グラフィクスとCAD 87－9
（1997. 10. 17）

VRMLによる可視化のためのデータ削減技術

小山田 耕二
日本アイ・ビー・エム株式会社

概要

本論文では,「可視化」に特化したデータ削減手法を2つ提案する．ひとつは，テクスチャマッピング機能を利用したスライス面単純化手法で，可視化の結果得られた幾何形状を対象とする．もうひとつは，遺伝的アルゴリズムを用いたボリュームデータ単純化手法である．単純化とは，誤差，サイズ両者を最小化する多目的最適化問題と考え，この問題解決のため，遺伝的アルゴリズムをベースとした基本的アプローチを提案する．

Simplification Techniques for Visualizing Volume Datasets Using VRML
Koji Koyamada
Tokyo Research Laboratory, IBM Japan Ltd.

In this paper, we propose two simplification techniques for visualizing volume datasets. One is to simplify a volume slice geometry using a texture mapping capability. The other is to simplify a volume dataset using genetic algorithms. We assume that this simplification can regarded as a multi-objective optimization problem of minimizing both the error and the dataset size, and propose a methodology based on genetic algorithms.

はじめに

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初めている．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ーに表示すること」を意味するものとする．VRML [VRML96] 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ー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と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，特別なソフトウエ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．

幾何形状のデータ削減については，すでにさまざまな手法が提案されている．もちろん，これらの手法を「可視化」の結果生成された幾何形状に対して適用することは有効である．本論文では,「可視化」に特化したデータ削減手法を考察する．例えば,「可視化」は，スライス平面上で行なわれることが多い．この場合については，3角形群は平面上に存在しているので，この平面を大きな1枚のポリゴンで表現することが有効である．また,「可視化」の対象であるCFD結果(以下ボリュームデータ)そのものを単純化することも有効な1手段である．単純化とは，もとのボリュームデータからの誤差の劣化をある程度認めつつ，そのサイズを削減することであるとしたならば，誤差，サイズ両者を最小化する多目的最適化問題となる．この問題解決のため，遺伝的アルゴリズムをベースとした基本的アプローチを提案する．

VRML2.0

Virtual Reality Modeling Language(VRML)は，インターネット上における3Dインタフェースとして開発された仮想3次元構築言語である．VRMLは以下のような経緯をもつ．

1994年4月	WWW国際会議で提案
1994年6月	メーリングリスト作成
1995年6月	VRML1.0の仕様決定
1996年8月	VRML2.0の仕様 [VRML96] 決定

VRMLは，ノードとフィールドから構成される．ノードは，仮想3次元空間のオブジェクトを記述するもの

で，フィールドは，それぞれのノードが表現するオブジェクトの属性を定義する．VRML1.0では，静的なモデルを記述する機能を持っていた．VRML2.0になって，まず，動きに関する記述が可能になった．これにより，アニメーションを実現したり，ユーザの動きを検知できる．そして，次に，オーディオやムービーといったマルチメディアデータがサポートされるようになった．最後には，VRMLファイル構造について，オブジェクト指向化がすすんだ．これにより，見やすくなったと同時に既成のノードの性質を継承して新たなノードを作成できるようになった．

インターネット上での可視化シナリオ

可視化についての「参照モデル」[Upso89]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エ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，ディスプレー上で表示の対象となるイメージデータ(I)となる．以下，レンダープロセスと呼ぶ．この変換では，どこから見ているのかといういわゆる視点が考慮される．

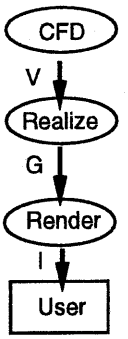

図 1: 可視化処理のながれ

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，イメージデータは，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・レンダー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，イメージ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・レンダー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[FAST94]．

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，そしてレンダー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．しかし，シナリオ2と違って，クライアント側に特別なソフトウエ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るのには向いていそうである．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．シナリオ3では，クライアント側に比較的ローエンドなPCを仮定するので，クライアント側で処理される幾何形状は，できるかぎり単純化しておく必要が出てくる．

可視化へのVRML2.0適用

シナリオ3では，サーバ側において，幾何形状データを適切なVRMLノードに写像し，結果をVRMLファイルとしてクライアント側に配信する．可視化手法とVRMLノードは，例えば，以下のように対応する：

スライス面表示	IndexedFaceSet
等値面表示	IndexedFaceSet
矢印図表示	Cylinder+Cone
流線表示	IndexedLineSet
パーティクル表示	Sphere+PositionInterpolator

これらのノードから構成されるVRMLファイルを受信することにより，原理的には，解析結果をWWWブラウザのプラグインツールで可視化することができる．ここで,「原理的には」といったのは，一般的に解析結果から変換された幾何データのサイズは膨大であるため，比較的ローエンドなPCでは，表示困難になるかもしれないからである．したがって，サーバ側で幾何形状の削減処理を行なう必要がある．次章で，サーバ側で必要となる削減技術について述べる．

クライアントからサーバへユーザの指示伝達は，例えば，Sensorノードと Scriptノードを組み合わせて実現する．Sensorノードは，ユーザの操作を検知して，操作に関する情報（イベント：データ付きメッセージ）を他ノードに送信する機能がある．また，Scriptノードは，そのフィールドにイベントが到着すると指定されたプログラムを起動する機能がある．現在，プログラム言語として，JavaやJavaScriptがサポートされてるので，これらを使って，サーバ側と通信するプログラムをかくことができる．したがって，Sensorノードから Scriptノードに対して，イベントが送られるように関係づけておけば，ユーザがVRMLノードから作られた3Dアイコンを操作することにより，サーバ側に対して，流線の開始点やスライス面の指示を行なうことができる．

シーンにおけるアイコンの配置

VRMLシーンの中には，アイコン化された標識（標識アイコン）や仮想空間内で持ち運び可能な計測器（プローブアイコン）を配置しておくと便利である．ここで，標識アイコンの設置場所として特異点 (critical point) [Kym96] を提案したい．特異点は、ベクタ場において、ベクタデータ (u,v,w) が零ベクタとなるような点である。特異点理論 (critical point theory) によると、特異点付近で与えられたベクタ場を線形化することにより、その点付近でのベクタ場の挙動を調べることができる [Helm91]. ベクタ場の挙動を調べるには，特異点における速度勾配テンソル固有値λおよび固有ベクタを計算する。特異点は、計算された固有値λを用いて分類される [Helm91]。固有値が全て実数の場合には、その特異点はソース点／シンク点／鞍点と分類される。それぞれの分類に応じたアイコン形状を割り当てることにする．正の固有値に対応する固有ベクタは特異点から離れる方向のベクタデータに対応し、負の固有値に対応する固有ベクタは特異点に向かう方向のベクタデータに対応する。また、固有値が複素数である場合は、ベクタデータが局所的に渦を形成していることに対応する．

渦の場合は，複素数の虚部が渦の回転速度に対応するので，この速度に応じた音を出すSoundノードを割り当てると効果的であろう．Soundノードは，音源の空間的な情報，音の指向性などを記述するノードである．音の指向性を，渦ベクトルの方向に一致させておくと臨場感のあるシーンを構築することが期待できる．Soundノードにおいて，音源は，AudioClipノードを用いて指定する．AudioClipノードでは，音源のピッチをコントロールできるので，渦の回転速度に比例するピッチを与えることにより，渦についての直観的な音表現を期待できる．

幾何形状データの削減

スカラ場の可視化では，しばしばスライス面でのコンタ図を表示する．そのスライス面は，ボリュームデータの格子群との交点を頂点とする膨大な数の3角形から構成されている．このようなスライスコンタ面を構成する3角形は，同一平面上に存在するので，テクスチャマッピング機能を使って幾何形状データを削減することが可能である．ベクタ場の可視化であってもスライス面上でテクスチャに変換されていれば，この削減方法を利用することができる．この手法のポイントは，スライス面について、レンダープロセスをサーバ側で行い、テクスチャの形式にしておくことである。一般的に、レンダープロセスでは，視点座標系を指定する必要がある．この場合，コンタ面に垂直な方向に視点を設定することができ、実際の視点とは関係がない。また，注視点は，スライス面の重心に，上方ベクタは，画面上におけるコンタ面の占有率ができるだけ高くなるように決めることができる．したがって、スライス面のテクスチャをサーバ側で先に計算しておくことが可能である。

サーバ側におけるレンダープロセスでは，スライスコンタ面を構成する全3角形を保持しているが，一旦テクスチャが出来上がるとこれらは不要になる．そのかわり，スライスコンタ面の投影面を1つの4角形で定義しておく（図2参照）．この場合，投影面のうち，実際にテクスチャが作成されていない領域がその裏側にあるオブジェクトを隠してしまうので，このような背景領域については，透明属性を与え，テクスチャ領域だけが有効となるようにする．

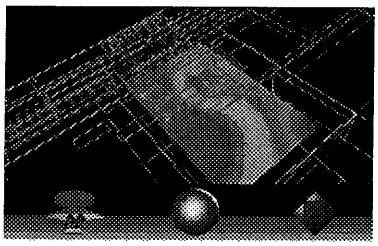

図 2: Application of the texture mapping

ボリュームデータの削減

スカラ場の可視化手法のひとつに等値面表示がある．この表示手法では，必ずしも等値面が平面とならないために，先ほど述べたテクスチャマッピングを用いた幾何形状データの削減手法を利用することができない．すると，考えられる方法としては，サーバ側で

- 得られた等値面データを削減する
- あらかじめ，ボリュームデータを削減する

のいずれかを行なうことになる．前者だとクライアントのアクセスごとに等値面データの生成に加えて，その削減処理が必要となる．後者は，一旦，サイズの小さいボリュームデータを作っておけば，クライアントのアクセスに対して，等値面データの生成処理だけとなる．しかも，対象となるボリュームデータのサイズが小さくなっているので，生成処理自身も軽くなるものと推定できる．このため，等値面表示には，あらかじめボリュームデータを削減しておくことにする．

ここでは，遺伝的アルゴリズム (GA) [GOL89] を用いて，精度をある程度保ったまま，格子の総数を削減する手法を開発する．では，まず，この精度を定義する．一般的に，もとの格子空間で張られるデータ空間 $S_{original}(x,y,z)$ と削減された格子空間（以下，構成する格子を単純化格子と呼ぶ）で張られるデータ空間 $S_{reduced}(x,y,z)$ との差が小さいほど精度が高いといえるであろう．したがって誤差指標として，

$$Err = \int_{all} \sqrt{(S_{original}(x,y,z) - S_{reduced}(x,y,z))^2} dV \quad (1)$$

と定義する．今，簡単のため，数値データは，スカラデータとしておく。式 1の積分には，ガウスルジャンドルの数値積分を用いる．この積分法は，数値シミュレーションでは，よく用いられている方法で，全体座標系 (x,y,z) に関する積分を格子の局所座標系 (p,q,r) で表現することがひとつの特徴である．我々は，以下の示す手順でこの積分を計算している (図 3参照)．

1. Err を 0 に初期化
2. 元の格子　全てについて
   (a) ガウスの積分点 $(p_{original}, q_{original}, r_{original})$ について
      i. 元の格子内で定義された補間関数を使って，$S_{original}$を計算
      ii. ガウスの積分点を全体座標系 (x,y,z) へ変換
      iii. (x,y,z) を含む単純化格子を検索
      iv. 単純化格子内で定義された補間関数を使って，$S_{reduced}$を計算
      v. $W(S_{original} - S_{reduced})^2 det[J]$ を Err に加算

ここで，W は，ガウス積分点で定義される重み，そして det[J] は，座標変換ヤコビアンの行列式を表す．

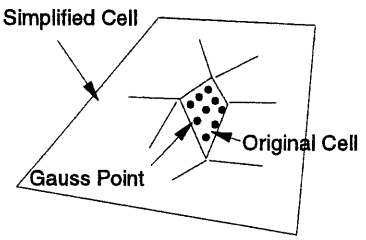

図 3: 誤差指標の計算

以下，CFD でよく使われる直交差分格子を仮定した格子削減手法について述べる．

各軸方向における格子の間引き

直交差分格子では，直交する 3 つの軸ごとに分割点が定義されている．これらの軸をそれぞれ I,J,K 軸とする．もっとも単純なのは格子の間引き方法は，それぞれの軸ごとに使わない分割点を指定していくことである．使う・使わないをそれぞれビットのオン・オフに対応させると，ひとつの間引き方法は，各軸方向の分割点数に等しい長さをもつ 3 本のビット列として表現される．3 本のビット列のパターンをいろいろ変化させて，サイズの小さなボリュームデータを考え，式 1から精度を計算する．そのなかで精度が高く，かつ格子数が少ないものを捜し出せば，精度をある程度保ったボリュームデータが得られることになる．

この探索をしらみつぶしに行なうと現実的な時間で最適解を得るのは困難であろう．各軸方向の分割数をIM, JM, KM とするとビット列のパターンの総数 N は，$2^{IM+JM+KM-3}$となるから，例えばIM+JM+KM-3=100 として N=1.26e+30 となってしまう．このような多次元パラメータの組合せ最適化に対して，遺伝的アルゴリズムが有効とされている．ここでは，まず遺伝的アルゴリズムの概要を述べた後，格子の間引き問題への適用について述べる．

遺伝的アルゴリズムについて

遺伝的アルゴリズム (Genetic Algorithm 以下，G.A.) は，生物の遺伝と進化のメカニズムを工学的にモデル化したものである．歴史的には，John Holland により 1970 年頃から研究され，1975 年正式に発表された．G.A. において，個体は，ビット配列として表現される遺伝子とその個体の適合度を用いて抽象化される．G.A. では，まず，この個体の集団を考えます．G.A. は，この集団のなかで各個体同士が交叉と突然変異を繰り返しながら，適合度の高い個体ほど生き残り，子孫を残すことができるようないわゆる自然淘汰のメカニズムを実現する上でのフレームワークを提供する．

G.A. の基本的操作には，次の 3 つである．

選択 (Selection)	集団のなかで適応度の分布に従って，次のステップで交配を行なう個体の生存分布を決定する．適応度の分布に基づいているので，適応度の高い個体ほどより多くの子孫を残しやすくなる．
交叉 (Crossover)	2 つの個体間で遺伝子をくみかえて，新しい個体を発生させる．
突然変異	遺伝子のある部分の値を強制的に変える．

G.A. の処理の流れを以下に示す (図 4参照)．

1. 遺伝子型を決定する．
2. 初期遺伝子集団を決定する．
3. 各個体の適応度を評価する．
4. 適応度に基づいた選択を行なう．
5. 選択された個体同士による交叉を行なう．
6. 個体の突然変異を行なう．
7. ステップ 3 に戻る．このループを世代と呼ぶ．

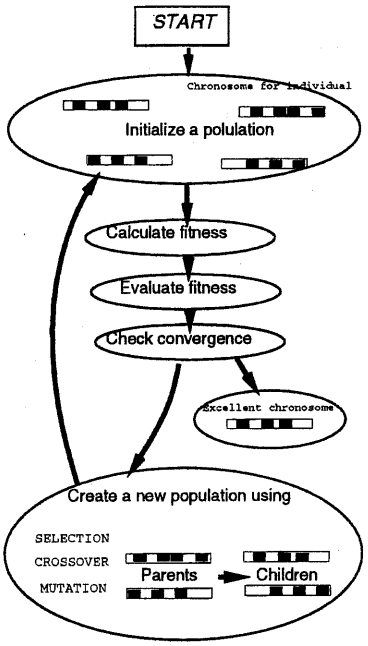

図 4: G.A. の概要

G.A. では，対象とする問題を遺伝子の形で表現し，遺伝子の要素が異なる様々な個体を発生させる．並列的な解の探索を行なう G.A. の良さが発揮されるよう，一般的には，個体の数は，一般的には数十以上とする．

格子の間引き問題への適用

GA において使用される個体として，I,J,K 軸毎に定義された分割パターンを対応させる．したがって，個体に対し，I,J,K 軸毎に 3 つの遺伝子を用意する．それぞれの遺伝子において，各軸方向の分割数に対応して，IM-1, JM-1, KM-1 個のビットを持たせる．このビットがオンのとき，そこには分割点が定義されているとする．個体のなかで精度が高く，かつ格子数が少ないものを捜し出すことを目標とする適応度として

$$Fitness = \frac{1}{(Err + 1.0)^5} \times \frac{N^{original}}{N^{reduced}} \qquad (2)$$

を定義する．

10*10*10 の格子を使い，各格子点に以下のようなスカラデータ

ボリュームデータ1　$S(x,y,z) = x + y + z$

ボリュームデータ2　$S(x,y,z) = sin(z) \times sin(y)$

を割り当て，定義された適応度の妥当性を検証した．尚，交叉率，突然変異率は，それぞれ，0.1, 0.02とした．ボリュームデータ1の場合，275世代目で1*1*1の格子に，また，ボリュームデータ2の場合は，95世代目で1*10*10の格子に収束した．

もう少し複雑な例として，

$$if(x > 5) \quad S(x,y,z) = x + y + z$$
$$else \quad S(x,y,z) = sin(z) \times sin(y) \times sin(x)$$

を考える．このボリュームデータ3では，x座標が5.0より大きいとき，線形に分布し，それ以外では，複雑な分布となっている．191世代目で以下の分割パターンが得られた．

x-direction	11111110001
y-direction	11111111111
z-direction	11111111111

x=6.0からx=10.0までの4つの格子が1つにまとめられているのでこの結果は満足のいくものといえる．この削減技術を使って，ボリュームデータのLevel of Detail(LOD)化を実現することができる．

おわりに

本報告において，サーバ側で出力されたCFD解析結果をインターネットを介して可視化するシナリオ，

シナリオ1	インターネット上にイメージデータを送信するシナリオ，
シナリオ2	インターネット上にボリュームデータを送信するシナリオ，
シナリオ3	インターネット上に幾何形状データを送信するシナリオ

を取り上げた。そのうち、VRMLの恩恵を受けることのできるシナリオ3について実用的な可視化のために必要となるサーバ側データ削減技術

- 幾何形状データ削減技術
- ボリュームデータ削減技術

について説明した．

参考文献

[Upso89] Upson C., et al, "The Application Visualization System: A computational environment for scientific visualization," IEEE Computer Graphics and Applications, Vol. 9, No. 4, pp. 30-42, 1989.

[GOL89] Goldberg D. E., "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.

[FAST94] Clucas J., "Interactive Visualization of Computational Fluid Dynamics Using Mosaic," Proceedings Second International WWW Conference'94, Chicago, October 1994.

[VRML96] Gavin Bell, Rikk Carey, and Chris Marrin, "The Virtual Reality Modeling Language Specification Version 2.0," http://vag.vrml.org/VRML2.0/FINAL/, 1996.

[Kym96] Koyamada K., Doi A., Chiba S. and Sannakanishi S., "Flow Visualization Using Tetrahedral Cell Grid-Application to Computational Flow Simulation," Trans. of IEICE, DII, Vol. J79-D-II, No. 11 pp. 1871-1878, 1996

[Helm91] Helman, J. and Hesselink, L., "Visualizing Vector Field Topology in Fluid Flows," IEEE Computer Graphics and Applications, Vol. 11, No. 3, pp. 36-46.