞれ保持していることになる。その配列をプロセス数で分割する。最終的に生成される画像データはプロセス数分割された区画をプロセス 0 がまとめることで生成される。プロセスによってその区画は決まっている。各プロセスは

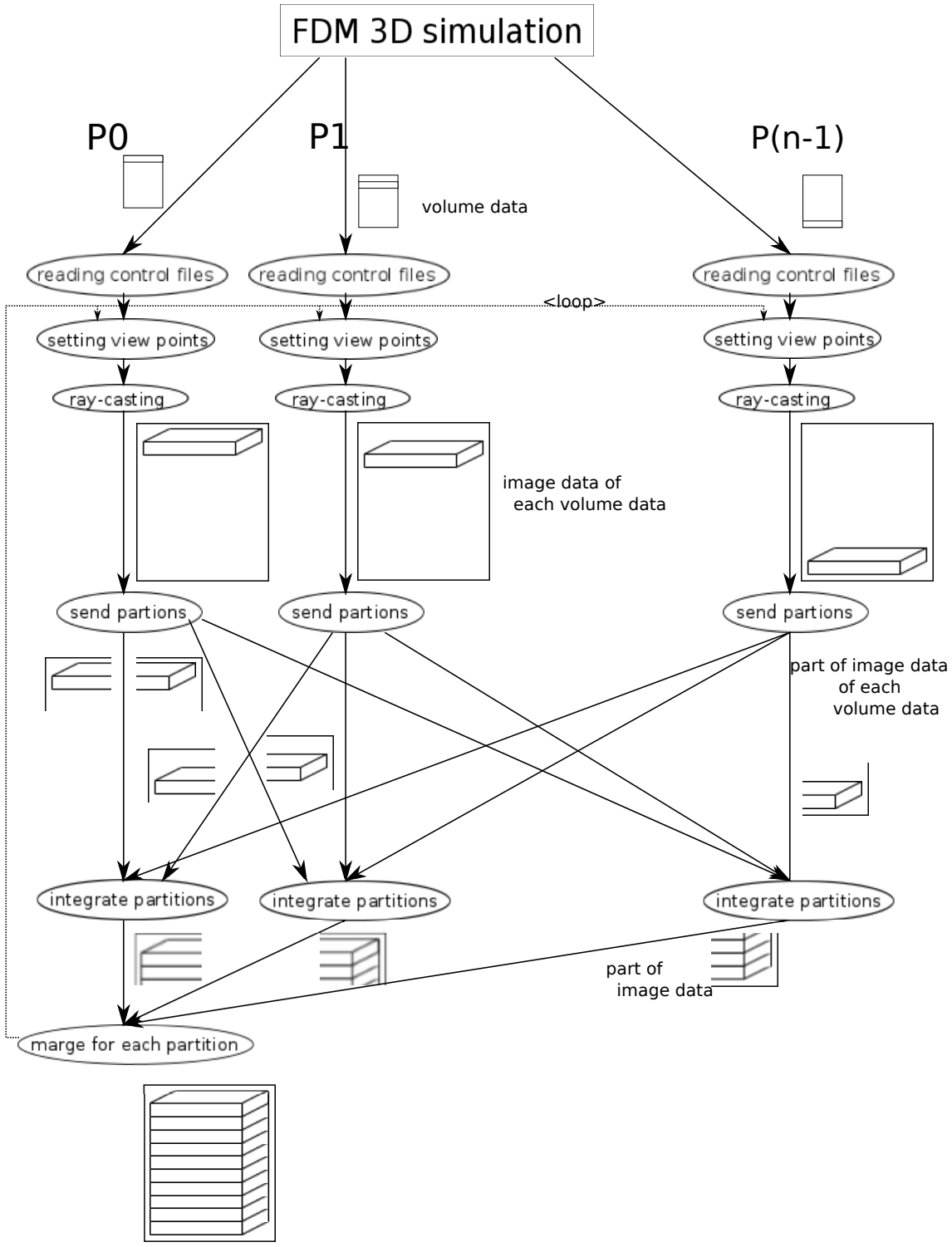


Fig. 6: Chart of pallarel volume rendering

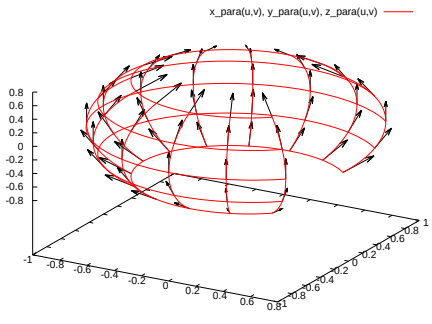


Fig. 7: u vector on yin

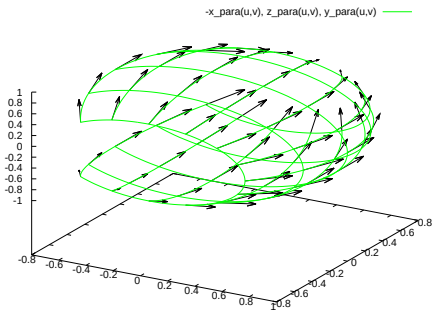


Fig. 8: u vector on yang

分割された配列をそれぞれ該当する区画を担当するプロセスに送る。最後にプロセス 0 がそれらの区画のデータをまとめあげ、画像データとして出力できる。

並列ボリュームレンダリングによって、130 視点で各視点ごとに 60 枚の PPM 形式の連番画像が生成される。そのデータを FFmpeg によって 130 個の MPEG-4 データに変換した。ffmpeg はコンソール上のコマンドとして連番画像から 1 つの動画を生成するのに使われるため、シェルスクリプトを併用することで複数の動画への変換を行っている。オプションの設定として、入力動画と同等の品質になるように設定し、コーデックは MPEG-4 を指定している。後は scp コマンドで操作 PC 上に転送し、今回作成したムービープレーヤで動画の再生を行った。

4.3 P 波と S 波の可視化手法

ϕ_p と ϕ_s の 2 種類のボリュームデータを、1 つのボリュームデータとして可視化するために、前節の手順に幾つかの調整を加えた。

まず ϕ_p と ϕ_s の値を、次の手順で 1 つのボリュームデータとしてまとめた。3 次元配列 w の表記を $w[i][j][k]$ とした時、$(i+j+k) \bmod 2 = 0$ の格子点には ϕ_p の値を、そうでないときには $\phi'_s = \phi_s + \epsilon$ を代入する。ここで ϵ は ϕ_p の最大値よりも大きな値である。今の場合 ϵ =1.00000003E-04 とした。この結果、ボリュームデータにおける ϕ_p と ϕ_s の配置は Fig.9 のように互い違いになる。

この ϕ_p と ϕ_s の値が混合したボリュームデータを可視化するにあたり、次に示す手続きによって、ϕ_p の値域と ϕ_s の値域を分離した。ボリュームデータの値が ϵ 以上であれば ϕ_s、そうでなければ ϕ_p と判別する。正規化したデータは 0 から 255 までの値域をもち、そのうち ϕ_p は 1 から 99 までの値域を、ϕ_s は 100 から 199 までの値域を、また地形データはそれ以外の値域をとるようにすればよい。c_p と c_s

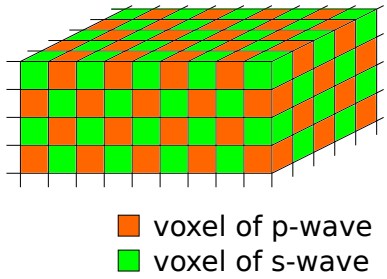


Fig. 9: voxel model of p-wave and s-wave

をそれぞれ ϕ_p と ϕ_s の仮の最大値として定義する。x を変換されるボリュームデータの値とすると、正規化関数 $f(x)$ は次の通りである。

$$f(x) = \begin{cases} 100 + 99(\frac{x-\epsilon}{c_s})^{\frac{1}{4}} & (x \geq \epsilon) \\ 50 + 49(\frac{x}{c_p})^{\frac{1}{4}} & (\epsilon > x \geq 0) \\ 50 - 49(\frac{-x}{c_p})^{\frac{1}{4}} & (x < 0) \end{cases} \tag{14}$$

Fig.10 は $c_s = c_p = 10^{-7}$ としたときの f(x) のグラフである。緑色で示されているのが ϕ_s、オレンジ色で示されているのが ϕ_p の関数である。ϕ_p の正規化によって、ϕ_p が非負の値であれば 50 から 99 までの値域をとり、負の値であれば 1 から 50 までの値域をとる。ここでキーへの変換に 4 乗根を使うことで、基準値 c_p,c_s を超えない限りではあるが、対数関数に近い性質をもつ。Fig.11 のグラフは横軸を対数として、$c_p = 10^{-5}$ のとき、ϕ_p の非負の値の、正規化関数 $f(x)$ である。

なお、ボリュームデータの正規化を行ってから地形データの正規化を行っている。このようにしてボリュームデータは ϕ_p、ϕ_s、そして地形データの値域を分離して、正規化される。

そして、正規化されたデータから、正規化する前の ϕ_p の絶対値及び ϕ_s の値が最大である格子点上で不透明度が最大になるように不透明度を決めた。まず、MPI 通信で、地形データの値域を除いた、正規化したデータの最大値と最小値を計算する。求めた最大値は ϕ_s の最大値を、最小値は ϕ_p の最小値を正規化した値とみなせる。それらの値を o_s,o_p とおく。正規化した値 x の不透明度の関数 $f(x)$ は次の通りである。

$$f(x) = \begin{cases} (\frac{x-100}{o_s-100})^2 & (o_s \geq x \geq 100) \\ (\frac{x-50}{o_p-50})^2 & (100 - o_p \geq x \geq o_p) \\ 1.0 & (200 > x > o_s, 100 > x > 100 - o_p, o_p > x > 0) \end{cases} \tag{15}$$

この式は ϕ_s の場合、100 から o_s へ増えていくにつれ ϕ_s が見えるようになり、ϕ_p

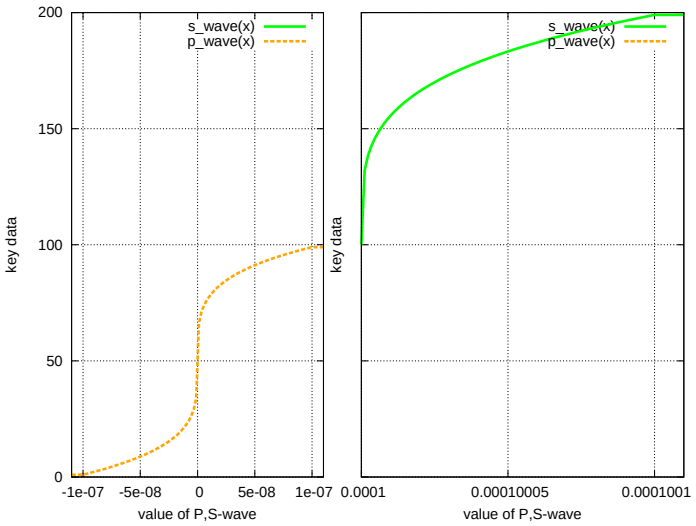


Fig. 10: Normalization of p-wave and s-wave

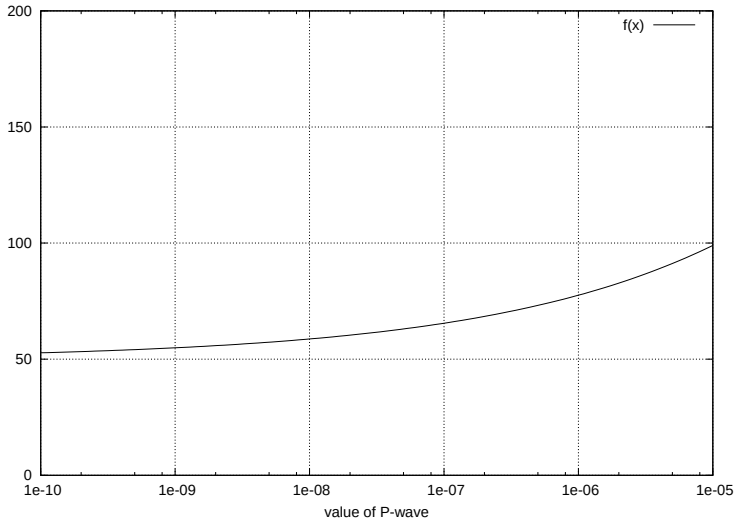


Fig. 11: Nomalized p-wave

の場合 50 から $100 - o_p$ へ増えていくもしくは o_p へ減っていくにつれ ϕ_p が見えるようになる事を意味する。グラフとして表すと Fig.12 の通りになる。

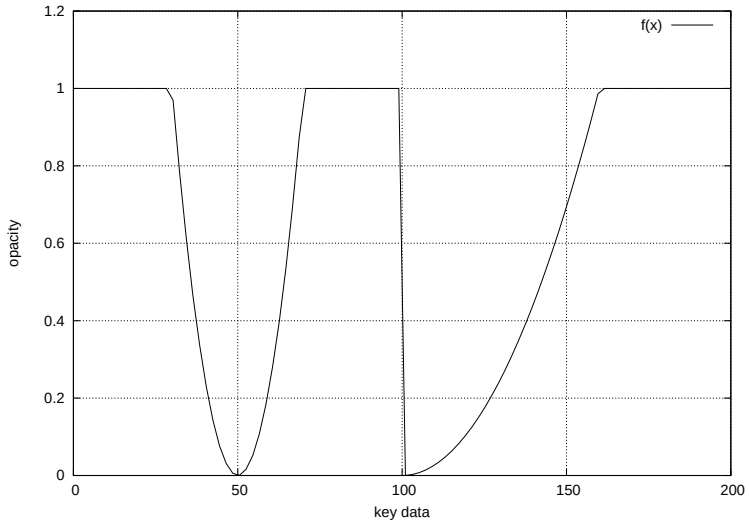


Fig. 12: Opacity of data

上述の手順によって、色の値は時間ステップに関わらず ϕ_p と ϕ_s の値で決まり、不透明度は各時間ステップにおける ϕ_p と ϕ_s の最大値で規格化されるようにした。このようにしたのは時間発展による、ϕ_p と ϕ_s の強さの減衰とそれに伴う波形の変化を可視化するためである。色の値も不透明度と同様に規格化すると、どの時間ステップでもくっきりとした波形を見る事ができるが、色が示す強さは同じではなくなってしまう。

4.4 可視化結果

今回の検証では、神戸大学システム情報学研究科計算科学専攻のワークステーション（VT64 WorkStation 4600 O）を使用した。

Computer type	VT64 WorkStation 4600 O
CPU	AMD Opteron 6176 SE(12cores)×4sockets
Memory	128GB
OS	CentOS 5.5

Table. 1: Specification of the computer

ジョブは、プロセス数を 40、使用コア数を 40 に設定した。地震波動場のシミュレーション計算は全部で 6000 ステップで 100 ステップごとに可視化のコードを呼び出した。シミュレーション計算によって $256 \times 512 \times 160$ のボリュームデータを用意した。可視化設定では、130 個のカメラで各 60 枚の PPM 形式への画像のレンダリングを行った。地形データの大きさは $512 \times 1024 \times 320$ で設定した。地形データを統合した後のボリュームデータの大きさを $256 \times 512 \times 160$ に設定した。画像の大きさはレンダリングにおいて、地形画像を含めたボリュームデータ全体を投影できる大きさを計算している。レイキャスティングの環境光、拡散光、不透明度の振幅値といった係数はそれぞれ $1.0, 0.5, 0.5$ に設定した。鏡面反射光については計算を行っていない。これらの設定をまとめると以下の通りである。

Model name	Tottori
Voxel model size	$256 \times 512 \times 160$
Expect voxel size	$256 \times 512 \times 160$
Background Voxel model size	$512 \times 1024 \times 320$
Ambient coef	1.0
Diffuse coef	0.5
Opacity coff	0.5
Image format	PPM

Table. 2: Vizualization settings

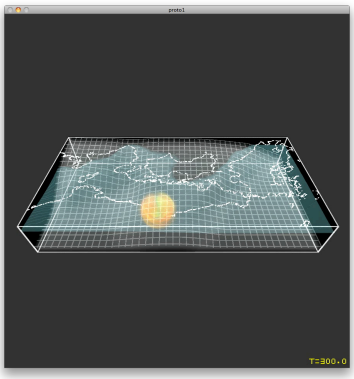
これらの設定で多視点実時間可視化のジョブを流した結果、全実行時間は 5 時間 17 分 32 秒だった。シミュレーション計算上の 1000 ステップ目で、可視化にかかった時間は 221 秒であった。可視化処理の処理時間は 1 つのタイムステップにおける処理時間から、全部で 3 時間 40 分と算出された。PPM 形式での出力時におけるデータ量は 19GB、MPEG-4 形式への変換を行うと 238MB のデータ量となった。そして、本実験で作成した動画プレーヤで再生した。

Fig.13 の写真群は Yin 座標系上における $(\theta, \phi) = (-\pi/4, \pi/2)$ から右キーを押し続けた結果の時間発展の動画のスナップショットである。(厳密には、この動画プレーヤには再生中断機能があり、中断する、スナップショットを撮影する、右キーを押して次の視点に切り替える、動画を 3 フレーム分再開するという作業を繰り返した結果である。実際には視点を回転させながら時間発展を見る事ができる。) カラーテーブルの設定は、ϕ_p の、負の値として最小値の色は (255,191,0) の黄土色、0 の値の色は (255,127,0) の橙色、正の値として最大値の色は (255,63,0) の朱色にした。ϕ_s は 0 の値の色は (0,255,255) の水色、最大値の色は (0,127,0) の緑色にした。

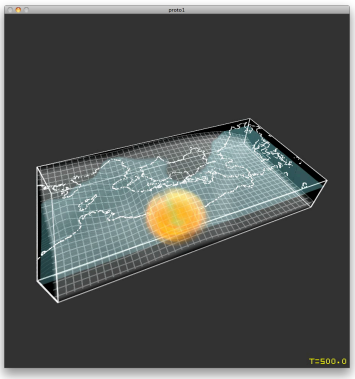
6 枚目から 7 枚目、10 枚目から 11 枚目の視点に移るとき視点位置のずれが起こった。それ以外は滑らかに視点を移動していた。1 枚目と 19 枚目の画像を比較すると経度方向の視点位置は同じであるが、余緯度方向の視点位置が 19 枚目が下に来ていた。また 10 枚目までのスナップショットにおいて橙色の周辺で数層にわたる縞模様が同心円上に広がる様子が見られた。その内側に緑色の模様が同心円上に広がる様子が見られた。

視点位置のずれが起こったのは、それぞれ Yin 座標系上の視点から Yang 座標系上の視点へ、Yang 座標系上の視点から Yin 座標系上の視点へ切り替わったからである。1 枚目と 19 枚目の視点位置の違いは、各座標系のメッシュの境界まで視点位置を移動してから座標系を変えるという方法では、元の視点位置に戻らない事を示している。

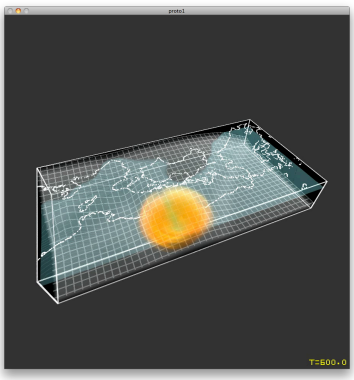
P 波の粗密波が橙色の縞模様として表されている。また S 波は P 波より遅い事が、緑色（S 波）の分布から分かる。



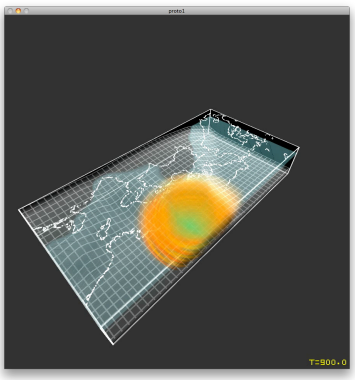
(a)



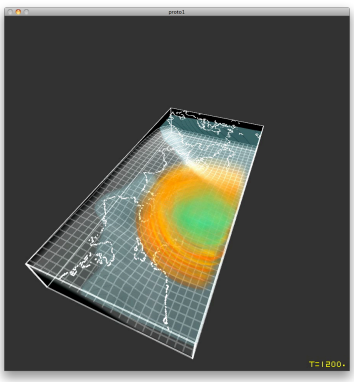
(b)



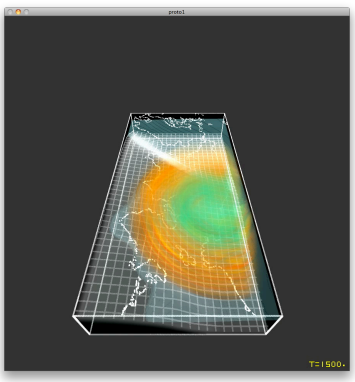
(c)



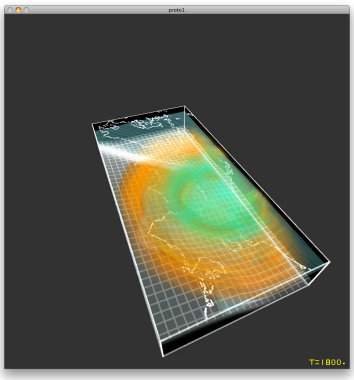
(d)



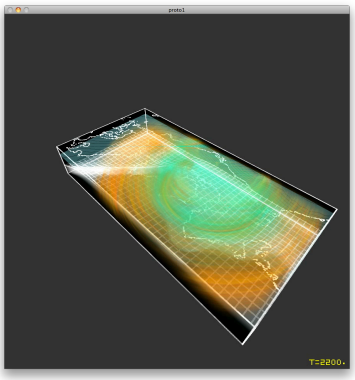
(e)



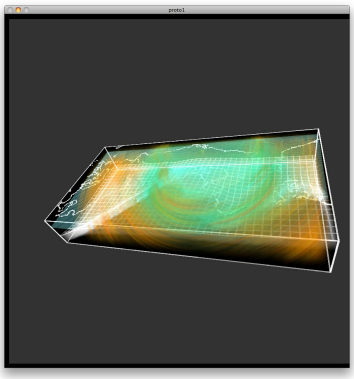
(f)



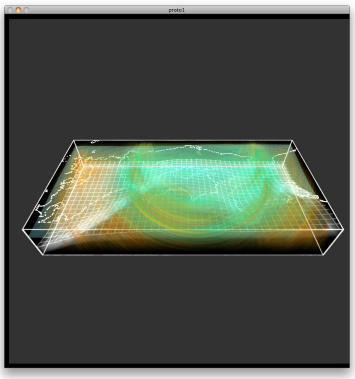
(g)



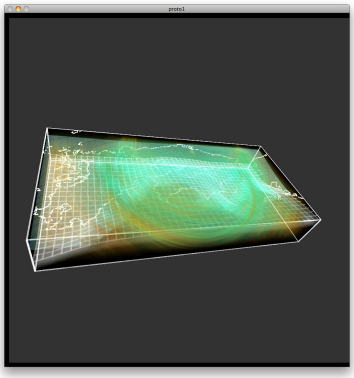
(h)



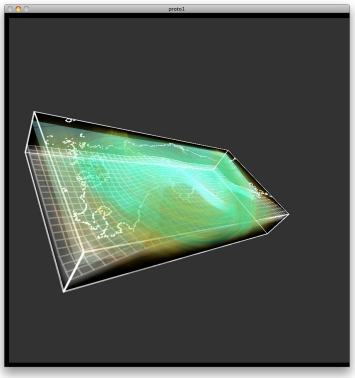
(i)



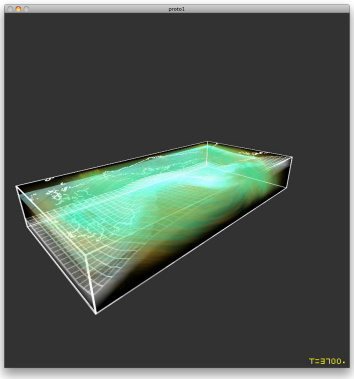
(j)



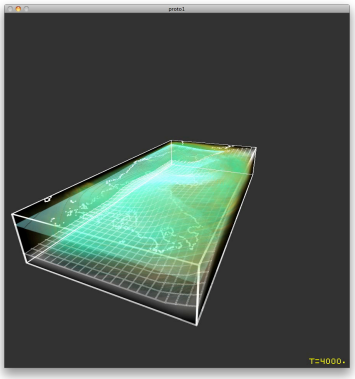
(k)



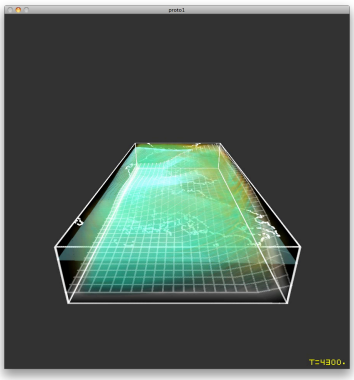
(l)



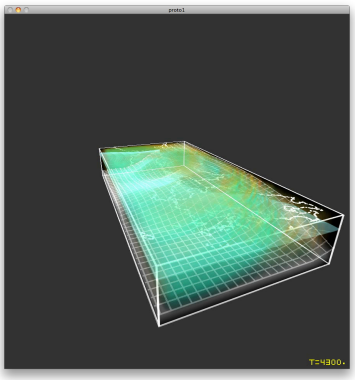
(m)



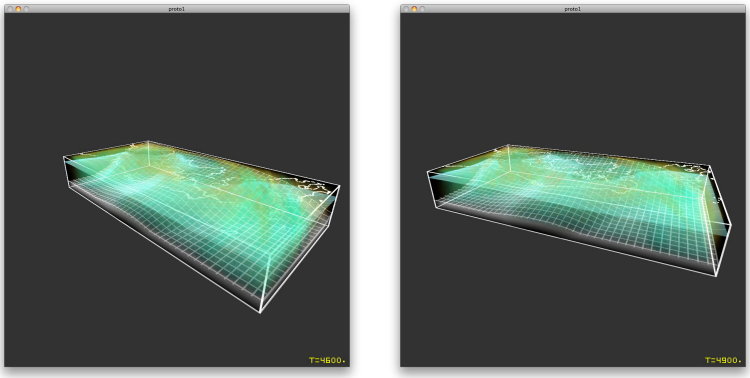
(n)



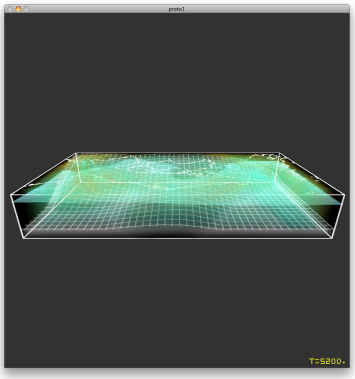
(o)



(p)



(q) (r)



(s)

Fig. 13: Snapshot sequence from the movie player

5　まとめ

本研究では、スーパーコンピュータを用いた大規模な計算機シミュレーションデータの新しい可視化手法を提案した。それは、実時間可視化を多視点で行うというものである。この手法を実現するためには、固定視点動画ファイル群から、対話的にデータを抽出する動画プレーヤが不可欠である。そこで OpenCV ライブラリを使って、多視点実時間可視化用の動画プレーヤを開発した。視点の配置は Yin-Yang 格子を利用した。これにより、各視点をほぼ等間隔に配置する事が可能となった。

応用として鳥取県西部地震 3 次元地震波伝播シミュレーションの多視点実時間可視化を行い、その出力データを MPEG-4 形式の動画像に変換することでデータの転送量を 1/100 程度にすることができた。そして転送した動画データ群を先ほどの動画プレーヤで再生した。このようにして P 波及び S 波の地震波伝播の様子を視点を切り替えながら見ることができた。今後はマウス操作から自由度の高い視点の切り替えを行えるようにする。また地震波以外でのシミュレーションで、この手法による可視化を行っていく予定である。

6 謝辞

1年間を通して熱心に指導していただいた、陰山聡教授と政田洋平助教に深く感謝いたします。東京大学地震研究所の古村孝志教授には今回の実証実験における鳥取地震のシミュレーション及びボリュームレンダリングコードを提供していただきました。また、P波とS波の分離法について地震学の立場からアドバイスを頂きました。東京大学地震研究所の岩井一央氏には、P波とS波の分離方法についてアドバイスを頂きました。神戸大学山本研究室の廣田悠輔氏には、レンダリングコードにおけるメモリの使用法に関するアドバイスをいただきました。陰山研究室の山浦優気氏にはボリュームレンダリングの方法についてアドバイスをいただきました。目野大輔氏にはFFmpegやOpenCVのインストールに関してのアドバイスをいただきました。山田耕平氏にはOpenCVに関するアドバイスをいただきました。西田泰大氏にはLaTeXの扱い等をはじめとしたアドバイスをいただきました。皆様に心から感謝いたします。

参考文献

[1] http://www.top500.org

[2] Kageyama A, Sato T, "'Yin-Yang grid': An overset grid in spherical geometry ", *GEOCHEMISTRY GEOPHYSICS GEOSYSTEMS*, VOL. 5, Q09005, pp3-5, 2004

[3] Marc Levoy, "Display of surfaces from volume data ", *Computer Graphics and Applications*, IEEE, pp.30-33, 1988

[4] http://www.ffmpeg.org

[5] Robert A. Drebin, Loren Carpenter, Pat Hanrahan, "Volume rendering", *SIGGRAPH '88 Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 1988

[6] 古村孝志, "地震波伝播と強震動の大規模並列FDMシミュレーション", 東京大学情報基盤センタースーパーコンピューティングニュース *Vol11*, pp1-7, 2009

[7] H. UEHARA, S. KAWAHARA, N. OHNO, M. FURUICHI, F. ARAKI, A. KAGEYAMA,"MovieMaker: a parallel movie-making software for large-scale simulations", *Journal of Plasma Physics*, 72, pp841-844, 2006

[8] T. Furumura, L. Chen, "Large Scale Parallel Simulation and Visualization of 3D Seismic Wavefield Using the Earth Simulator", *Computer Modeling of Engeneering and Sciences*, 6, 2, 153-168, 2004