

# 点群によるシミュレーション結果の可視化：ポイント・ビジュアリゼーションの提案

## Point Visualization: A Proposal of Scientific Visualization by point clouds

宮地英生<sup>1</sup>

### 1 はじめに

コンピュータシミュレーションや実験・観測結果などの科学的なデータをコンピュータグラフィックス(以下 CG)で画像化する技術・そのプロセスは、サイエンティフィック・ビジュアリゼーション(以下、可視化)と呼ばれる(図1)。この分野は1987年に米国計算機学会コンピュータグラフィックス部会(ACM SIGGRAPH)が出版した Visualization in Science Computing(ViSC)レポートにおいて「サイエンティフィック・ビジュアリゼーションの活用は、その国の科学技術の進展、さらには国家の隆盛にも大きな影響を与える」と提言されたことで研究者の注目を集めることになった<sup>[1]</sup>。この時期、CGの能力を標準装備した高速な計算機“ワークステーション”が台頭し、その広がりとともに可視化も普及していった。

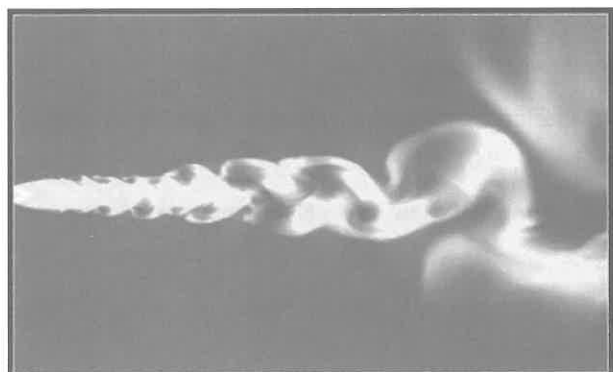


図1 数値流体シミュレーション結果の可視化事例

(左から出る噴流が右へカルマン渦を形成しながら拡散していく様子を可視化している)

<sup>1</sup> 東京都市大学メディア情報学部教授

続いて、90年代の半ばから Windows OS を搭載したパーソナルコンピュータ(以下、PC)が普及し、文字通り一人1台 PC を利用することがオフィス業務では普通になった。しかし、これにより群雄割拠していたワークステーションは絶滅へ向かうこととなり、3次元の汎用グラフィックスライブラリの規格 PHIGS、および、その後継も廃れていくことになった。その結果、規格に消極的だったシリコングラフィックス社が提供する GL をオープン化した OpenGL が3次元グラフィックスの業界標準となった。1995年、3DLabs 社が OpenGL 専用のグラフィックスチップ(アクセラレータ)の開発に成功し、3次元 CG は誰もが利用できる時代となる。

一方、CG ソフトも急速な発展を遂げ、さまざまな分野へ分化していった。また、コンピュータシミュレーションは計算機の高速化により、複雑な形状を扱うことが可能となり、CAD で設計されたデータを基に計算格子を生成、そして、シミュレーション、可視化、分析・評価という一連の流れが確立される。前述したように CG は標準化され誰でも簡単にコーディングできるようになり、これら一連の作業を包含した統合システムが次々と開発されていった。

可視化の技術開発も汎用的な技術は90年代前半にひと段落し、後半からは分化していった<sup>[2]</sup>。その分化した流れの1つが、本稿で扱う大規模可視化である。これは特に宇宙、機械、分子のような適用分野を絞ったものでなく、計算機の高性能化に付随して大規模化するデータを効率的に可視化することを課題とする。ハードウェアの進展と深い関係にあり、その時代に提供されるシステムにおける可視化処理の最適化問題とも考えられる<sup>[3]</sup>。

ここでは、現在のシミュレーション環境に適した可視化システムを、可視化の原点に立ち返って考える。第2章で可視化の処理を概説し、第3章で大規模可視化の対応策を述べる。それを踏まえ、第4章で新しい概念「ポイント・ビジュアリゼーション」を提案する。

### 2 科学技術データの可視化

冒頭で述べたように科学技術データの可視化は、コンピュータシミュレーションの出力データだけで

なく、実験・計測で得られるデータも対象である。また、シミュレーションの種類によってデータの構造、可視化処理の流れに多少の違いが生じるが、ここでは著者の専門分野である差分法の数値流体解析に焦点を当てて記述する。

## 2.1 可視化のパイプライン

可視化は、次の4ステップのパイプライン処理である(図2) [4]。

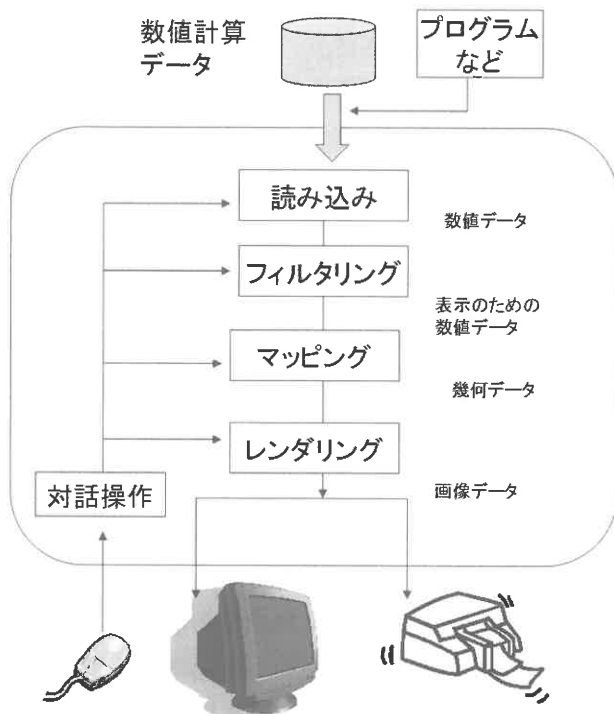


図2 可視化処理の流れ

### (1) データの読み込み

ディスク上に記録されたシミュレーション結果をメモリ上に正しく読み込む処理。

### (2) フィルタリング

シミュレーション結果全体から可視化の対象となるデータを抽出する処理。例えば、圧力  $P$  と速度の3成分  $U, V, W$  の4変数から、断面の速度分布を見るため、ある断面の  $U, V, W$  を抽出する処理(図3)。または、可視化のために必要なデータを生成する処理。例えば、3方向の流速  $U, V, W$  から速度の絶対値を計算するような処理。

### (3) マッピング

可視化の対象となるデータをレンダリングの対象となる幾何データと色データに変換する処理。

例えば、速度ベクトル図の場合、速度3成分  $U, V, W$  を色のついた矢印に変換する処理(図3)。

### (4) レンダリング

色のついた幾何データを2次元の画像に変換する処理。これに対話処理が加わり、可視化システムが構成される。

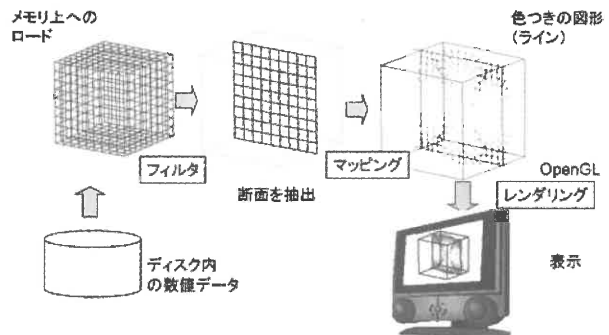


図3 可視化パイプラインの具体例

## 2.2 サイエнтиフィックビジュアライゼーションの目的

可視化の目的はシミュレーション結果の妥当性の判断と定性的な結果の把握にある。結果の定量的な分析も重要な目的だが、ここでは人間が直接的に画像を見て判断する範囲を対象とする。

また、可視化は研究者自身が観察者の場合と、伝達目的に他者に見せる場合に分類されると言われる。前者を目的とするシステムは対話処理を重視し、後者では、映画作成のように時間がかかっても解り易く、リアリティのある画像や動画を作成する。本提案は、前者を目的とした対話性能を重視するシステムを対象とする。

## 3 大規模データ可視化の課題と解決策

### 3.1 データの大規模化による可視化の課題

1980年代の後半、著者は「計算機性能が向上し、シミュレーションの出力データが大量になり、従来のグラフや表では結果が評価できなくなった。」のような記述ではじめ、それ故、画像による視覚化が必要と記述していた。当時は真面目に書いていたのだが、その10年後には、このデータの大規模化が可視化の課題になる。それは計算機の各構成要素の進化速度が異なることに起因する。

### 3.1.1 スーパーコンピュータの性能向上

スーパーコンピュータの性能は 1986 年米国クレイ社の Cray-2 が 1GFlops を超え、30 年後の 2016 年の TOP500 ラインキングの 1 位（中国国立スーパーコンピュータセンター、システム名称:Sunway TaihuLight）は 125PFlops のベンチマーク性能を記録している<sup>[5]</sup>。

このようにコンピュータの演算性能は、30 年間で 1,000,000 倍性能が向上している。しかし、可視化処理のフィルタリングやマッピングでは演算能力が重要だが、読込時はデータ IO、レンダリング時はグラフィックスの能力が重要となる。

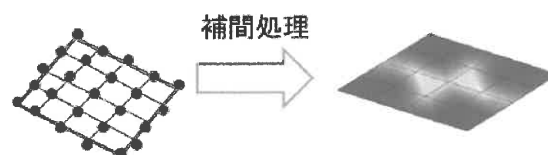
### 3.1.2 ディスクの読取り速度

ディスクの読み取り速度は、1986 年に ANSI で制定された SCSI-1 では 40Mbit/sec に対して、2011 年に発表された Serial ATA Express の規格上の最大転送速度で 16Gbit/sec である。30 年間で 400 倍程度しか性能が向上していない。このため現在、データの読取りが可視化処理の最大のボトルネックになっている。

### 3.1.3 ディスプレイの解像度

ディスプレイの画素数は、1986 年頃広く使われた SXGA(縦 1280 ピクセル、横 1024 ピクセル)で 131 万ピクセル、8K モニター（縦 8192 ピクセル、横 4320 ピクセル）で約 3500 万ピクセルと 30 倍程度に過ぎない。ディスプレイをタイル状に並べて高解像度を実現するシステムもある<sup>[6]</sup>。それでも、最大規模の米国 Texas advanced computing center のシステム STALLION は 328 メガピクセルで SXGA の 300 倍弱に過ぎない<sup>[7]</sup>。一方、数値流体シミュレーションの最大規模は 100 億格子点を超える。単純に割り算しても 1 画素あたり 300 セル程度が含まれる計算になる。1990 年代はシミュレーション結果を可視化するとき計算セルの内部を補間処理で塗りつぶしていたが、このような状態で三角形をレンダリングする意味がなくなっている（図 4）。

#### 1990年代の状況



#### 現在の状況

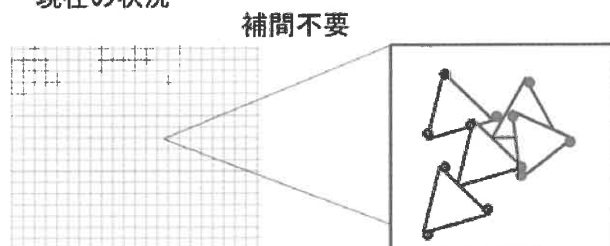


図4 計算セルと画素の大きさの関係

しかし、画素数を増やせば良い可視化ができるとは限らない。ディスプレイを観察するのが人間であるから、人間の認識との関係を考慮する必要がある。データを単純に画像に変換するだけでなく、理解・分析のためのコンピュータ支援が望まれている。

### 3.1.4 コンピュータ間の性能差

コンピュータの運用にも課題はある。スーパーコンピュータの性能は飛躍的に向上したが、近年の高速化は並列化によって成し遂げられている。先述の 2016 年 11 月の世界最速のコンピュータは 1000 万コアを超える並列処理で世界最高速を達成している。これは大規模な数値計算に適した仕様で、さほど演算性能を要しない可視化処理にはオーバースペックである。特に対話処理を想定した場合、人間の操作はスーパーコンピュータにとってあまりに遅く、費用対効果が悪すぎる。したがって、超並列計算機を対話処理で運用することはない。現在のコモディティな PC は 30 年前の Cray-2 の演算性能を上回っているが、現在のスーパーコンピュータが扱う規模のデータを処理するとなると、メモリ、演算性能ともに不足することになる。

## 3.2 大規模データ可視化の解決策

### 3.2.1 データ転送について

スーパーコンピュータで計算し、デスクトップの

パソコンで可視化を行う場合、コンピュータ間の性能差が問題となる。しかし、それ以前に問題となるのがデータの移動である。2017年現在、大規模な数値流体計算は格子点で100億程度(10Gポイント)ある。1点あたりXYZの座標値とPUVWの4変数があり、これらを単精度の浮動小数点(4バイト)で出力したとしても1点28バイト、1時刻分の出力データ量は280Gバイトになる。これは1Gバイト/秒(8Gbit/sec)で転送できたとして280秒を要する。これに時系列の計算では出力回数分を掛けたデータ量になる。このような大量のデータ転送は一般組織では他人の迷惑になるので、事実上、データ転送はできない。そこで、3つの対策が考えられている。

### (1) リモート可視化

データが転送できないので、転送せずに元々在る場所で可視化を行うという構想。対話操作に遅れが生じることが問題だが、クラウドシステムの普及によりリモート端末環境も良くなっている。

### (2) In-SITU 可視化

転送もできないデータは書き出す必要もなく、シミュレーションと同時に可視化をしてしまう構想。シミュレーションプログラムの中に可視化機能を組み込み直接画像出力する<sup>[8-10]</sup>。基本的にバッチ処理となり、対話的な操作ができない。再計算が可能であれば、何度でも処理を流せば新しい可視化結果は容易に得ることができる。また、予め多くの可視化を行ってしまい、そこから選択的に見る方法も検討されている<sup>[11]</sup>。

### (3) データリダクション

データが大きすぎるので、小さな代替となるデータを生成する構想。可視化処理の第2ステップであるフィルタリングを事前に行っているとも解釈できる。可視化したい変数の値の抽出、可視化したい場所の部分抽出、間引きした粗いデータの生成など、シミュレーションプログラムの出力側に少しコードを足すだけなので、自作のコードであれば容易に実装できる。

## 3.2.2 レンダリング性能

グラフィックスアクセラレータは安価で高性能

になり、小規模なデータを可視化している限りレンダリング性能が問題になることは無い。問題はグラフィックスメモリの容量が主記憶に対して圧倒的に少ないことである。対話的な可視化ではリアルタイムに実現できるグローシェーディングを用いるので物体相互の計算は無く完全な並列処理が可能である。1枚のグラフィックスボード上では共有のグラフィックスメモリに全ての物体データが入っていれば高速に計算できるが、その入れ替えが発生すると急激に遅くなる。複数のグラフィックスボードを用いて並列でレンダリングを行う場合も、グラフィックスボード間でのデータ交換が速度のボトルネックになる。データ交換を無くすには全データを全てのグラフィックスメモリに格納すれば良いが、グラフィックスメモリの不足が理由で並列化が必要な場合、それは不可能である。

そのため、大規模データの可視化において並列でレンダリングする際は、物体を分割してレンダリングを行い、最後に画像を重ね合わせる方法が主流となっている(図5)<sup>[12,13]</sup>。

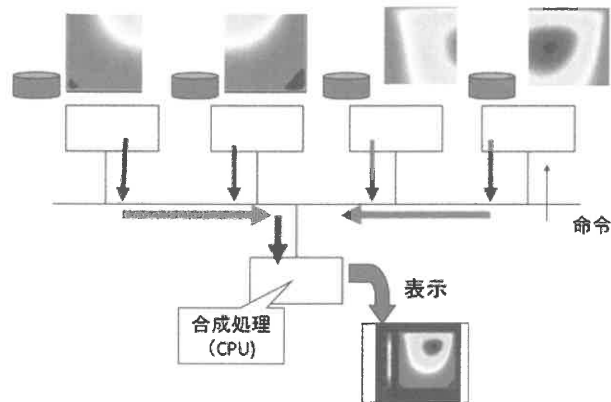


図5 ソートラスト（画像重畳）の並列レンダリング

### 3.2.3 対話性能

可視化処理はシミュレーション結果を1枚の画像に変換するだけでなく、人間を含めた処理ループの中での対話処理が大切とされる。その対話処理の中で大量のデータから有益な情報を抽出する手法を、データマイニングの一種としてビジュアルデータマイニングと呼ぶ<sup>[14]</sup>。ここでは、対話操作に基づく再計算を可視化プロセス上流のどこまで戻るのが性能の鍵になる。それに応じて再計算を必要最低限にするには、可視化プロセスの各ステップでデータを保

持することが望まれるが、それには相応のメモリ領域が必要となる。

この問題を解決するために粒子または点（ポイント）を用いたレンダリングが考案されている。本稿の提案は、その思想の延長にあると考えられる。

## 4 点群による可視化の提案

### 4.1 関連研究

#### 4.1.1 ポイントグラフィックス

差分法の3次元シミュレーション結果は六面体のボクセルで定義される。これを直接ボリュームレンダリングで画像に変換する方法と、可視化のマッピング処理で一旦ポリゴンデータに変換した後にレンダリング処理を経て画像に変換される場合がある。いずれの場合も最終的に表示される画素の大きさよりもポリゴンやボクセルが小さくなる場合、レンダリング処理の前にデータを点に変換することが有効である。[15]

そこで物体を点の集合として扱うポイントベースのレンダリングパイプラインのアイデアが提案された。その代表的な手法にスプラッティングがある[16]。これはカーネルと呼ばれる関数分布を持った点をスクリーンに投影することで画像を生成する。その他、点の隙間を無くす技術、高速化に関する技術が研究されてきた。

#### 4.1.2 粒子ベースレンダリング

最終的な表示解像度に着目し、ボリュームデータを、それに応じた密度の粒子に変換する粒子ベースのボリュームレンダリング(PBVR: Particle Based Volume Rendering)[17]、これに類似した考え方でサーフェイスデータを粒子に変換する粒子ベースサーフェイスレンダリングがある[18]。これらは、マルチパスのレンダリングで、レンダリング毎に考慮する粒子を確率的選択することで並列化が困難な物体の前後ソーティング処理を不要とし、高品質な半透明レンダリングを並列処理で高速化することを特徴とする。これらの手法は対話的な可視化に有効で応用事例も報告されており、著者もこれらの手法を用いた可視化システムを開発してきた。[19]

### 4.1.3 点群に表示によるデータリダクション

データの軽量化のためにレンダリング処理を用いてサーフェイスモデルから軽量化された点群を生成する方法を著者は提案してきた(図6)[20]。

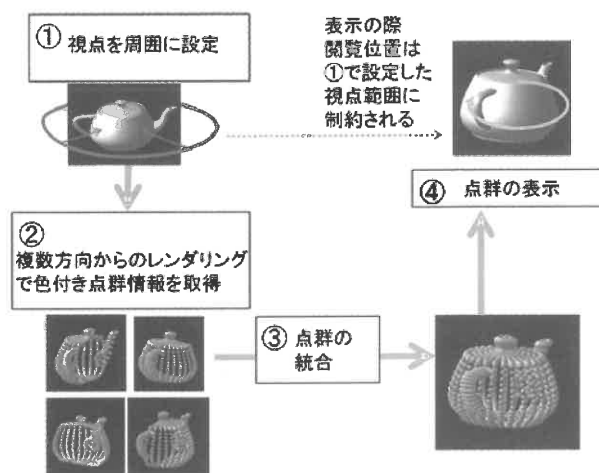


図6 レンダリング結果からの色付き点群の生成

1 方向からZバッファ法でレンダリングした結果はフレームバッファに色情報が、Zバッファに奥行情報が格納される。これらのデータは描画された画素数の2.5次元の色付き点群データと考えられるので、複数方向からの点群を統合することで3次元の外観を示すデータが再構築できる。この方法はオリジナルの3次元物体を完全に再構築できないが、物体中心を周囲から一定距離で観察する限り、単純な凸形状の物体ではサーフェイスでレンダリングの結果画像とほぼ同じ画像を得ることができる。この手法は、ポイント・ビジュアライゼーションにおいて回転操作の負荷軽減のための代替オブジェクトとして利用する。

## 4.2 ポイント・ビジュアライゼーション

ポイント・ビジュアライゼーションの提案は、画素サイズよりも密なデータは点で扱えば簡単になるという点において、関連研究と同じ思想に立脚する。しかし、本提案は、レンダリングだけでなく、可視化処理全般に点群の適用を拡張することで、従来から行われてきた可視化処理のパイプラインを変更することにある。提案する可視化パイプラインを図7に示す。

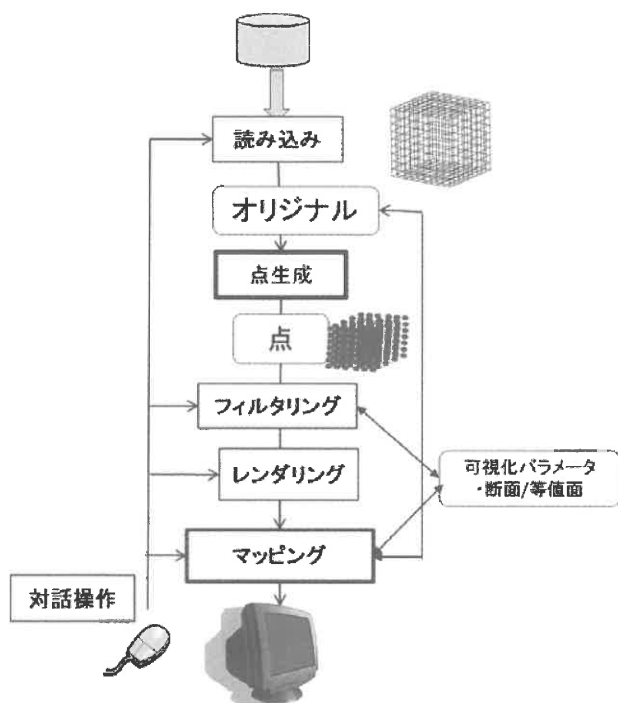


図7 ポイント・ビジュアライゼーションのパイプライン

図7と図2を比べると2つの点が大きく違っている。1つはデータの読み込み直後に点生成のプロセスが入っていること、もう1つはマッピングとレンダリングの順序が入れ替わっていることである。

#### 4.2.1 データの読み込み

本提案では、シミュレーション結果のデータは可視化の最初のステップであるデータのロード時に点情報に変換される(図8)。このとき、各点はオリジナルのセルへのリンク情報(インデックス)を保持する。

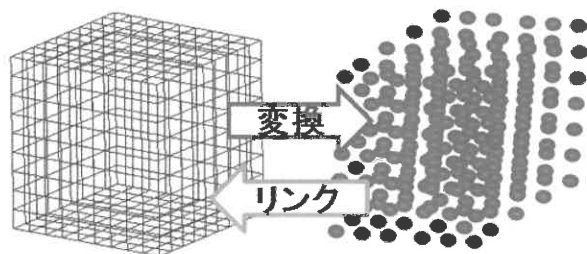


図8 ボリュームデータから点群への変換

各点はセルを代表し、その座標値  $X, Y, Z$  は、例えば、セルの重心とする。圧力のようなスカラー量は平均、最大、最小の3つの値に変換する。速度は速度の絶対値、各成分の平均値とする。隣接点との計

算を必要とする渦度などは、この段階で計算をしておく必要がある。

流線は、点群データでは扱えないので、必要になった場合はオリジナルのデータに戻して計算する。この処理は図7では割愛している。

#### 4.2.2 フィルタリング

ポイント・ビジュアライゼーションのフィルタリングは、可視化手法とパラメータと深い関係があり、従来のマッピングに近い処理となる。

##### (1) 断面分布図(コンター図)

スカラー値の断面分布図を表示するには、少し幅をもった断面に含まれる点群を抽出する。併せて、コンター図の最大最小値の間の値を持つ点のみを抽出する(図9)。

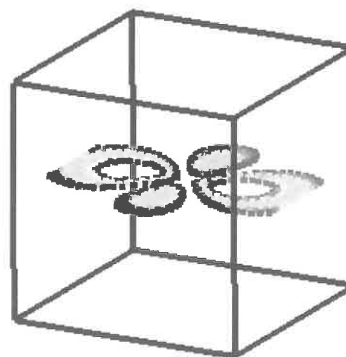


図9 点群による断面コンター図

##### (2) 等値面

等値面は、コンター図と同様の処理で描画したい等値面の値に幅を持たせて、その範囲内の値を持つ点のみを抽出する(図10)。等値面として一般に利用されるマーチンキューブ法の欠点は、生成した等値面のポリゴン数が増えることにある。しかし、点群による等値面は抽出処理だけなのでデータ量が増えることは無い。

##### (3) 速度ベクトル図

画素にセルが含まれるくらい密な状態であるから、隙間が必要な速度ベクトル図は表示できない。一定の空間を代表させるフィルタリング処理が必要である。

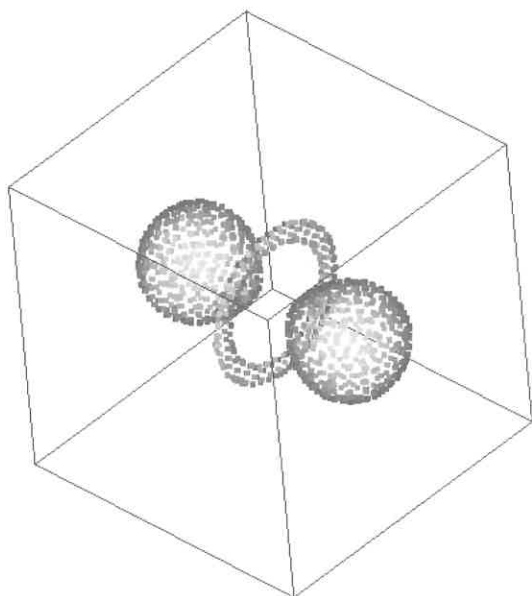


図 10 点群による等値面

#### 4.2.3 レンダリング

ポイント・ビジュアライゼーションの最大の利点は、このレンダリングのプロセスでラスタライジングを行わないことである。ポイントベースのレンダリングでは、隙間を無くすことが求められたが、ポイント・ビジュアライゼーションでは隙間には拘らない。もし、隙間によって結果を観察するための思考が妨げられるのであれば、単純に周囲の数ピクセルを同じ色で便宜的に埋めることとする。

このレンダリング時にフレームバッファと同じサイズの付加的なバッファに、Z バッファ処理と同じようにオリジナルのセルへのリンク情報を残す。これにより、フレームバッファに残ったセル、つまり、最も表面にあるセルのオリジナルデータへのリンク情報を獲得することができる。フレームバッファ上に残る画素数は、解像度に依存したもので、通常のディスプレイでは 200 万ポイント以下になるだろう。

この処理は、代替となる点群情報を用いて視点から見えるセルだけを Z バッファ法を用いて抽出したことになる。

#### 4.2.4 マッピング

最後にリンク情報を使って、フレームバッファに残ったセルのオリジナル情報へアクセスして、その画素の色を決定する。点の持つ色は、変数値と色の対応を決める伝達関数（カラーマップ）で与えられ

る。画素値は、可視化手法に応じた法線を計算してライティングを施して決定する。この処理はレンダリングとして実装することもできるが、画素値を変更するだけなので 2 次元の画像処理として実装することもできる。色の決定には法線情報が必要となる。

##### (1) 断面分布図（コンター図）

平面の分布図を表示する場合、面の法線は可視化のパラメータとして与えられる。

差分法では IJK の格子断面の分布図を表示する場合がある。この場合、一旦、オリジナルのセル情報へアクセスして、断面の法線を計算しなければならない。

##### (2) 等値面

等値面の各面の向きはオリジナルのセル情報へアクセスし、対象となる値の勾配を計算して法線とする。これらの法線を用いてライティング計算を行う。4.2.3 のレンダリング処理で物体の視点変換を行っているので、本来、その幾何変換行列を獲得しなければ、3 次元空間に設定したライトでと物体の位置関係は解らない。しかし、シミュレーション結果の可視化においてライティングは重要でない。ここでは、等値面の形状の把握、複数の断面を表示したときの面の区別のため便宜的にライティングを行う。したがって、ライトは 3 次元の世界座標内に設定する必要がなく、常にカメラの斜め上後方から中心に向かう平行光源で固定する。

#### 4.2.5 対話処理

従来の可視化パイプラインと順序が異なるので、対話処理のフィードバックする場所が少し異なる。可視化のパラメータを変更すると、多くの場合フィルタリング処理へ戻る必要がある。しかし、投影変換を伴わない処理については、最後のマッピングだけの処理で画像を再生成できる。

投影変換を要する拡大、回転などの操作は再レンダリングが必要で、その後、大きなオリジナルデータへのアクセスが発生してしまう。実装上は、法線の再計算が必要ない場合にはオリジナルデータの再アクセスが発生しないようにするべきである。

## 5 実装

本提案システムは、まだ計画段階だが、2つの実装を検討している。

1つはレンダラを含めすべての処理をグラフィックスボードに依存しない実装である。シミュレーション結果の可視化において、映像作成に重要なライティングの処理が簡易的なもので良く、投影法も平行投影だけで十分である。また、大規模データの可視化を対象とする場合、グラフィックスボードに依存しない方が良いことから、ソフトウェアで専用の簡単なレンダラをコーディング中である。

もう1つは既存の可視化ソフトウェアに本処理を統合する方式である(図11)。各種データ書式の読み込み、および、ユーザインターフェイスは既存システムに任せ、ポイント・ビジュアライゼーションはサブシステムとして動作させる。シミュレーション結果を読み取った後、点生成からポイント・ビジュアライゼーションの処理に入る。可視化のパラメータ、CGに関わる視点位置などのパラメータは既存の可視化システムと共有する。法線を計算して簡易的にライティングを施した色付きの点群データが既存システムに戻される。ここで速度ベクトル図、凡例、タイトルなど、ポイント・ビジュアライゼーションで扱いが難しい可視化と統合しディスプレイに表示する。

回転などの幾何変換操作は2種類のフィードバックを考えている(図11の幾何変換(1),(2))。ポイント・ビジュアライゼーションでは、先にデータの代替となる点群をレンダリングすることで、可視セルのみに可視化処理を実施することが要点である。これを忠実に実装するならば、ポイント・ビジュアライゼーションサブシステムにフィードバックしない幾何変換(2)の操作は、縮小(及び、縮小後、元へ戻すまでの拡大)、平行移動が幾何変換(2)に相当し、それ以外の拡大・回転処理は起色付き点群の再生成が必要な幾何変換(1)となる。しかし、回転時に毎回フィードバックが生じると対話操作が遅くなるため、回転操作中だけ負荷の軽い代替オブジェクトを表示する方法を適用する。そのオブジェクトとして、4.1.3で述べたデータリダクションにより生成される3次元点群を用いる。

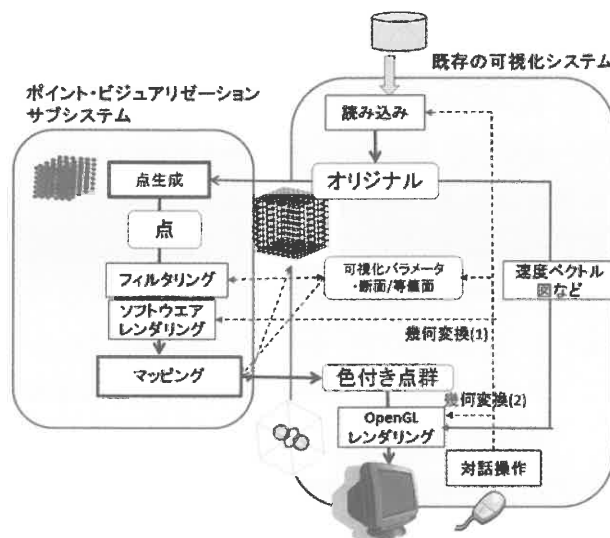


図11 既存可視化システムへのポイント・ビジュアライゼーションの統合

## 5 おわりに

シミュレーション結果のデータ量が大きくなってきたため、従来の可視化手順では効率が悪くなってきた。可視化結果がディスプレイに表示されるとき、1つの画素の中にたくさんのセルが含まれる場合、ポリゴンを生成するよりも点で表示の方が有利であるから、従来の可視化の処理順序を変更し、まず、最終的に表示されるセルを代替となる点群をレンダリングすることで抽出し、該当するセルのみに従来の可視化処理を施すシステムの設計について提案した。

現在、ソフトウェアによる点群レンダリングを実装中であり、これが完了すれば本提案の速度ベンチマークが可能となる。2017年度中にはプロトタイプを完成する予定である。

## 参考文献

- [1] 中嶋正之, 藤代一成. インターネット時代の数学シリーズ 4 コンピュータビジュアライゼーション, 共立出版, 2000
- [2] 宮地英生, 吉川慈人. 汎用可視化アプリケーション AVS に見る 1990 年代の可視化技術について, 可視化情報 Vol.20, No.78, 2000, pp.209-214



- [3] 寺坂晴夫, 清水泉介, 竹島由里子. 可視化のための大規模数値データの圧縮と復元, 可視化情報学会論文集, Vol.23, No.6, 2003, pp.52-57
- [4] 宮地英生, 荒木, 鈴木. 計算力学レクチャーコース 可視化入門, ISBN 9784621085837, 丸善出版, 2013
- [5] TOP500 Project ホームページ, <https://www.top500.org/> (2017 年 1 月 12 日閲覧)
- [6] L. Renambot, A. Rao, R. Singh, B. Jeong, N. Krishnaprasad, V. Vishwanath, V. Chandrasekhar, N. Schwarz, A. Spale, C. Zhang. Sage: the scalable adaptive graphics environment, Proceedings of WACE, vol. 9, 2004, pp. 2004-09.
- [7] Byungil Jeong, Jason Leigh, Andrew Johnson, Luc Renambot, Maxine D. Brawn, Ratko Jagodic, Sungwon Nam and Hyejung Hur. Ultrascale Collaborative Visualization Using a Display-Rich Global Cyberinfrastructure, IEEE Computer Graphics and Applications, vol. 30, no. 3, 2010, pp. 71-83
- [8] Kwan-Liu Ma. In Situ Visualization at Extreme Scale: Challenges and Opportunities, IEEE Computer Graphics and Applications, Vol.29, Issue6, Nov.-Dec. 2009, pp.14-19
- [9] 村松一宏, 土肥俊, 松本秀樹, 武井利文, 相川裕史. 並列計算機上での流体解析のための実時間可視化システム, 計算工学講演論文集, Vol.2, 1997, pp.109-112
- [10] 白山晋, 太田高志. クラスライブラリによる並列化実時間可視化システムの構築, *Transactions of JSCES* 1999, 1999, Paper No.19990002
- [11] Akira Kageyama and Tomoki Yamada. An Approach to Exascale Visualization: Interactive Viewing of In-Situ Visualization, Computer Physics Communications, vol.185, 2014, pp.79-85
- [12] S. Muraki, E.B. Lum, K.L. Ma, M. Ogata, and X. Liu. A PC cluster system for simultaneous interactive volumetric modeling and visualization. In Proc. Parallel Visualization and Graphics '03, 2003, pp.95-102
- [13] J. Nonaka, N. Kukimoto, N. Sakamoto, H. Hazama, Y. Watashiba, X. Liu, M. Ogata, M. Kanazawa, and K. Koyamada. Hybrid hardwareaccelerated image composition for sort-last parallel rendering on graphics clusters with commodity image compositor. In Proc. Symposium on Volume Visualization and Graphics '04, 2004
- [14] D. A. Keim. Information Visualization and Visual Data Mining, IEEE Transactions on Visualization and Computer Graphics, Vol.8, No.1, 2002, pp.1-8
- [15] 藤本忠博, 今野晃市, 千葉則茂. ポイントグラフィックス概説, 芸術科学論文誌, Vol.28, No.2, 2004, pp.8-21
- [16] L. Westover. Footprint Evaluation for Volume Rendering, SIGGRAPH 90, 1990, pp.367-376
- [17] 坂本尚久, 小山田耕二. 粒子ベースボリュームレンダリング, 可視化情報学会論文誌, Vol.27, No.2, 2007, pp.7-14
- [18] Kyoko Hasegawa, Yuta Fujimoto, Rui Xu, Tomoko Tateyama, Yen-Wei Chen, Satoshi Tanaka. Fused Visualization with Non-Uniform Transparent Surface for Volumetric Data using Stochastic Point-based Rendering, The 4th International Conference on Innovation in Medicine and Healthcare (InMed 2016), Tenerife, Spain, June 15-17 (June 17), 2016.
- [19] Hideo Miyachi, Tomoyuki Ishida, Naohisa Sakamoto, Koji Koyamada. Fusion Visualization for Fluid Dynamics in Blood Vessel, Journal of Artificial Life and Robotics, Vol.19, No.3, 2014, pp.286-290
- [20] 宮地英生. 観察者の視点領域を考慮した3次元点群によるデータ軽量化, 可視化情報学会論文集, Vol.36, No.8, 2016, pp.40-45