VRMLを用いたCFD結果の可視化技術*

小山田 耕 二^{*1}

A Visualization Technique of CFD Results Using VRML

Koji KOYAMADA^{*2}^{*2} Faculty of Software and Information Science, Iwate Prefectural University,
152-52, Sugo, Takizawa-village, Iwate, 020-0193 Japan

In this paper, we propose a new technique for visualizing volume datasets using VRML. In general, a visualization algorithm generates tens thousands of geometric triangles with numerical data, when it is visualized. They are huge in number, if we transfer them through the Internet. To solve the problem, we restrict the visualization on a slice plane and represent triangle mesh as a single quadrilateral with a texture image. The texture image are constructed at a server side. We apply the technique to some volume datasets and confirm the effectiveness.

Key Words: Flow Visualization, VRML, Finite Volume Method, Computational Fluid Dynamics

1. 緒言

計算流体力学 (Computational Fluid Dynamics: CFD) 技術の向上の結果, その利用分野が拡大すると同時にその解析結果の恩恵を受ける人も増えている. 例えば, 営業担当者が解析結果を使って, 顧客に商品をプレゼンテーションするケースも出始めている. しかし, 多くの解析結果は, 技術論文・報告書中の参照図という形で公表され, さらに入手するにも困難な場合が多い. WWW のサーバに解析結果ファイルを格納し, これをインターネットを介して可視化する仕組みを構築すれば, 多くの人々が, WWW のクライアント端末を使ってその恩恵にあずかることが可能になる.

本論文において, 「可視化」は, 「CFD 解析結果から意味のある幾何形状を取り出し, グラフィックスディスプレイに表示すること」を意味するものとする. また, 本論文では, CFD 解析結果として 3 次元格子の各頂点で定義されたスカラデータ, ベクタデータを取り上げ, 以降これをボリュームデータと呼ぶ. 有限体積法 [1] では一般に格子の重心や格子界面の面心において物理量が定義されることが多いが, この場合, 各物理量は, 格子の頂点に外挿されていると仮定する.

Virtual Reality Modeling Language (VRML) [2] に代表されるように, 幾何形状をグラフィックスディスプレイに表示する環境は整ってきているので, 幾何形状へ変換しておく特別な環境を前提とすることなく, その恩恵に預ることが可能となる. このような形態では, 従来と比べて幅広いユーザ層が対象となるので, クライアント端末には, 特別なソフトウェアを前提としないことが望ましい. また, ローエンドクライアント端末においても実用的な「可視化」を実現するには, サーバ側でできる限り幾何形状のデータ量を削減しておく必要がある. 可視化処理で取り扱う幾何形状として最も頻度の高いデータ付 3 角形メッシュ^(注 1)を例にとると, 実用レベルで用いられているボリュームデータから生成されるデータ付 3 角形メッシュの数は数万程度である. このような規模の幾何形状データをインターネットで送信し, これをクライアントで受け取って表示するのは困難である.

この問題を解決するため, 我々は, 可視化処理をスライス面に限定し, データ付 3 角形メッシュを大きな 1 枚のテクスチャ付 4 角形で表現する手法を提案する. また, 幾何形状データをインターネットで配信するためのフォーマットとしては, VRML [2] を採用する. 第 2. 章では, インターネットを利用した可視化処理に

* 原稿受付 1999 年 5 月 6 日.

^{*1} 正員, 岩手県立大学ソフトウェア情報学部 (〒020-0193 岩手県滝沢村菓子 152-52).

E-mail: koyamada@soft.iwate-pu.ac.jp

(注 1): 各頂点で数値データが定義されているような 3 角形メッシュのこと

について説明する。第3.章では、ボリュームデータから切り出したスライス面のスキャンコンバージョン（2次元直交格子上でサンプリング）の方法について説明する。スキャンコンバージョン後については、2次元直交格子上で定義された数値データの画像化処理となる。第4.章、第5.章では、定義された数値データがスカラ、ベクタの場合の画像化手法についてそれぞれ説明する。第6.章では、本手法の評価について説明する。また、VRMLを利用した可視化サーバの構築事例について述べる。

2. インターネット上での可視化

可視化についての「参照モデル」[3]では、可視化を連続する2つの変換プロセスから構成されると抽象化する(図1参照)。ボリュームデータ(V)は、第1の変換を経て、なんらかの意味のある幾何形状データ(G)に変換される。この変換を以下、リアライズプロセスと呼ぶ。この幾何形状データは、多角形・線分・点といった、いわゆる3次元グラフィックスソフトウェアの表示プリミティブとなる。これらは、第2の変換を経て、ディスプレイ上で表示の対象となる画像データ(I)となる。以下、レンダリングプロセスと呼ぶ。

Fig. 1 Visualization Process

インターネット上で可視化を行なう場合、ボリュームデータは、サーバ側にあり、画像データは、クライアント側で表示されるものと考えても差し支えないであろう。この場合、変換プロセスがどちらで行なわれるかに関して3つのシナリオを考えることができる。

シナリオ1

第1のシナリオでは、リアライズ・レンダリングプロセスの両方がサーバ側で実行される。このケースでは、画像データがサーバ側で作られ、それがインターネットを通じてクライアント側に送られ、WWWブラウ

ザのビルトインツールを使って表示される。複数枚をまとめて送り、アニメーション表示することができるが、クライアント側で、対話的に視点を変更したりすることができない。

シナリオ2

第2のシナリオでは、リアライズ・レンダリングプロセスの両方がクライアント側で実行される。このケースでは、まず、ボリュームデータがインターネットを通じてクライアント側に送られる。クライアント側では、対話的に視点を変更したりするだけでなく、可視化手法を自由に選択することができる。このケースに分類されるNASAのFASTシステムでは、複数ユーザの協調に基づく可視化処理を実現している[4]。

シナリオ3

第3のシナリオでは、リアライズプロセスがサーバ側で、そしてレンダリングプロセスがクライアント側で実行される。このケースでは、サーバ側で生成された幾何形状データがインターネットを通じてクライアント側に送られる。シナリオ3は、今まで述べた2つのシナリオの中間的性格を持つ。

シナリオ1では、画像データだけがネットワークを転送される場合を想定した。また、シナリオ3では、幾何形状データ（多角形、線分）とともにサーバ側で作成した画像データが転送される場合を含む。これは、テクスチャマッピングされた多角形を転送する場合を想定している。

また、シナリオ3では、シナリオ2と違って、クライアント側に特別なソフトウェアをインストールする必要がない。これは、解析担当者が試行錯誤して可視化した解析結果をその他の人が様々な視点で見のには向いている。クライアント側の処理は、幾何形状が対象となるので、これらをVRMLで記述しておけば、一般に流通しているVRMLブラウザを使って表示することができる。本報告では、このシナリオ3に焦点をあてていくことにする。

3. スライス面のスキャンコンバージョン

数値データを表示する手法として、あるスライス面を設定して、その面上でボリュームデータを可視化する手法を考える。ボリュームデータをスライス面にマッピングするには、スライス面と格子稜線との交点を計算し、交点を頂点とする3角形メッシュを作成する。頂点では、格子頂点で定義された数値データを使って補間計算を行う。スライス面上で定義される3角形の

頂点は、スライス面と格子の稜線との交点である。その数は、格子の数の $2/3$ 乗に比例し、数十万の格子からだと数万個程度となる。我々は、インターネットを通じてこのスライス面を転送しようと考えている。この場合、スライス面を構成する 3 角形の数が数万というのは大きな問題となる。

この問題を解決するためのひとつの方法は、3 角形メッシュとして表現されたスライス面形状を 1 個のテクスチャ付き 4 角形として表現することである。3 角形メッシュは、まず、スライス面を投影面とするように座標変換が行われる。この場合スライス面の法線と平行な視線を仮定して、視点座標系への変換を行う。そして引き続き、スクリーン座標系への変換を行う。各画素におけるデータ値は、頂点で定義されたデータを使ったスキャンコンバージョンによって画像データにおける各画素に展開される。

スライス面を表現する 4 角形を定義するためにスクリーン上でスキャンコンバージョンの対象となった画素領域を囲む最小面積をもつ長方形を計算する。長方形のうちスキャンコンバージョンの対象とならなかった領域には、この長方形が不自然に裏面の形状を隠蔽しないように透過属性を与える。図 2 では、網掛けになっているところに透過属性を与える。長方形の各 4 頂点の座標をスクリーン座標系から世界座標系へと変換する。この変換された座標を使って世界座標における長方形を定義する。また変換前のスクリーン座標系もテクスチャマップ用として保存しておく。

本手法では、いったん、スライス面を 3 角形メッシュとして表現して、次にその 3 角形メッシュのスキャンコンバージョンを行うという手順を踏む。この方式は、現在高速化されている可視化技術の恩恵を受けることができるというメリットがある。従って、可視化技術の専門家でなくても、ビジュアルプログラミング型可視化ソフトウェアを使って、本手法を実現することができる。例えば、スライス面作成モジュールからの出力を、視線がスライス面と垂直になるように、レンダリングすれば、テクスチャ画像を生成することができる。

一方、このような 3 角形メッシュといった中間媒体を経由することなく格子毎にスライス面上の画素点でサンプリングしてテクスチャを生成することも 1 手法（以降ボリュームスキャン法と呼ぶ）として考えることができる。このボリュームスキャン法では、スライス面上で設定される画素点を含む格子を特定し、その格

Fig. 2 Scanconversion of a slicing plane

子の頂点で定義されるデータ値を用いて、画素点におけるデータ値を補間計算する。補間計算されたデータ値は、カラーテーブルを用いて各色成分に変換される。画素点毎に格子の特定を行うと効率が悪いので、この場合、ボリュームデータを画面解像度レベルでボクセル化しておく必要がある。ボクセル化したボリュームデータを前提にすれば、スライス面上に位置するボクセルデータを効率よく取り出すことができる。

昨年、ボクセルデータをリアルタイムでボリュームレンダリング表示するための PC 用専用ボード [5] が発表されるなど、ボクセルデータを前提とした可視化環境が整備されてくる可能性が出てきている。将来的には、ボリュームスキャン法についての検討も行っていきたいと考えるものの、以下の理由で、今回はボリュームスキャン法を採用しなかった。

- 格子をボクセル化するための計算時間が大きい
- 画面解像度レベルで作成されたボクセルデータを保存するためのメモリ空間が大きい
- 非構造格子の場合、境界付近でアーチファクトが出やすい

4. スカラ場の画像化処理

可視化の対象がスカラデータが頂点で定義された 3 角メッシュである場合、一般的に、等高線表示を行うか、または、カラーlookupアップテーブルを用いたコンタ表示を行うことが多い。カラーlookupアップテー

ブルは、スカラデータとカラーデータの関係付けを行うために使用する変換表である。

等高線表示を行う場合、3 角形単位で局所的な等高線を線分として計算し、この線分の集合を画像データにおける各画素に展開する。次に、コンタ表示を行う場合、3 角形の各頂点でスカラデータからカラーlookupアップテーブルを使ってカラーデータに変換し、スキャンコンバージョンアルゴリズムを使って、3 角形ごとに面内の塗り潰しを行う。各頂点でカラーデータに変換することなく、スカラデータをそのまま使い、各画素に展開する段階でカラーlookupアップテーブルを使ってカラーデータに変換する方法（データマッピング）も存在する。一般的に、3 角メッシュの総頂点数よりも画素数のほうが多いので、データマッピングでは、カラーlookupアップテーブルの参照回数が多くなり、若干の計算時間が多く必要となる。これら塗り潰し計算は、グラフィックスライブラリを用いることができるので処理の高速化を期待することができる。

いずれの表示法においても線分や3 角形といった幾何形状から画像への変換は、サーバ側で行う。このような幾何形状を IndexedLineSet や IndexedFaceSet といった VRML のノードに変換して、これらをクライアント側に転送し、クライアント側でスキャンコンバージョンを実施することも可能である。この場合、ネットワークにおける転送量は3 角形メッシュの大きさに比例する。一方本手法のように画像をサーバ側で作成しておく、三角メッシュの大きさに関わらず、ネットワークにおける転送量は、画像の解像度を固定化すれば、一定値となる。

5. ベクタ場の画像化処理

ベクタデータが頂点で定義された3 角メッシュをスキャンコンバージョンすると2次元直交格子上で定義されたベクタ場が生成される。このような2次元ベクタ場の可視化に対しては、Cabral が提案した流線畳み込み (Line Integral Convolution: LIC) 法が有効である [6]。LIC 手法では、ホワイトノイズ画像を入力とし、ベクタ場から計算される局所流線に沿った畳み込み積分計算を行い、ベクタの方向に対してコヒーレンスのある画像を出力する。この結果、LIC 手法により生成された画像を通してベクタ場の大局的な傾向や複雑な構造を直感的に理解することが可能となる。

LIC により作成された画像では、ベクタの方向がわからないという問題がある。この問題を解決するため

に方向つき流線畳み込み (Oriented LIC: OLIC) 法が提案されている [7]。この方法では畳み込み積分の対象をホワイトノイズ画像とせず、適当な間隔でスポットの置かれた画像とし、そして畳み込みの関数としてベクタの方向に沿って単調減少するようなものを使用している。この結果インクがベクタ場にしがたってにじんていくような画像を生成することができるので、ベクタデータの方向を理解することが可能となる。その一方で LIC の特徴であるベクタ場全体の稠密な表現という特徴を失ってしまう。

我々は、ベクタデータの方向を明確にするために、LIC 画像に方向を示すアイコンを重ね合わせる (図 3 参照)。LIC では、ベクタの方向に沿って輝度の大き

Fig. 3 Shape of Arrow Icon

さに連続性のもった画像を生成する。この連続性のある領域に対して、

- 輝度値が極大となる
- 輝度値の最大傾斜方向がその点でのベクタデータと直交する

という条件を満足する画素点にアイコンを配置する。アイコン自身の輝度は、その画素の輝度値に合わせるものとする。図 4 は、輝度が極大となる画素点近傍における輝度分布を模式的に表現したものである。画素点の位置を示す白丸からその上方に位置する黒丸までの距離は輝度値を示す。

輝度値が極大となる点を探索するためにそのまわりに隣接する画素点での輝度値との間で比較を行う。また、極大点における最大傾斜ベクタについては、極大点を頂点とする最大4つの格子毎に勾配ベクタを計算し、そのなかで絶対値が最大となるものを選択する。それぞれの4 角形格子内部では4つの頂点で定義された輝度値から2次元線形補間を使って輝度分布が計算できると仮定すると、第1象限に位置する格子から計算される勾配は、

$$\nabla S_{i,j}^1 = (S_{i+1,j} - S_{i,j}, S_{i,j+1} - S_{i,j}) \quad (1)$$

Fig. 4 Placement of Arrow Icon

同様に、第2、3、4象限では、

$$\nabla S_{i,j}^2 = (-S_{i-1,j} + S_{i,j}, S_{i,j+1} - S_{i,j}) \quad (2)$$

$$\nabla S_{i,j}^3 = (-S_{i-1,j} + S_{i,j}, -S_{i,j-1} + S_{i,j}) \quad (3)$$

$$\nabla S_{i,j}^4 = (S_{i+1,j} - S_{i,j}, -S_{i,j-1} + S_{i,j}) \quad (4)$$

となる。ここで、 $S_{i,j}$ は、座標 (i,j) に位置する画素における輝度値を表す。また、上添字は、象限を表す。図4では、第3象限において計算された勾配が最大傾斜ベクタとなる。

6. 評価

テクスチャベース可視化手法の有効性を検証するために、直交格子で定義されたスカラー場から2通りの方法でスライス面を計算した。ひとつは、1つのテクスチャ付き4角形を使って、そしてもうひとつは、複数のカラーデータ付き3角形を使ってスライス面を表現する。前者において、GIFファイルで表現されたテクスチャの解像度は、428*603画素で、そのサイズは、1枚あたり9.87KBであった。VRMLファイル中テクスチャ付き4角形を1つ表現するのに、0.74KB、そしてその他の記述の部分で、4.05KB必要とした。以上から、N枚のスライス面を表現するために必要とな

るディスクスペースSは、以下のように計算される。

$$S = 4.05 + 10.61 \times N(KB)$$

スライス面をカラーデータ付き3角形(頂点の総数が1246)で表現した場合、1枚のスライス面を表示するために150KB必要となる。したがって、この場合、N枚のスライス面を表現するために必要となるディスクスペースは

$$S = 4.05 + 150.0 \times N(KB)$$

となる。数十万格子のボリュームデータから計算されるスライス面は、さらに多くの3角形から構成されるからディスクスペースの観点において本手法が有利であることがわかる。

ベクタ場の画像化処理手法についての評価を行う。2次元直交格子で定義されたスカラー場から次のベクタ場

$$(u_1, v_1) = (y - 0.5, -x + 0.5)$$

をLIC手法を用いて画像化する。入力ホワイトノイズ画像、および出力画像ともに200*200画素から構成される。図5では等間隔にアイコンを配置している。輝度が低くなっているところに対してもアイコンが描画されるので、アイコンの形状が認識されにくい画像となった。すべてのアイコンを高い輝度で描画するとアイコンにより重要なベクタ場の分布が遮蔽される場合があり、望ましくない。

Fig. 5 Icons located at a fixed interval

輝度値が極大となり、最大傾斜方向がその点でのベクタデータと平行するような点にアイコンを配置した

画像を図 6 に、また、最大傾斜方向がその点でのベクタデータと直交するような点にアイコンを配置した画像を図 7 に示す。極大点は、2586 あり、そのうち最大傾斜方向がその点でのベクタデータと直交するような点の数は、72 であった。また最大傾斜方向がその点でのベクタデータと平行となる点の数は、68 であった。

Fig. 6 Icons located at extrema points(1)

Fig. 7 Icons located at extrema points(2)

図 7、図 6 においては、図 5 に比べると周辺に比べて輝度値の高い領域にアイコンを配置しているので、アイコン形状が認識しやすく、したがってベクタ場の方向がわかりやすくなっている。さらに、図 7 では、図 6 にくらべてアイコン形状が見易くなっている。こ

れは輝度がベクタ場に沿って細長く分布している領域にアイコンが配置される割合が高くなっているからであろう。

我々は本論文で提案するテクスチャベースの可視化技術を使ってリモート可視化システムを開発した。図 8 は本システムの概要図である。ユーザは、JAVA アプレットをダウンロードし、必要な可視化パラメータを入力する。入力されたパラメータを、サーバ側に転送し、スカラ場、ベクタ場にしながら適切なテクスチャ付 4 角形を生成し、VRML フォーマットで配信する。

Fig. 8 Remote Visualization System

ベクタ場を可視化する場合、しばしば渦の位置を調べるのが目標となる。一般的に渦の場所を特定するには、試行錯誤しながらさまざまなスライス面上でベクタデータを矢印で表示することが多い。我々は、スライス面の設定については、できるだけ自動化したいと考える。WWW の利用を前提とした可視化では、最適なスライス面を探し出すための試行錯誤を十分に行うことができないと予想されるからである。

我々は、渦の周りの速度場を効果的に可視化するために自動スライス面設定機能を実装している。渦中心の特定を自動化するために特異点理論を用いている [8]。特異点は、ベクタ場において、ベクタデータが零ベクタとなるような点である。特異点理論では、特異点において定義される速度勾配テンソル J_v の特性方程式を解き、固有値を求める。固有値の中に共役複素数が含まれる場合、渦中心であると判断する。この場合、他の実固有値に対応する固有ベクタを法線とし、かつその特異点を通過するスライス面を設定する。

図 9 は、自動スライス面設定機能で生成されたテクスチャつきスライス面である。我々は、与えられた

Fig. 9 Visualization of vortex flow using VRML

渦中心と渦ベクタをもつベクタ場を計算し、3次元格子の各頂点にそのデータ値を定義した。このベクタボリュームデータから渦中心と渦ベクタを計算し、前述の方法を用いてスライス面を自動設定した。LIC画像をマッピングしたスライス面の表示にはVRMLブラウザを使用した。

7. 結言

本論文において、サーバ側で出力されたCFD解析結果をインターネットを利用して可視化するために有効なテクスチャベースの手法を提案した。ベクタ場を可視化する場合には、LIC手法を使ってテクスチャ生成を行った。ベクタ場の方向を明確にするためにLIC画像において輝度がベクタ場に沿って分布する場所を探索し、そこに方向を示す矢印アイコンを配置した。また、テクスチャベースの手法と3角形メッシュベースの手法についてその転送データ量を比較し、後者が前者に比べ10倍ほどになることを確認した。

謝辞 可視化サーバシステム開発に御協力いただいた(株)ソフトウェアクレイドル新規事業推進室室長高橋氏に謹んで感謝の意を表する。

文献

- [1] Patankar, S. V., "Numerical Heat Transfer and Fluid Flow," Hemisphere Publishing Corporation, (1980).
- [2] Gavin Bell, Rikk Carey, and Chris Marrin, "The Virtual Reality Modeling Language Specification Version 2.0," <http://vag.vrml.org/VRML2.0/FINAL/>, 1996.
- [3] Upson C., et al, "The Application Visualization System: A computational environment for scientific visualization," IEEE Computer Graphics and Applica-

tions, Vol. 9, No. 4, pp. 30-42, 1989.

- [4] Clucas J., "Interactive Visualization of Computational Fluid Dynamics Using Mosaic," Proceedings Second International WWW Conference'94, Chicago, October 1994.
- [5] Pfister, H., Hardenbergh, J. Knittel, J. Lauer, H. and Seiler, L., "The VolumePro Real-Time Ray-Casting System," Computer Graphics, Vol. 24, No. 5, pp. 63-70, 1999.
- [6] Leeuw, W. D. and Liere, R., "Comparing LIC and Spot Noise," Proceedings of IEEE Visualization'98, (1998).
- [7] Rainer, W. and Eduard G., "Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet," Proceedings of IEEE Visualization'97, pp. 309-316 1998.
- [8] Koyamada K. and Itoh T.: Seed Specification for Displaying a streamline in an Irregular Volume, Engineering with Computer, Vol. 14, pp. 73-80 (1998).