J. Plasma Fusion Res. Vol.96, No.4 (2020) 199-206

研究論文

4次元ストリートビュー：計算機シミュレーションの新しい可視化法*

4D Street View: A New Visualization Method for Computer Simulations

陰山　聡，坂本尚久，大野暢亮[1)]
KAGEYAMA Akira, SAKAMOTO Naohisa and OHNO Nobuaki[1)]
神戸大学システム情報学研究科，[1)]兵庫県立大学シミュレーション学研究科

（原稿受付：2020年2月4日／原稿受理：2020年3月11日）

得られた数値データを一度ディスク等に保存し，それを後で可視化処理するのが現在の計算機シミュレーションでは一般的である．シミュレーション規模の増大とともにこの方法が難しくなってきていることからシミュレーションをしながら可視化を行う「その場」（in-situ）可視化と呼ばれる手法が最近注目を集めはじめている．対話的な解析ができないのがこの手法の弱点である．我々は，多数の全方位カメラをシミュレーション空間に配置することで対話的なその場可視化を可能にする新しい可視化法「4次元ストリートビュー」を提案する．

Keywords:
data visualization, scientific visualization, high performance computing, in-situ visualization, supercomputing

1．はじめに

データ可視化はシミュレーション研究に必須である．シミュレーションの規模が大きくなるにつれ，可視化に費やす研究者の労力と時間も増大する．大規模なシミュレーションになると，シミュレーションジョブの実行時間よりも，その出力数値データを転送したり，可視化処理したりするのにかかる時間の方がずっと長いという場合も珍しくない．

現在ではシミュレーションの実行後に数値データを転送・保存し，それに可視化処理を施すのが普通である．事後（post-hoc）可視化と呼ばれるこの可視化のスタイルはこれまで主流であったが，近い将来そうではなくなるであろう．計算規模が大きくなるにつれて，転送・保存すべき数値データのサイズ，そしてその可視化処理にかかる時間がますます増大することが予想されるからである．

そこで近年，その場（in-situ）可視化と呼ばれる別の可視化スタイルが大規模シミュレーション研究者の注目を集めている[1-3]．その場可視化では，シミュレーションの実行中に可視化の処理も行う．つまり数値データではなく画像がシミュレーションの結果として生成されるので，研究者はこれらの可視化画像をすぐに見ることができる．

次のような議論からも事後可視化からその場可視化への切り替えは必然と言える．（なお，以下では空間3次元で時間発展型のシミュレーションを想定する．）

1つの次元方向の空間離散点数Nをシミュレーションの空間解像度で表すことにする．事後可視化のために出力すべき単精度（4バイト）の数値データのサイズは，時刻ステップあたり$O(64N^3)$である．画像の解像度も一つの次元，つまり画像の縦または横方向のピクセル数Mで表すと，その場可視化で出力される画像データのサイズは1ピクセルバイトとして$O(16M^2)$である．小さいNの計算しかできなかった時代には3次元の数値データをそのまま保存するのは十分可能であったし，また自然なことでもあった．しかし，スーパーコンピュータの性能の向上とともにNの値は伸び続けており，$O(64N^3)$の3次元数値データを出力することは大きな負担となってきている．一方，可視化画像の解像度Mは基本的には上限があるといってよい．どれほど高い解像度計算のシミュレーションをしたとしても，結局のところ研究者が「見る」のは相変わらずPCのモニタ画面や紙である．一枚の可視化画像の解像度はM＝1,000から2,000程度あれば十分で，これ以上高くしてもあまり意味がない．

M＝2,000とすると$64N^3=16M^2$となるNは100である．つまり100^3格子点の3次元シミュレーションの場合，3次元数値データひとつ（1時刻のスナップショット）を保存することは2000^2ピクセルの（ほぼ実用限界の）高解像度画像ファイルを（圧縮しないで）1枚保存するのと少なくともデータサイズという意味では同じである．もちろんN＝100の時代は既にはるか昔であり，現在はN＝1,000の計算があたりまえである．まもなくN＝10,000の計算も日常的になっていくであろう．N＝10,000シミュレーションの3次元数値データを保存するディスクスペースにはM＝2,000の画像を100万枚保存できる．しかも画像データには効果的な圧縮アルゴリズムが存在する．画像を10%に

＊本論文は第36回プラズマ・核融合学会年会の招待講演の内容をまとめたものである．

Kobe University, Kobe, HYOGO 657-8501, Japan
corresponding author's e-mail: kage@port.kobe-u.ac.jp

圧縮すれば，3次元数値データを一つ保存するディスクスペースに，1000万枚の画像データが保存できる．

このようにデータサイズの点からは事後可視化よりもずっと有利であることが明らかであるにも関わらず，その場可視化が大規模シミュレーションの可視化スタイルとしていまだに主流とはなっていないのには理由がある．それは対話性の欠如である．対話性とは，PCの画面に表示された可視化画像を見ながら，マウス等を使って視点の位置や視線の方向を変えたり，あるいは可視化対象の物理量や，適用する可視化手法（たとえば等値面やボリュームレンダリング等）とそのパラメータをリアルタイムに変更できることを意味する．変更を指示したら瞬時にその結果が画面に反映されなければならない．

効率的な可視化には上の意味での対話性が不可欠である．だが，その場可視化手法では，シミュレーションジョブを投入するときに可視化に関する全ての設定を決めなければならない．つまりあらかじめ決めた視点位置と視線方向に可視化用のカメラを設定し，あらかじめ決めた可視化手法と対象の物理量を可視化するというのがその場可視化である．シミュレーションジョブが終了した後，出力結果として得られた可視化画像を見て，「この現象は面白い．別の角度から見てみよう」と思った場合，可視化視点と視線方向を設定し直してもう一度シミュレーションジョブをスーパーコンピュータに投入する必要がある．

したがって，その場可視化手法に対話性を導入することが最近の可視化研究の大きなテーマである．レイヤードイメージ法[4]，粒子ベースのレンダリング[5,6]，イメージベースの可視化[7]など，様々な手法が提案されている．

我々も対話的なその場可視化を実現する方法を提案した[8]．その基本的なアイディアは，「その場可視化を採用する限り，ディスクスペースは有り余るほどあるのだから，後で見たくなるかもしれない可視化画像を全て出力してしまえばよい」というものである．この手法を提案した当時は対話性として視点位置の変更を主に考えていたが，その後，視線方向や適用する可視化手法も対話的に変更できるよう大きく改善した．我々のもともとの発想は別のところにあるが，この可視化法の原理がGoogleのストリートビュー[9]に似ており，それを（地表の）2次元から（時空間の）4次元に拡張したものと表現することもできるので，我々はこの手法を「4次元ストリートビュー（Four-Dimensional Street View）」と呼んでいる．

2．4次元ストリートビューの考え方

4次元ストリートビューでは，可視化のプロセスを「撮るフェーズ」と「見るフェーズ」という2つの段階に分ける．「撮るフェーズ」は，その場可視化の画像群を作る段階であり，シミュレーションプログラムから呼び出す専用のライブラリで実現する．そうして作られた画像ファイル群を一種のデータベースとし，対話的に探索するのが次の「見るフェーズ」である．これはPC上の専用アプリで実現する．

上に述べたように，通常のその場可視化手法では興味深い現象が起きると想定される位置の近くに可視化カメラを設定し，その方向にあらかじめ視野角を設定した上でシミュレーションジョブを流す．想定外の画像が得られた場合にはもう一度設定し直してジョブを再投入なければならない．

一方，4次元ストリートビューでは，可視化視点を千個から数万個，あるいはそれ以上に設定する．そしてその可視化用のカメラを仮想的な全方位カメラにする．**図1**（a）にシミュレーション空間に配置した多数の全方位カメラの模式図を示した．シミュレーション領域の外側に配置したカメラからはその方向から見えるシミュレーションの全体像が記録され，内部に配置したカメラからはその近傍で起きる現象が詳細に記録される．ここではカーテシアン格子的に3次元全方向，等間隔にカメラを配置しているが，必ずしもこうする必要はない．重要な現象が起きる位置があらかじめわかってる場合にはその近傍に全方位カメラを多めに配置することもできる．

全方位カメラとは，ある視点から見えるすべての風景，つまり全視野角（4π ステラジアン）の画像情報を一度に撮影するカメラである．その情報は通常のカメラと同様に二次元のピクセルデータとして一つのファイルに保存するが，全方位を強引に長方形に収めたそのファイルをそのまま見ても人間の目には画像が歪んで見えるのであまり意味がない．メルカトル図法で描いた地図が北極と南極近くで大きく歪んでいるのと同様である．全方位画像ファイルを見るときには，ユーザが指定した視線方向を一定の視野角で切り取り，適切な座標変換をかけて表示する必要がある．その切り取りと変換が高速に処理されれば，ユーザがマウス等を使ってすばやく視線方向を変更してもリアルタイムに見たい方向の風景を画面に表示させることができる．全方位画像は近年，主にエンターテインメント関連で注目を集めており，数万円で全方位カメラを買うことができる．全方位画像の規格（Omnidirectional MediA Format, OMAF）も決まりつつある[10]．

その場可視化用の仮想的な全方位カメラをシミュレーション領域の内外に多数，散布させるのが4次元ストリートビューの「撮るフェーズ」である．シミュレーション終

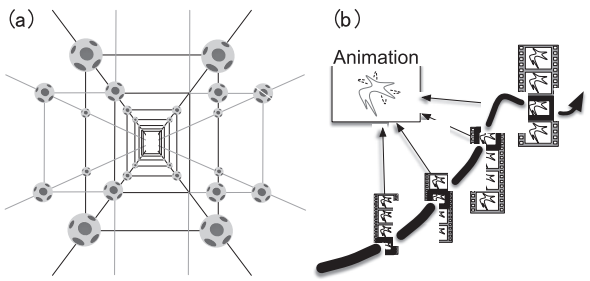

図1　我々が提案する新しい可視化手法「4次元ストリートビュー」の概念．(a)シミュレーション空間に多数の全方位可視化カメラを散布し，それぞれの視点から全方位（4π ステラジアン）でin-situ可視化を行う．時間経過は動画ファイルで保存する．(b)次の「見る」段階においてはユーザの指定にリアルタイムに応じて，動画ファイルの個別のフレーム（静止画）列を抽出する．

了後には視点の数だけの全方位可視化動画ファイル群が生成される．散布された全方位カメラの密度が十分に高ければほぼ任意の位置から見た「風景」が動画ファイル群の中に収められているはずである．一つ一つは全方位画像なので，そこから好きな方向を対話的に抜き出す，つまり見回すことも自由にできる．

それぞれの全方位動画ファイルには視点位置，つまりその画像がどの位置の仮想カメラで撮影された全方位動画なのかという位置情報を付加する．4次元ストリートビューの最初の段階，つまり「撮るフェーズ」は，この動画のデータベースの生成が最終目的である．

この動画データベースは3次元空間に分布した全方位「動画の場」とみなすことができる．そして一つの動画は多数の静止画を時間方向に積み重なったものなので，これは4次元時空間に分布した「静止画の場」とみなすこともできる．コンピュータグラフィクスの世界ではこの「場」をplenoptic functionと呼ぶ．空間（表示のために2次元とした）と時間1次元の中に分布する静止画の場の概念図を**図2**に示した．空間の各点にはその位置で可視化した画像が置かれている．シミュレーションの進行と共に蓄積されていく画像は時間方向に座標位置を変えて配置する．4次元ストリートビューの次の段階，つまり「見るフェーズ」は，この「4次元時空間中の全方位静止画の場」を対話的に探索する段階である．

4次元ストリートビューのユーザは，PCのマウスなどを使って4次元空間中に任意の経路を指定する．そのような経路の一例を**図2**の中の矢印付きの曲線に示した．すると専用のアプリケーション「動画データブラウザ」が，その経路上に乗っているか，あるいは近いところに位置する静止画，つまり特定の動画ファイル中の特定の時刻フレームの全方位静止画列を次々とデータベースから取り出す．**図1**(b)の矢印付きの太い曲線はユーザが指定した経路を表す．この経路上に位置する画像を4次元時空間中の全方位静止画像の場から抽出し，次々と画面に表示する（細い矢印）．その画像は全方位画像なので，マウス等を介したユーザからの指示に応じて一定の視野角画像を切りとり，適切な座標変換をかけてから画面に表示する．動画データベースからの全方位静止画フレームの取り出しと，視野角の切りとりが瞬時に行われれば，その画像列はなめらかなアニメーションとして画面に表示されるであろう．したがってユーザはあたかもシミュレーション空間中を自由に動きながら現象をリアルタイムで観察しているように感じられる．

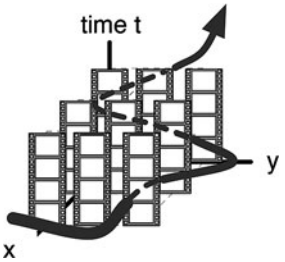

図2　多視点・全方位カメラでin-situ可視化した結果得られる動画ファイル群に，各視点の位置情報を紐付けした動画データベースは4次元時空間中の全方位静止画の「場」とみなすことができる．

表示された画面を見て，面白い現象が観察されたら，その近くまで視点を移動させればよい．その移動はマウスクリックでもできるしキーボードに割り当てたキーを押すことでもできる．ある視点の動画ファイルを再生中に隣の視点に移動するようユーザが指示をすれば，画面の中の動画再生を止めることなく瞬時に移動が実現される．つまり，入力動画ファイルを切り替え，次の時刻フレームの画像をそこから取り出して画面に表示する．

動画ファイルから任意の時刻フレームの画像を瞬時に取り出すことができるので，空間を移動しながら，時間を遡ることができる．興味深い現象を見つけたら，その近くに移動し，周囲を見渡して，その現象を引き起こした原因を過去にさかのぼって探ることができる．

3．実装方法

以上で述べたように，4次元ストリートビューはアイディアとしては簡単であるが，実現するのは容易ではない．開発すべきソフトウェアが2つあり，それぞれ「撮るフェーズ」と「見るフェーズ」に対応している．「撮るフェーズ」を実現するためには，千個から数万個の視点でのその場可視化を，しかも全方位で実行するスーパーコンピュータシミュレーション用のライブラリが必要である．一方，「見るフェーズ」を実現するためには，上述の機能を実装したPC用のアプリ「動画データブラウザ」が必要である．以下ではこの二つのツールの開発について紹介する

3.1　多視点・全方位・その場可視化ライブラリ

スーパーコンピュータ上でその場可視化をするためのツールやフレームワークとして，Catalyst[11]，libsim[12]，ADIOS[13]，Embree[14]，OSPray[15]など既にいくつか開発されている．その中で我々はVISMO[16]を使うことにした．VISMOには等値面，ボリュームレンダリング，流線，グリフなど基本的な可視化手法はほとんど実装されており，大規模シミュレーション用の可視化ライブラリとして当然のことながら，領域分割されたシミュレーションの並列可視化にも対応している．VISMOが特徴的なのは，これらの可視化機能がほとんど全てFortran言語だけで書かれており，VISMO自身からは下位のライブラリを呼びだす必要がないという点である．つまりFortranコンパイラさえあればスーパーコンピュータ上で手軽にその場可視化が実現可能である．VISMOを使った最近のその場可視化のシミュレーション例として[17,18]がある．

下位ライブラリを必要としないというのは実用上，大変大きな利点であることを強調しておきたい．スーパーコンピュータシミュレーション用のその場可視化ライブラリは上述のようにVISMO以外にもいくつか存在し，それらはオープンソースのものがほとんどだが，その内部で下位ライブラリを呼びだすため，まずはその下位ライブラリのインストールから始めなければいけない．たとえば多くのそ

の場可視化ライブラリではOSMesaが必要である．これは本来GPU（シェーダ）で機能するOpenGL関数群をCPUでエミュレートするためのライブラリである．我々は複数のスーパーコンピュータシステムを利用しており，4次元ストリートビュー手法を当然それら全てのシステムで利用できるようにしたい．しかしOSMesaのインストールはどのシステムでも大変な労力と時間が必要で，結局はあきらめたシステムもあった．一方，VISMOはFortranコンパイラさえあればよいのでのインストールと利用は極めて容易である．

VISMOはもともと複数の視点を指定する機能を持っているが，全方位可視化機能は備わっていない．そこで我々はcube mapping[19]手法に基づき一つの視点から6つの方向に対して通常の可視化をすることでVISMOによる全方位可視化を実現した．cube mappingとは，視点の周囲に仮想的な立方体を設定し，その立方体の6つの面（正方形）に対して通常の透視射影計算を行うものである（**図3**）．その結果，一つの視点に対して6枚の可視化画像（それぞれ視野角90度の正方形画像）が生成される．現在ではこの6枚の画像をシミュレーション終了後に1枚の全方位画像に融合する処理をしてから後述する動画データブラウザに渡す．なお，VISMO内部で一度に全方位可視化画像をつくる機能をVISMOに追加する研究開発を現在進めている．

3.2 動画データブラウザ

動画データブラウザはKVS[20]のフレームワークの上で開発した．KVSはもともと汎用可視化ソフトとして開発されたC++のクラスライブラリであるが，今回のように画像データを処理する各種のアプリケーションソフトを開発するためのフレームワークとしても大変便利である．ここでは我々が開発した動画データブラウザの典型的な利用方法を実況的に述べることでその機能を説明する．

N個の可視化視点を設定した3次元時間発展シミュレーションを考え，シミュレーションの時間発展に伴って計M回（M時刻ステップ）のその場可視化を実施したと想定する．各視点で行う可視化はL通りあるとする．例えば二つのスカラー場を等値面手法で可視化する時，等値面のレベルパラメータをそれぞれ5つずつ設定すればL＝2×5＝10である．一つの視点から（cube mappingで）6方向に対して可視化を行うので，一回のシミュレーションジョブでは合計6NLM個の可視化画像がVISMOから生成される．シミュレーション終了後，この6NLM個の画像ファイルをスーパーコンピューターのディスクシステムから手元のPCに転送する．そしてPC上でまずはそれぞれのカメラ位置と時刻での6枚の部分画像ファイルを融合して全方位画像にした上で，各視点ごとに時間発展のMフレーム分の全方位画像をまとめてMP4（H264/MPEG-4 AVC）コーデック[21]の動画ファイルにする．MP4は画質を維持しながらも高いデータ圧縮率を得ることができる優れた動画コーデックである．こうして合計NL個のMP4ファイルが得られる．それぞれのMP4ファイルにはMフレームの全方位画像が入っている．

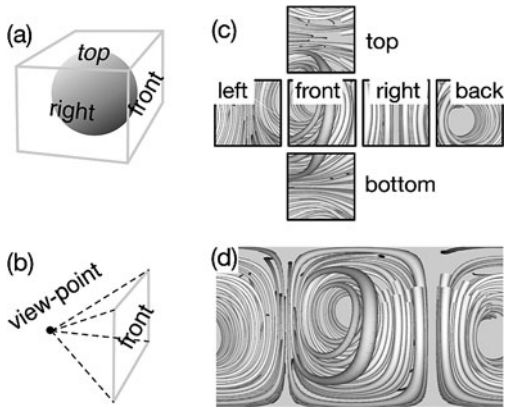

図3 VISMOを用いた全方位in-situ可視化．(a)各視点の周囲に仮想的な立方体を配置する．(b)立方体の各面に視野角90度の透視射影で可視化を行うと，(c)全部で6枚の可視化画像が生成される．(d)6枚の画像を1枚の全方位画像に合成する．

図4は我々が開発した動画データブラウザを立ち上げた直後の様子である．PCのデスクトップにウィンドウが一つ現れる．ウィンドウ内に表示されているのはデフォルトの視点位置，視線方向，時刻フレームにおける可視化画像である．マウスのカーソルをこのウィンドウ内におき，ドラッグすると視線の方向を自由に変えることができる．画面左上にはその時に読み込んでいる動画ファイルに関する情報がテキストで表示されている．画面右下には動画の再生・逆再生や特定の時刻フレームへのジャンプなど，時間方向の移動のためのボタン類が配置されている．

既に述べたように，視点の移動はマウスのクリックまたはキーボードで行う．ウィンドウ内にマウスカーソルを置き，マウスボタンをクリックすると，その方向にある近傍カメラに視点を移動することもできる．動画を（時間的に順方向または逆方向に）再生中であっても移動が可能である．カメラの距離が離れているときには突然ジャンプしたという感じが当然あるが，カメラ密度が十分に高いと滑らかに空間移動しながら動画を再生できる．これは映画「マトリックス」で有名になったbullet time法とよばれる映画の撮影技術と本質的に同じものである．

動画データブラウザには可視化手法を切り替えるためのユーザインタフェースも用意されている．もちろん「撮るフェーズ」においてあらかじめ撮影した可視化手法でなければ（そもそも動画データが存在しないので）切り替える

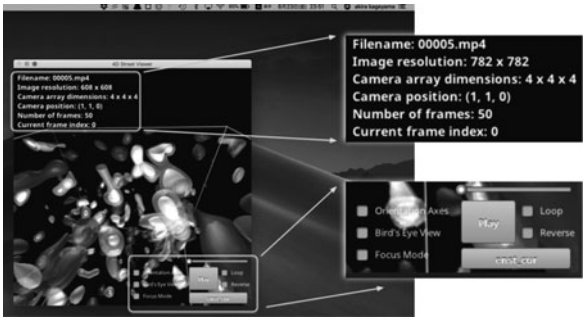

図4 動画データブラウザ．

ことができない．世の中に可視化手法は無数に存在するが，それぞれのシミュレーション研究者が必要とする可視化手法は実際にはそのほんの一部という場合がほとんどであろうから，無数の動画を用意しなければいけないというわけではない．

4．応用例

4次元ストリートビューを実際に活用してシミュレーションデータを解析する応用例をここでは二つ紹介する．別の応用例としてHall MHDシミュレーションへの適用があるが，それについては別の論文に書いた[22]．

4.1 渦輪のシミュレーション

最初に紹介するのは渦輪のシミュレーションである（**図5**）．直方体領域内の圧縮性流体（空気）の運動を考える．境界条件は3次元周期境界条件である．シミュレーション開始時には流れがないものとし，シミュレーション開始直後に箱の中の小さな領域内部に短時間，力の場をかけて流れを駆動する．力をかける領域は短い円筒状で，その円筒領域内部では一様に力をかける．その向きは図の$+x$方向である．するとこの円筒領域内部の空気がx方向に流れ始める．パルス状にかけるこの外力場はすぐに消えるが，流体が得た運動量は残る．はじめは様々な空間スケールの入り混じった複雑な流れが生じるが，しばらくすると自己組織化によりきれいな渦の輪が形成される．渦輪の中心軸はx軸に並行で，渦輪の穴の中では$+x$方向，渦輪の外側には$-x$方向の流れがある．渦輪全体はその形状を保ったまま$+x$方向に進んでいく．

我々はこの渦輪の形成と伝播を解くためのシミュレーションコードを開発した．空間方向には中心差分，時間方向にはルンゲ・クッタ法による時間積分をする基本的なコードである．並列化は領域分割によるMPI並列化とOpenMPとのハイブリッド並列化である．

このコードにVISMOベースの多視点・全方位・その場可視化機能を組み込んだ．可視化視点の数Nやその配置方法を何通りか変えて実験したが，その中から視点の数をN＝1000とした場合について以下に述べる．**図6**にその視点配置を示した．x，y，z軸方向にそれぞれ等間隔に20×10×5の配置でシミュレーション領域を取り囲むよう，その内外に視点を置いた．シミュレーションの差分格子点数は480×240×240，MPIプロセスの数（＝領域分割数）は432である．可視化手法の数はL＝2とした．一つはエンストロフィー密度の等値面，もう一つはそれに加えて流れ場を示す矢印（グリフ）を重畳して可視化したものである．一つの動画のフレーム数，つまり可視化した回数はM＝420である．上述の手順で生成した動画データベースにはN×L＝2000個のMP4動画ファイルがあり，そのサイズは合計6.0 GBであった．このサイズは画像の種類（圧縮のしやすさ）に依存する．今回はMP4によるデータ圧縮率は4.7%，シミュレーション全体の時間の中でその場可視化にかかる時間は65%であった．データ圧縮率は画像によって変わるがそれほど大きな変動はない．一方，その場可視化に必要な時間は可視化の設定によって大きく変わる．

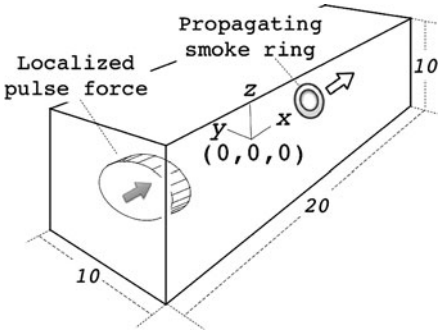

図5　渦輪の形成・伝播シミュレーション．20x10x10の長さの直方体領域中の空気の流れを解く．

視点の数や可視化の種類を増やすと可視化に当然，さらに時間がかかるが，その場合にはシミュレーションと可視化を完全に分離し，可視化には専用の計算ノードを割り当てるMultiple Program，Multiple Data方式の特殊なその場可視化手法を適用すればよい．我々はそのためにMembraneと名付けたフレームワーク[23]を既に開発済みである．シミュレーションを実行する計算ノードとその場可視化を実行する計算ノードをハードウェア的に分けて割り当てるのがこの手法である．シミュレーションプログラムとその場可視化プログラムがそれぞれ独立した並列プログラムとして別々の計算ノードで別々のMPIプログラムとして実行される（それぞれがMPI_INITをもつ）ので，その場可視化の実行によってシミュレーションの実行速度が遅くなることはない．ただし，可視化プログラム用に余分な計算ノードを割り当てる必要がある．

今回の渦輪のシミュレーションの場合，上述のように65%の時間をその場可視化に費やしたが，これは空間と時間を移動する際，動画データブラウザの画像表示が滑らかさであるためには少々過大であった．もっと少ない視点と可視化回数でも実用上は十分なめらかな解析が可能であった．適切な視点数と可視化回数を自動的に設定する手法の開発は今後の課題の一つと考えている．

視点位置を変更させない，つまり固定したカメラで，さらに視線方向も変更しないで見たときの様子を**図7**に示す．この時の視点は箱型のシミュレーション領域の内部，$+x$側の境界面近くにある．渦輪を駆動する力の場はその反対側（$-x$面の側）にある．**図7**(a)は渦輪の自己組織化がほぼ終わった頃のエンストロフィー等値面である．(b)，(c)，(d)と時間が進むにつれて渦輪がその形状を

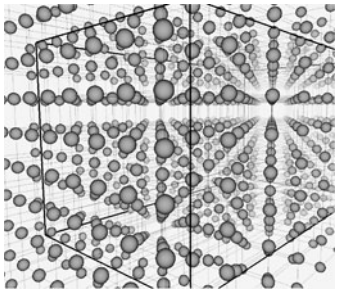

図6　渦輪シミュレーションの4次元ストリートビューのために配置した1000台の全方位カメラ．

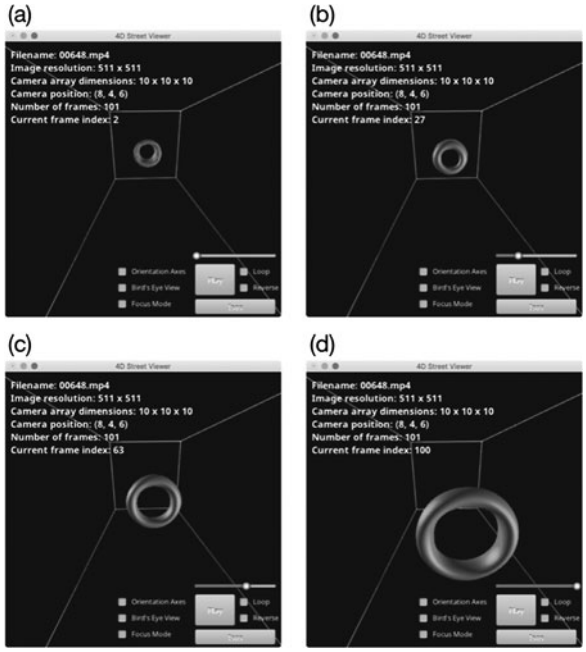

図7　固定した視点から見た渦輪の伝播．再生中にマウスドラッグで視線方向を自由に変えることができる．このままだと通常の（全方位動画）プレーヤと同じだが，再生中に視点の移動や可視化手法の切り替えもできるのが動画データブラウザの特徴である．

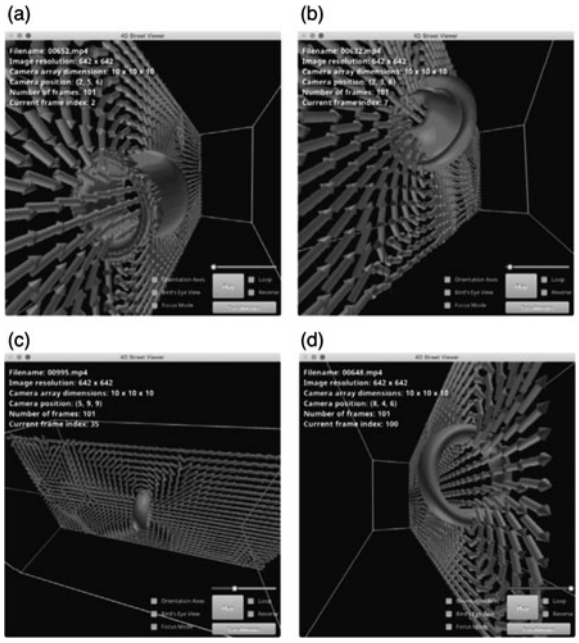

図8　渦輪の形成と伝播を移動しながら観察する様子．エンストロフィーの等値面と速度ベクトルの矢印グリフによる可視化．動画データブラウザを操作することで，渦輪を追いかけてつつ，各時刻で渦輪と周囲の流れ場を様々な角度から観察することができる．

保ったまま手前に近づいてくる．ここでは一つの動画ファイルの中身（画像フレーム）を次々と画面に表示しているだけであり，普通の動画プレイヤーと変わりがない．ただし通常の動画プレーヤーとは異なり，再生中にマウスをドラッグすることで注目する視線方向をリアルタイムで変更することができる．とはいえ最近では YouTube の全方位動画等でもそのような操作ができるので格別珍しいことではない．

4 次元ストリートビューの真骨頂は，動画の再生中に視線方向だけでなく視点の移動もできることである．

図 8 はエンストロフィー密度の等値面と流れのグリフ表示を重畳した可視化である．可視化手法（入力動画）の変更も瞬時にできるが，この図で強調したいのは，渦輪が x 方向に移動するのをユーザがリアルタイムで追いかけながら，その形状変化と周囲の流れ場の様子を対話的に観察している一連の様子を示しているという点である．**図 8** (a) は渦輪の自己形成の最中である．エンストロフィー密度が筒状に集中している．この筒はパルス状にかけた力の円筒領域の側面部分に相当する．筒の内部に位置する速度ベクトルのグリフが示すようにこの円筒領域の内部にも流れは存在しているが，その分布が空間的にほぼ一様であるため渦度は円筒の周辺部に集中している．その後，時間の経過とともに，渦輪はきれいなリング状に自己組織化していく．自己組織化の終盤での形状が(b)である．(c)の時刻では渦輪の形成は既に完了し，その形状を保ったまま x 方向に移動している．(d)では渦輪を追い抜き，先回りをして渦輪が近づいて見える位置からその伝播と周囲の流れ場（グリフ群）の時間変化を観察している．

図 8 では，(a)から(d)にかけて，渦輪を追いかけながら，そして見る方向を変更しながらカメラ位置や視線方向を対話的に変更した．ここでは，時間を順方向に進めているだけのように書いたが，時間進行を途中でとめたり，逆方向に再生したり，あるいは特定の時刻フレームにジャンプすることも可能である．

上述のようにこの動画のフレーム数 M は420である．今見ているのは比較的単純な現象であるのでこれは十分に高い時間解像度である．事後可視化手法によってこれだけの時間解像度もつ可視化をしようとする場合，保存すべきエンストロフィー密度（等値面用）と速度場（グリフ用）の単精度（4 バイト）の 3 次元数値データのサイズを計算すると，480×240×240（格子点）×420（時刻）× 4（変数，ベクトル 3 成分＋ 1 スカラー）×4 B（バイト）＝186 GB である．これは 4 次元ストリートビューの「撮るフェーズ」で作成した動画データベースのサイズ（6.0 GB）のサイズの30倍以上である．シミュレーションの規模が大きくなればこの差はますます拡大する．

図 9 に示すのは渦輪が形成される様子を動画データブラウザで時間方向に詳しく見ている時の様子である．当初筒状だったエンストロフィー等値面が複雑な流れ状態を経てリング状に自己組織化する過程を詳しく観察できる．順方向・逆方向の時間再生切り替えや一時刻フレーム毎の調整をしつつ，また同時にカメラ位置や視線方向を対話的に変更しながらこの渦輪の形成過程を観察しているうちに興味深い現象に気がついた．図中の小さな矢印（右斜め上向き）に示した位置に見える細くて小さいリング状の構造である．本来の（最終的に生き残る）渦輪はまだ形成途中で，この時刻では複雑な表面をもつ大きな曲面として見えているが，その曲面に隠れそうなほどか細いリング構造が短時間

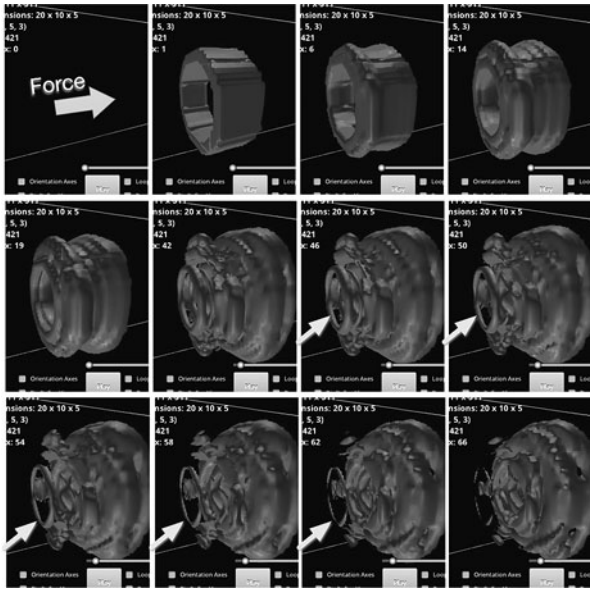

図 9　視点位置を対話的に移動させながら，渦輪形成の時間発展を詳細に見ているときに気がついた細くて短寿命の渦輪．

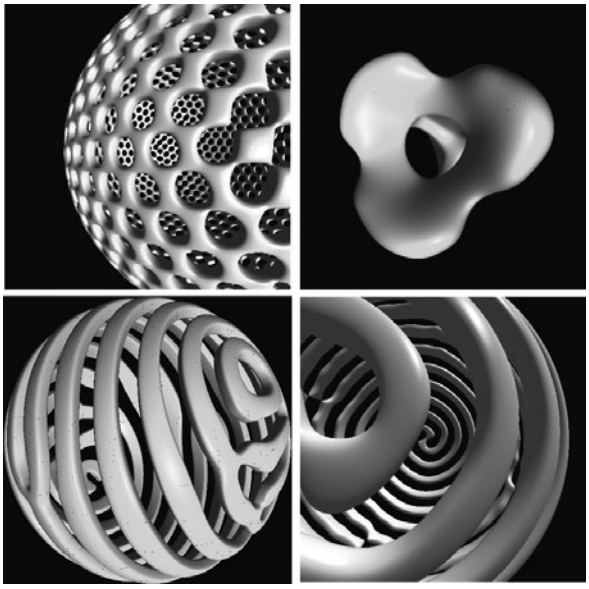

図10　半径比を変えた 4 種類の球殻中の熱対流シミュレーションのを VISMO ベースの 4 次元ストリートビュー手法で解析しているようすのスナップショット．

だけ生成されている．本来の渦輪の自己組織化過程に付随する副次的な現象で，流体力学的に重要な現象とも思えないが，ここで強調したいのは，このようにごく短い寿命をもつ構造は，時間方向に大きく間引きをしてしまう通常の事後可視化手法では見逃してしまうに違いないということである．また，見る角度によっては他の物体に隠されてしまうようなこの種の小さな構造は，通常の（視点と視野を固定した）その場可視化手法では見落とす可能性が高い．時刻だけでなくカメラ位置も対話的に変更することのできる 4 次元ストリートビュー手法がその威力を特に発揮するのはこのような現象の解析においてである．

4.2　球殻熱対流シミュレーション

汎用可視化ツールをシミュレーション研究者が使う際にまず最初に気にするのは自分のシミュレーションが用いている格子系，あるいは可視化したい物理量が定義された座標系にその可視化ツールが対応しているかどうかであろう．我々の場合はそれはインヤン格子[24]である．

インヤン格子とは，二つの合同な格子を相補的に組み合わせることで 2 次元球面または 3 次元球殻領域を解く格子系である．もともと地球ダイナモのシミュレーション[25,26]をするために我々が考案したものであるが，その後，マントル対流[27]，地球大気[28-31]，太陽ダイナモ[32]，超新星爆発[33,34]など，球ジオメトリを扱う様々なシミュレーションに広まった．日本の次世代気象予測モデルの基本格子系の一つとしても検討されている[35]．

VISMO はカーテシアン系（一様格子，rectilinear 格子）や非構造格子など，基本的な格子系には以前から対応していたが，上に述べたように様々な分野でインヤン格子が利用が広まっていることから，最近，インヤン格子にも対応できるよう大幅な改訂が加えられた．

図10にインヤン格子を用いて行った球殻熱対流シミュレーションの 4 次元ストリートビュー解析の様子を示す．このシミュレーションでは二つの同心球面に挟まれた領域内に閉じ込められた流体の運動を解いている．球の中心方向に向かう重力と境界の温度差（内側の球面は高温，外側の球面は低温に固定されている）によって熱対流が駆動される．これは地球外核など天体内部の対流のモデルとしてよく使われるモデルである．ただし，この計算では系の回転（自転）の効果は入れていない．ここに示した 4 つの図は球殻の厚さ（2 つの球面の半径の差）とレーリー数を変えたときの熱対流の様子を示したものである．右上の図は厚い球殻，それ以外の図は薄い球殻の結果を表している．対流パターンを可視化するためにここでは速度の半径方向成分の等値面を描いた．対流層の深さとレーリー数によって多面体状（上の二つの図），あるいはロール状（下の二つの図）という対照的な対流パターンが生成されることがわかる．

これらの図は 4 次元ストリートビュー手法で解析している途中，動画データブラウザの画面に表示された画像の途中のスナップショットである．実際の 4 次元ストリートビュー解析では，例えば球の中心に移動して周囲を見回してから，興味深い構造が出現し始めた部分に目標を定め，そこに向かって移動し，近くの視点からその構造が出現する前後の周囲の流れの様子を観察する．少し前の時刻に別の場所で発生した何らかの現象がその興味深い構造を誘起していることが推測できた場合は視点をその別の場所まで移動させて詳しく観察する，というのが典型的な 4 次元ストリートビュー解析の方法である．

インヤン格子対応版の VISMO と球殻シミュレーションへの応用については別の論文で報告する予定である．

5．まとめ

ほとんどの大規模シミュレーション研究者は現在，可視化用に出力する数値データを少なくとも時間方向には大幅

に間引きしてしまっているであろう．高度なシミュレーションを実行してせっかく得られた詳細な時間情報を大量に捨て去ってしまっているこの現状は，シミュレーション結果を数値データとして保存しておき，それをあとで可視化処理するという現在主流の可視化スタイル（事後可視化）が既に限界を超えていることを意味する．

大規模シミュレーションでは近い将来，シミュレーションをしながらその場で可視化を行うスタイル（その場可視化）が主流となるであろう．だが，その場可視化手法を単に適用するだけでは対話的な可視化解析ができない．

この問題を解決するために我々は4次元ストリートビューという新しい可視化のスタイルを提案した．シミュレーション空間に散布した多数の全方位カメラでその場可視化を行い，その結果を収めたデータベースを対話的に探索する，というのが4次元ストリートビューである．多数の視点を設定することで対話的な視点移動が可能になり，全方位可視化をすることで視線方向も自由に変更できるようになる．シミュレーションの時間発展を時間解像度の高い動画として保存するので，時間方向に自由に移動することもできる．

その場可視化ライブラリVISMOを使い，我々はスーパーコンピュータ上でこのような多視点・全方位・その場可視化を実現した．そして，生成された動画データベースを対話的に探索するPCアプリをKVSのフレームワークで開発した．このアプリ，動画データブラウザは，ユーザーが指定する視点位置，視線方向の変更にリアルタイムに応答し，該当する画像列を動画データベースから瞬時に取り出し，滑らかなアニメーションとして画面に表示することができる．

4次元ストリートビューは，スーパーコンピュータの計算資源を可視化のためにふんだんに，最終的にはシミュレーションそのものよりもずっと多く割り当てることを前提とする方法である．シミュレーションよりも可視化に計算資源を多く割り当てるというのは非常識に思えるかもしれない．しかし，現在のスーパーコンピュータシステムには膨大な数の計算コアが搭載されており，一つのシミュレーションジョブだけでそのすべてを使い尽くすのは既に至難の業である．今後，多くのシミュレーションアルゴリズムは成熟する一方，その規模だけはますます増大しつづけるであろう．そうなると可視化を中心としたデータ解析こそがシミュレーション研究の難しさとなるにちがいない．可視化は数値データをピクセル列に変換する一種の数値演算であり，しかもそれは本質的に大規模並列化に向いている処理である．有り余るほどあるスーパーコンピュータの計算コアを可視化に使うという考え方があってもよいのではないであろうか．

謝　辞

本研究はJSPS科研費16K12434，16K00173，17H02998，立石科学技術振興財団　研究助成(B)，I-O DATA財団研究助成，SCAT助成金の補助を受けたものです．

参考文献

［1］K.L. Ma *et al.,* J. Phys. Conf. Ser. **78**, 1-10 (2007).
［2］R.B. Ross *et al.,* J. Phys. Conf. Ser. **125**, 012099 (2008) doi: 10.1088/1742-6596/125/1/012099.
［3］K.L. Ma, IEEE Comput. Graphics Appl. **29**, 14 (2009).
［4］A. Tikhonova *et al.,* Proc. PacificVis 2010, 177 (2010).
［5］N. Sakamoto *et al.,* J. Adv. Simulation. Sci. Eng. **2**, 76 (2015).
［6］T. Kawamura *et al.,* Proc. 2nd Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV 2016) 18 (2017).
［7］J. Ahrens *et al.,* Proc. SC14: The International Conference for High Performance Computing, Networking, Storage and Analysis, 424 (2014).
［8］A. Kageyama and T. Yamada, Comput. Phys. Commun. **185**, 79 (2014).
［9］D. Anguelov *et al.,* Computer **43**, 32 (2010).
［10］R. Skupin *et al.,* Proc. IEEE Visual Communications and Image Processing (VCIP 2017) 1 (2017).
［11］U. Ayachit *et al.,* Proc. The First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV2015), 25 (2015).
［12］B. Whitlock *et al.,* Proc. Eurographics Symposium on Parallel Graphics and Visualization 2011, 101 (2011).
［13］Q. Liu *et al.,* Concurrency and Computation: Practice and Experience **26**, 1453 (2014).
［14］I. Wald *et al.,* ACM Trans. Graphics **33**, 8 (2014).
［15］I. Wald *et al.,* IEEE Trans. Visualization Comput. Graphics **23**, 931 (2017).
［16］N. Ohno and H. Ohtani, Plasma Fusion Res. **9**, 3401071 (2014).
［17］H. Miura, Fluids, **4**, 46 (2019).
［18］H. Miura *et al.,* Phys. Rev. E **100**, 063207 (2019).
［19］G. Sellers *et al., OpenGL Super Bible*, 7th edition (Addison-Wesley, 2015).
［20］N. Sakamoto and K. Koyamada, J. Adv. Simulation. Sci. Eng. **2**, 76 (2015).
［21］A. Puri *et al.,* Signal Process. Image Commun. **19**, 793 (2004).
［22］A. Kageyama *et al., submitted to* Plasma Fusion Res..
［23］A. Kageyama *et al.,* Proc. 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTEC) 106-111 (2018).
［24］A. Kageyama and T. Sato, Geochem. Geophys. Geosyst. **5**, Q09005 (2004).
［25］A. Kageyama *et al.,* Nature **454**, 1106 (2008).
［26］T. Miyagoshi *et al.,* Nature **463**, 793 (2010).
［27］P.J. Tackely, Phys. Earth Planet. Inter. **171**, 7 (2008).
［28］Y. Baba *et al.,* Mon. Wea. Rev. **138**, 3988 (2010).
［29］A. Qaddouri, Q.J. R. Meteorolog. Soc. **137**, 810 (2011).
［30］K. Saito *et al.,* J. Meteorolog. Soc. Jpn. **85B**, 271 (2007).
［31］Xingliang Li, Adv. Atmos. Sci. **30**, 1320 (2013).
［32］H. Hotta *et al.,* Science **351**, 1427 (2016).
［33］E. Müller, *et al.,* Astron. Astrophys. **537**, A63 (2012).
［34］J.M. Blondin and E. Raymer, Astrophys. J. **752**, 30 (2012).
［35］気象庁予報部：数値予報課報告 別冊第60号，1-151 (2014).