東京大学大学院新領域創成科学研究科
人間環境学専攻

修士論文

フラクタル図形を利用した３次元構造データのマルチレゾリューション可視化に関する研究

2009 年 2 月 17 日提出

指導教員　岡本　孝司　教授　　印

学生証番号　４７－０７６７８５

岩丸　雅紀

目次

図目次

表目次

第１章　序論

１-１　可視化とは

可視化（visualization）とは，人間が直接的には認識不可能もしくは困難である対象に対し代替となる描像を与えることで人間が視覚によって直接的に認識可能な形態へとその対象を変換する行為を指す．この可視化のために用いられる技術を可視化技術と呼ぶ．他の多くの技術と同様，可視化技術はそれを適用するにあたり目的をもつ．一般的に可視化の目的は可視化対象に対し視覚的に認識可能な形態への変換を施すことによって何らかの発見・洞察・状況把握を可視化主体である人間に促すことにある．可視化対象に可視化技術が適用された後の描像を可視化結果と呼称する．可視化結果は人間の視覚によって認知せらるることを前提としているためほぼ例外なく図や記号・絵といった形態をとり，これに文字が併用されることもある．人間の認識能力により解釈されることを期待されるものであるため，線や点などといった単純かつ抽象度の高い図形が好んで用いられる傾向にあり，これらの図形を組み合わせることで可視化対象に潜むある種の構造を描出し可視化主体に認識させることもしばしば行われる．したがって visualization と graphics とは極めて密接な関係にある．そのうえで可視化とグラフィクスとを区別にするものは，可視化は先に挙げた通り何らかの発見・洞察・状況の把握といった効果を期待して行われる点である．

１-２　Scientific Visualization

より実用に即した可視化例を取り上げるために，可視化における２大潮流である Scientific Visualization（科学的可視化）と Information Visualization（情報可視化）について述べる．まず本項において Scientific Visualization における実例を概説する．Scientific Visualization とは科学現象の実測実験もしくは科学的現象を再現した数値シミュレーションの結果として得られたデータを可視化する研究領域である．これは科学実験における根幹をなす観察・分析を補助する重要な分野であり，その原型はコンピュータの登場以前まで遡ることができる．Scientific Visualization では科学現象における観測もしくは計算可能な物理量の分布・連続性・時間変化等を可視化することでその原理や影響を解き明かしたり，可視化結果に現れた未知の現象を発見したりすることを目的とする．必ずしも観測や数値シミュレーションにより得られた物理量をそのまま可視化するわけではなく，目的とする現象が可視化結果において視認しやすいように必要に応じてある種のデータ変換・事前処理・フィルタリング等を施すことは一般的であ

る．Scientific Visualization の代表的な適用領域は流体・医療データ・宇宙物理・分子化学計算などの可視化である．Scientific Visualization で用いられる可視化手法には細かな変種は多数あるもののいくつかの限られた手法が改良を重ねられながら用いられている．例えばボリュームレンダリング（direct volume rendering）[5][6][7]，等値面（Iso Surface），流線（streamlines）[8]などがそれにあたる．

１-２-１　流体の可視化

目視でその運動を認識することの難しい流体は Scientific Visualization の格好の対象である．流体の複雑な挙動とその影響を解析することは自動車・航空機を始め様々な産業分野において重要であり関心の高いテーマである．

インクや染料，煙，水蒸気等を用いた流体の可視化実験はコンピュータの登場以前から行われている．この手法は実験により実際の流体を直接的に可視化可能であり，複数の流体の攪拌や混合の様子を可視化することも可能である．現代においても様々なトレーサー粒子を用いた可視化実験が盛んに行われており高速度カメラを用いて微小時間幅におけるトレーサー粒子の変位を測定することで撮影領域の局所流れ場を可視化する粒子画像流速測定（PIV / Particle Image Velocimetry）などの手法がある．

しかしながらこうした実験計測ベースの手法はある問題点を抱えている．ある物理現象を可視化しようとする際には当然実験によりその現象を再現する必要がある．従って経済的・時間的・労力的コストを鑑みて再現困難もしくは不可能な現象に対してはこうした実験計測による可視化手法を適用することはできない．

そこで近年ではコンピュータによる数値計算性能向上に伴い数値シミュレーションを用いて様々な流体運動を解析する数値流体力学（CFD / Computational Fluid Dynamics）が大規模流体実験を代替するようになった．CFD では流体運動の支配方程式である Navier-Stokes 方程式を数値的に解くことによって実験では再現不可能な複雑かつ大規模な流体運動を低コストでシミュレートすることが可能である．数値シミュレーションの結果は電子的なデータファイルとしてストレージに蓄えられる．これらの電子データは数値の羅列をビット列で表現したものに過ぎないため人間が直接的に生データを眺めて流体運動を理解把握することは不可能である．このため 1980 年代中ごろからコンピュータを用いて数値シミュレーションの結果を可視化する技術が発展を遂げた．折しもグラフィクスワークステーションの登場などでコンピュータのグラフィクス性能が格段の進歩を続けていた時期である．これ以後コンピュータ・グラフィクス（CG）と Scientific Visualization とは切り離すことのできない技術となった．

1-2-1-1　流線による可視化

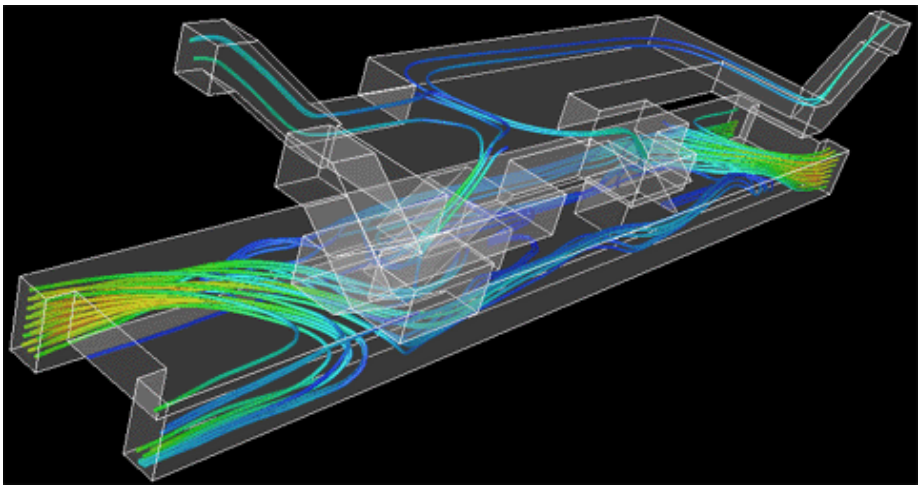

Fig. 1　流線による可視化 [36]

流線(Fig. 1)はある時刻における流れのベクトル場において定義される曲線もしくは直線群であり，その線上のあらゆる点において接線が速度ベクトルの方向と一致するものを指す．流線はある一瞬における流れ場の様子を把握する目的においては非常に効果的な可視化手法である．ただし流線その定義において流れ場の時間変化を考慮していないため非定常流れにおける物理量の経時輸送を可視化するためには流体粒子の時間的移流を曲線表示した流跡線を用いる方が効果的である．流線（および流跡線）によって大規模かつ複雑な流れ場における流体の挙動を把握するためには多数の流線を一度に描画する必要が生じる．またその際に用いられるデータは CFD によってシミュレートされた計算結果である場合が多い．従って現代においては先に述べたコンピュータのグラフィクス処理・描画能力を有効に活用し，多数の流線・流跡線をポリゴンのラインとしてリアルタイムに可視化することによってそのような流れ場の挙動を把握することがスタンダードになっている．2 次元流れにおいて，流線は流れ関数Ψが一定であるような点の集合により表される曲線と考えることができる．流れ関数は次の連立微分方程式をみたす関数である．

$$V_x = \frac{\partial \Psi}{\partial y} \quad V_y = -\frac{\partial \Psi}{\partial y} \qquad (1)$$

ここで V_x および V_y は流速の X 軸方向成分・Y 軸方向成分を表す．そして流線はこの流れ関数Ψを用いて

$$\Psi = \text{const.} \qquad (2)$$

を満たす点の集合として導出できる．
しかし数値計算により高速に 3 次元的な流線を描画するという実用上は，空間上にある

始点を設定しそこから仮想的な粒子を流れ場に沿って移流させることで陽的に流線を描出する方法が適している．このときある時刻における流れ場のみを参照して粒子の軌跡を描けば流線を求めることができるし，流れ場の時間変化を考慮すれば流跡線を描画することができる．

1-2-1-2　ベクター・プロットによる可視化

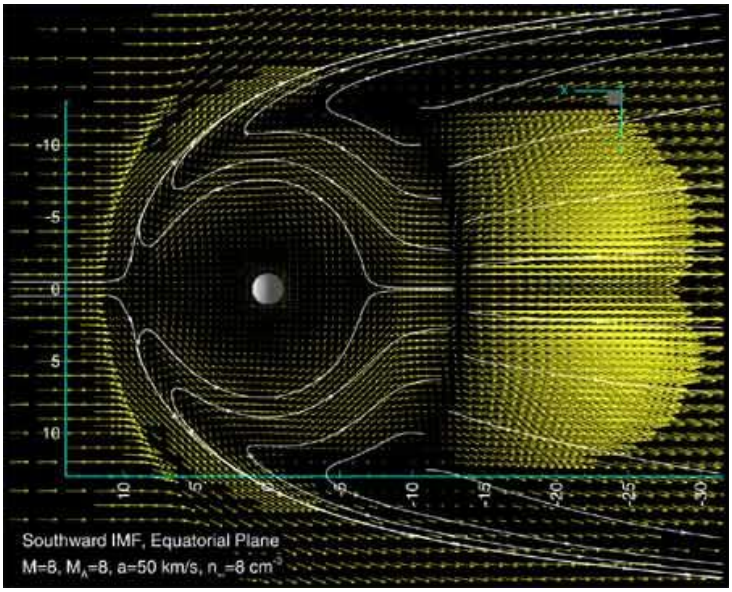

Fig. 2　ベクタープロットによる可視化 [35]

ベクター・プロット（vector plots）(Fig. 2)はベクトル場の様子を多数の矢印アイコンで表現したものである．スカラーデータの単純な可視化に用いられる散布図（scatter plots）をベクトル場に適用したものと考えることができる．この手法は離散的なサンプリングを行うだけで良いので流線の描画に比べ実装が容易である．また，矢印アイコンの形状・長さ・色といった表現要素を活用することで流線よりも多様で直感的な表現が可能な場合もある．David らは様々な２次元ベクター・プロット表現を比較・議論した[9] [10]．その結果描画パターンやアイコンの数・大きさといった要因によって可視化の質が左右されるため，伝統的に用いられている可視化表現の中には２次元ベクトル場の可視化において効果的とはいえない手法もあるということが示されている．従って流れ場の特性に合ったアイコンサイズ・形状および描画密度を選択する必要がある．

1-2-2　医療用データの可視化

人体内部の構造を可視化により把握することは医師が患者を診断し治療計画を立てるうえで重要である．元来人体組織構造の把握には解剖が必須であったが X 線撮影により非侵襲的に人体内部の様子を可視化することが可能となった．さらにコンピュータ断層撮影（CT / Computed Tomography）の登場により複数の断層撮影像から 3 次元的な人体内部構造を再構成することが可能となった．この再構成されたデータのように 3 次元空間的構造をもつデータをボリュームデータという．ボリュームデータには現実空間における温度・密度・速度といった定量的データの分布が記録されている．最も単純なボリュームデータの形式は 3 次元的に格子状に配列したボクセルとよばれる 6 面体（キューブ）の各々にデータ値（スカラー値）を格納したものである．ここでボクセルの 3 次元的な格子配列構造はそのまま空間 3 次元構造を表す．従ってこれは単純な空間の離散化モデルである．本来 3 次元空間は連続であり，ある物理量 V の分布は座標 $\boldsymbol{x}$(x, y, z)を用いて V($\boldsymbol{x}$)と表される．しかしながら有限量のビット配列としてデータを記録するためにはいくつかの離散的なサンプリング点において物理量 V のサンプリングを行いその点における物理量のみをデータとして保管せねばならない．このサンプリング点の密度を増やすほど高精細な物理量分布を記録可能であるがサンプリング点の個数に比例してデータサイズは増加する．近年 CT や数値シミュレーションの発達により大規模なボリュームデータが生成されるようになったためボリュームデータを可視化するための手法の開発が重要視されるようになった．

1-2-2-1　ボリュームレンダリングによる可視化

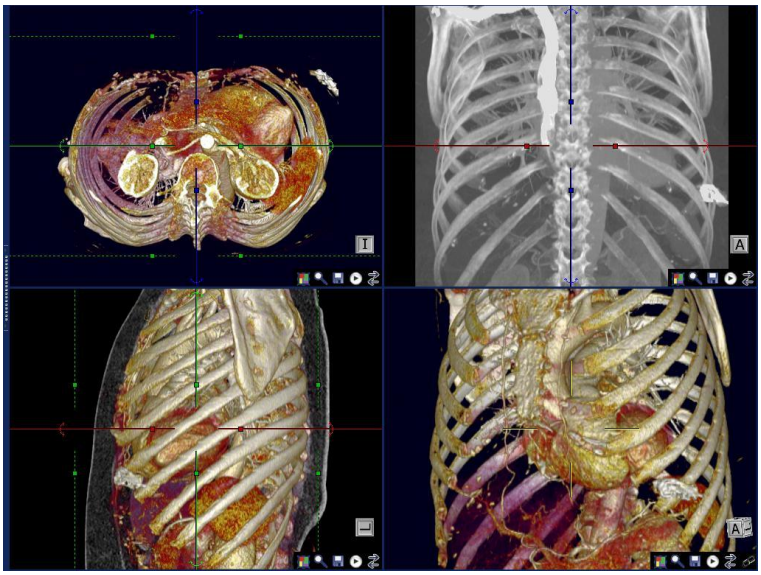

Fig. 3　ボリュームレンダリングによる人体の可視化 [34]

ボリュームデータの可視化手法として最もポピュラーなものがボリュームレンダリング（Volume Rendering）(Fig.3)である[7]．現在では多数の領域で用いられるこの手法は 1988 年に Drebin らが CT・MRI による医療用人体データを３次元的に可視化するために開発したことに端を発する[5]．この論文の中で Drebin らは視線に沿った光線(ray) のボリューム内部における減衰を考慮することで骨・筋肉・脂肪などが混合する人体内部組織を効果的にレンダリングする方法を提示している．ボリュームレンダリング法の特徴はボリュームデータをポリゴンに変換することなくレンダリングできることである．各レイの処理は独立しておりレイの投影アルゴリズム（レイキャスティング）が単純であるため並列化に適する．現在ではグラフィクスハードウェア（GPU）の並列処理能力を利用した高速なボリュームレンダリング法が多数提案されている（GPU based Volume Rendering）[6].

ボリュームレンダリングは医療以外の分野においても様々なスカラボリュームデータの可視化に用いられている．代表的なものが量子化学計算における電子確率密度分布の可視化である．量子化学計算では量子力学の原理に基づいた数値シミュレーションを行い，これにより様々な分子の構造・電荷分布状態・物性を求める．量子力学の不確定原理に従いその計算結果として得られる電荷の分布は電子がある領域に存在する確率の分布となる．この電荷存在確率密度の分布をボリュームレンダリングにより可視化する（電子雲の可視化）ことは頻繁に行われている(Fig. 4).

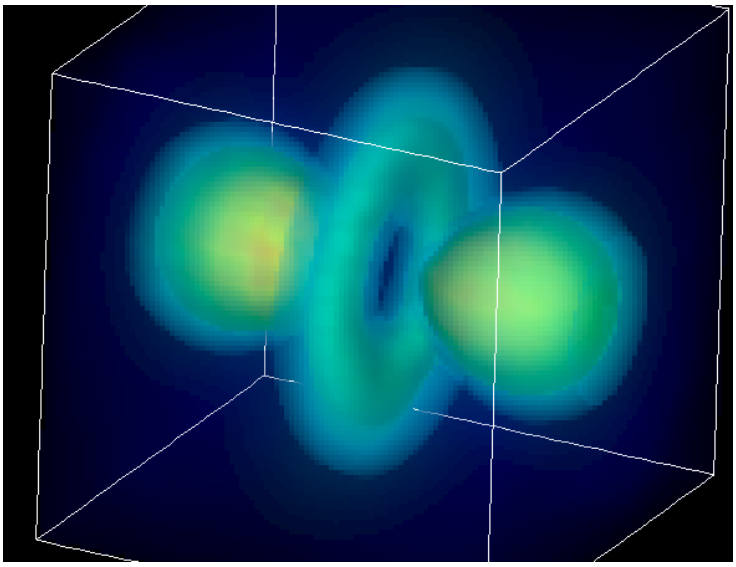

Fig. 4　ボリュームレンダリングによる電子雲の可視化

注意するべきは，可視化を目的するボリュームレンダリング法ではフォトリアルな画像を生成することよりも大量のボクセル群からなる大規模ボリュームデータを迅速に可

視化することが重視されたということである. したがってボリュームの媒質内部における光の散乱・それによる２次以降の吸光などは考慮されなかった. このようなボリュームの直接可視化に用いられるボリュームレンダリング手法のことをダイレクト・ボリュームレンダリングと呼ぶことも多い. 一方でコンピュータグラフィクスの分野においては関与媒体（participating media）内部の微粒子による光の散乱を考慮した物理的に正確なモデル化を行うことでリアルなガス・水蒸気・煙などのボリューメトリックな媒体や大理石・皮膚などの材質をレンダリングするための一手法としてのボリュームレンダリングが発達した(Fig. 5). 代表的なものに Max や Stam らの研究がある[11][12]. 関与媒体を用いたボリュームレンリングでは光エネルギーの伝達を物理的な観点から正確にモデル化したボリュームレンダリング方程式（LTE / Light Transport Equation）を解くことによってリアリスティックな画像を得ることが可能であるが, 計算時間や得られる画像品質のコントロールが難しいという点から一般的な可視化用途には適さない.

Fig. 5　ボリュームレンダリングによる Participating Media の可視化 [33]

１-２-２-２　等値面による可視化

等値面（Iso Surface）(Fig. 6)はスカラボリュームデータからユーザーが設定したある閾値に相当するデータ値をもつ連続面を抽出することによりソリッドレンダリングを行う可視化手法である. ボクセルデータに対する代表的な Iso Surface 抽出手法としてマーチングキューブ法がある[13]. ソリッドな面では法線が定義されているので仮想

的なライティング(陰影付け)によって３次元形状の把握が容易となるメリットがある.

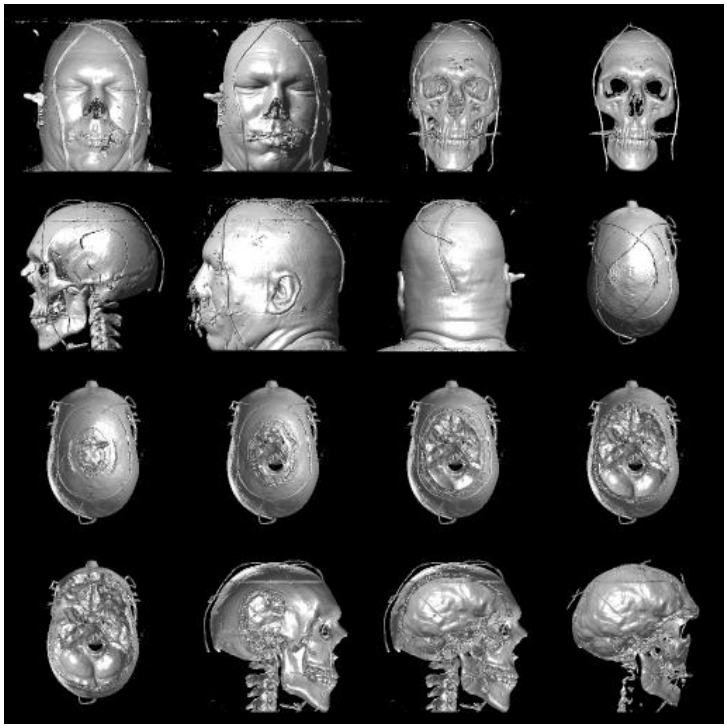

Fig. 6　Iso Surface による MRI 測定データの可視化［32］

１-３　Information Visualization

Information Visualization（情報可視化）はそれまでほぼ科学的データという対象に限定されていた可視化の概念をより一般的で幅広い情報・データへと拡張したものであり，１９９０年代初頭以降に発展を見せ始めた．情報可視化の目的は世の中にあふれる様々なデータ，特に電子情報などの目に見えない大量のデータに可視描像（アイコンなど）を与えることで可視化主体による情報探索を補助する，もしくは大規模情報に潜む大域的・局所的構造を抽出・認知させるといったことにある．特に近年ではコンピュータによる処理能力の向上・ネットワーク速度向上を受けた情報伝達の電子化・データストレージの大容量化といった要因を背景に，様々な大規模な電子情報が生成され・拡散され・また蓄えられるといった状況を生んだ．このような大規模電子情報を人間が認識・分析・把握するための情報視覚化技術は専門家のみならず一般市民の日常生活においてもその必要性は今後ますます高まっていくと思われる．

この情報可視化と呼ばれる分野の特長のひとつはその可視化対象の多様性にある．Scientific なデータに限定することなく地理情報・金融・インターネット上における検索結果やコミュニケーションなどあらゆる種類の情報を扱う．もうひとつの特徴として多種多様な可視化手法が存在し，新たに提案されつづけているという点が挙げられる．

情報可視化では先に述べた可視化対象データの多様性に加え，具体的な可視化目的・可視化を行う主体（専門家か一般ユーザーか）についても様々なケースが考えられる．またどのような表示・操作デバイスを用いるか（デスクトップＰＣか携帯電話か等）によっても適用可能な手法が異なる．さらにグラフィクスによって情報の大域的・局所的特性をユーザーに認識させる必要があるため一見して分かりやすく，なおかつ特徴的でインパクトのあるビジュアルデザインが求められる．以上の要求を満たすために情報可視化にはユニークなアイデアが盛り込まれた手法が多数存在する．本章では代表的な情報可視化手法および近年における情報可視化のトレンドについて概説する．

１-３-１　情報可視化の有効性

まず情報可視化の有効性とはどのようなものであるかを考えてみたい．すなわちなぜ情報を可視化することが重要なのかである．ここでは多くの現代人にとって身近ものとなったパーソナルコンピュータのデスクトップ画面を例に考察する．近年の多くの OS（オペレーティング・システム）はグラフィカルなユーザーインターフェース（GUI）を備えている．ファイルシステムの最下層をデスクトップ・メタファーで表現する手法は Apple 社の Macintosh によって一般に認知されるようになった．それ以前の OS では文字列によってファイルシステムをユーザーに通知するキャラクター・ユーザー・インターフェース（CUI）が主流であった．次の Fig. 7 に示す図は近年の洗練された GUI と CUI の比較である．

Fig. 7　デスクトップ GUI（左）と CUI（右）の比較

あるディレクトリ（コンピュータのファイルシステムの階層）にどのような種類のファイルがいくつ存在するのか，といった電子情報は本来目に見えないものであるが，これをアイコンを用いてグラフィカルに示す―すなわち情報可視化を行うことで当該ディレクトリに存在するファイルの情報をユーザーが直感的に把握できるようになる．これが情報可視化の典型的な効能である．デスクトップ画面の場合にはファイルのアイコンだけでなくアプリケーションを起動するためのショートカットやシステムの使用状況（システムメモリや CPU の占有状況）も可視化される．さてデスクトップ上のアイコンはそれぞれ形状やデ

ザインが異なり，そこにはアイコンが表すファイルの種類やアプリケーションの種類がデザインとして表現されている．すなわちこれらのアイコンが可視化しているのはそれらファイルやアプリケーションの存在だけではないのである．ファイルの種類，関連付けられたアプリケーション，アプリケーションの機能といった質的特長がアイコンとして可視化されており，それゆえユーザーはファイルアイコンやアプリケーションのショートカットを見ることで，それらを起動し作業を行うという行動を思いつく．すなわち情報が可視化されたことによって可視化主体にある種のアクションが促されるのである．例えばあるディレクトリのアイコンをマウスでクリックすればそのディレクトリの内容をアイコンで表したウィンドウがスクリーンに表示される．ショートカットをクリックしてアプリケーションを起動すればアプリケーションのウィンドウが表示され，そのウィンドウにはユーザーに行動を促すような操作アイコン（ボタン等）が配置されている．これらのボタン類もまたアプリケーションが備える機能をアイコンとして可視化したものであり，ユーザーに行動を想起させ，促すものである．このように可視化が行動を促し，それが実行された先には別の可視化があることでそれに続く新たな行動が想起される．可視化のチェーンが形成されることでユーザーは絶えず状況を認識し続けることが可能となり，連続的に作業を進めることができる．ここに情報可視化の有用性を見て取ることができる．

ここでユーザーが行動を実行する際にマウスでアイコンを指定しクリックするという行為を注意深く考えてみる．先ほど述べたようにこのアイコンはファイルの種類やアプリケーションを図形として示し可視化したものである．つまりこれらアイコン自体が可視化結果の一部であり，それをクリックして各種行動を実行するということは可視化結果そのものをユーザーインターフェースとして活用しているということである．このように情報可視化とユーザーインターフェースが対の関係をなすことがある．これもまた情報可視化のもたらす特性のひとつである．

1-3-2　階層データの可視化

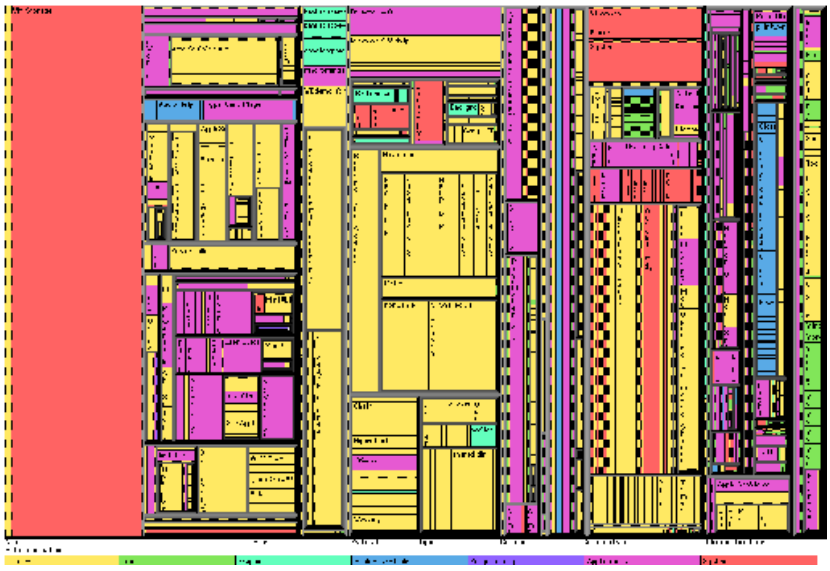

Fig. 8　Tree Map によるファイルシステム構造の可視化［16］

ディレクトリのような階層的構造をもつデータを効率的に可視化する試みは情報可視化の初期の段階から行われていた. Johnson らの提案した Tree Map(Fig. 8)は２次元平面を矩形に分割していくことで階層構造を可視化する手法である[16]. Tree Map では各矩形アイコンが階層データにおける末端ノードを表す. コンピュータのファイルシステムを可視化した場合を例にとれば各矩形はファイルもしくはディレクトリに相当する. Tree Map では矩形の大きさと色によって２種類の特徴量（attributes）を表現することができる. 従って矩形の大きさによってファイルサイズを表現し色によってファイルの種類（拡張子別など）を表現することが可能である. このような一覧可視化手法をとることで階層データの全体像を概観することが可能であり，特徴的な attributes をもつノード・局所領域も同時に認識することができる.

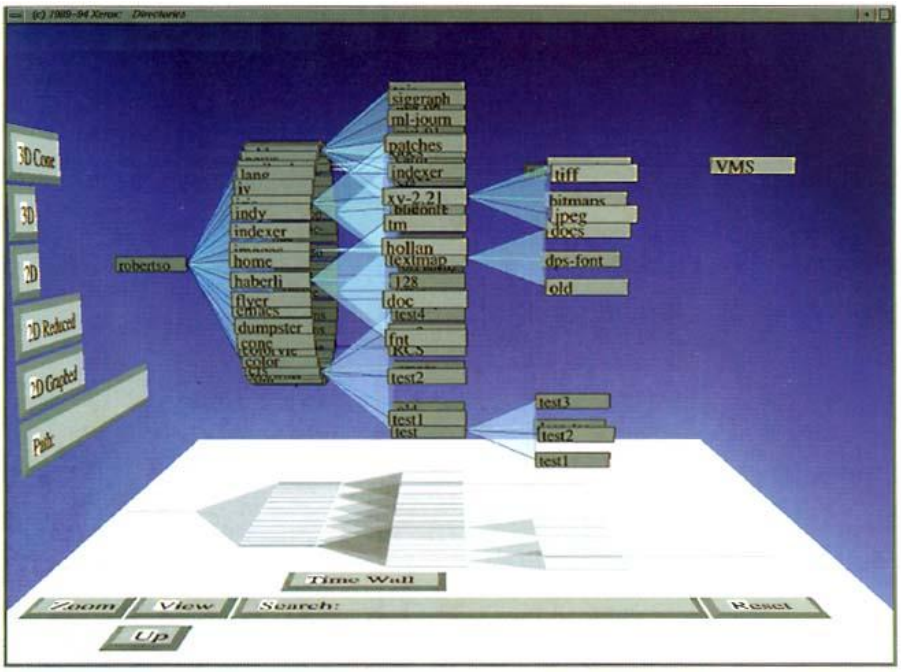

Fig. 9　Cone Tree による階層データの 3 次元可視化［17］

George らの Cone Tree(Fig. 9)は階層構造を 3 次元的に可視化する手法である[17]. Cone Tree は樹形図を 3 次元に拡張したものであると考えることができる．各ノードはアイコンによって表現され，子ノードは親ノードから派生する．伝統的な樹形図と異なる点は親子関係にあるアイコンのレイアウトが 3 次元的なコーン（円錐）を形成するという点である．3 次元的な表示方法をとることで大量のデータを有限の画面領域内に収めることができる．一般にこのような 3 次元的な表示を用いて大量のアイコンを表示した場合仮想的なカメラの奥方向にあるアイコンは手前にあるアイコンに遮られて可視性が損なわれる（occlusion）という問題が生じる．これに対し視点移動や可視化対象（オブジェクト）の回転操作を施して再可視化することで，それ以前の状態では不可視であったオブジェクトが視認可能となる．また，パースペクティブ・ビューを用いた場合画面奥方向のオブジェクトほどスクリーン上では小さく描画されることになるが一方で大量のオブジェクトを表示可能となるため結果的にデータを概観し把握する上では利点となるという指摘がなされている．

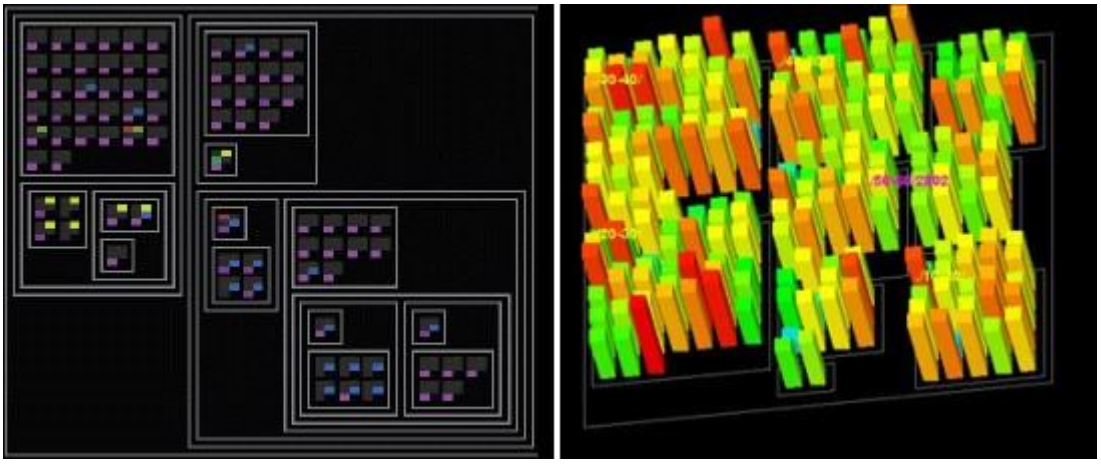

Fig. 10　2 次元版（左）および 3 次元版（右）の平安京ビュー [15][18]

Itoh らの平安京ビュー（Heiankyo View）(Fig. 10)は平面上に矩形アイコン配置し階層データを表現するという点はTree Map と同様であるがアイコンの配置を最適化する点が異なる[15][18]．末端ノードは矩形アイコンで表され，末端ノードの属する親ノードは末端ノードを取り囲む四角い枠で表現される．この表示手法をとることで，ある親ノードにいくつの子ノードが属しているか（ファイルシステムの例ではあるディレクトリにいくつのファイルや子ディレクトリが含まれているか）の把握が容易となる．またノード配置の最適化計算を行うことにより限られた表示領域内に多数のノードを効率的に配置する手法を提案している．Tree Map の充填率には劣るものの枠・矩形アイコンの配列に適度な間隔を与えることでノードの個数や包含関係が認識しやすくなるという好例である．平安京ビューは３次元への拡張も行われている．３D 版の平安京ビューではアイコンとして直方体を用いる．この場合アイコンの高さと色により一度に２種類の attributes を表現可能である．

１-３-３　Focus + Contextを利用した可視化

Focus＋Context は情報可視化において頻繁に取り上げられる概念である．ユーザーの注視領域（Focus）における局所的な詳細情報と大局的な概観像（Context）という異なるスケール・抽象度における可視化像を同時に提示する．Focus 部を前景とするなら Context は背景にあたる．Focus＋Context の利点は常に大域像を把握しながら局所的詳細像を探索することができる点である．また，大域表示（Context）を眺めることを介してユーザーが次に注目すべき領域（Focus）を発見するという可視化のワークフローを実現することも可能となる．

Furnas らが 1981 年の論文にて発表した Fisheye View は Focus＋Context の概念が取り入れられた最初の例である[19]．Perspective Walls，Table Lens，Hyperbolic Browser など様々な可視化手法において Focus＋Context が導入されている

[20][21][22][27][29]. これらの手法に共通する点はいずれも空間構造を意図的に歪めることによって局所詳細表示と全体像の概観表示の共存を可能にしている点である. これは 3 次元実空間における物理量分布の可視化および把握を目的とする Scientific Visualization の分野ではほとんどみることのできない特徴である. 次に示す Fig. 11 は Hyperbolic Browser による可視化図である。

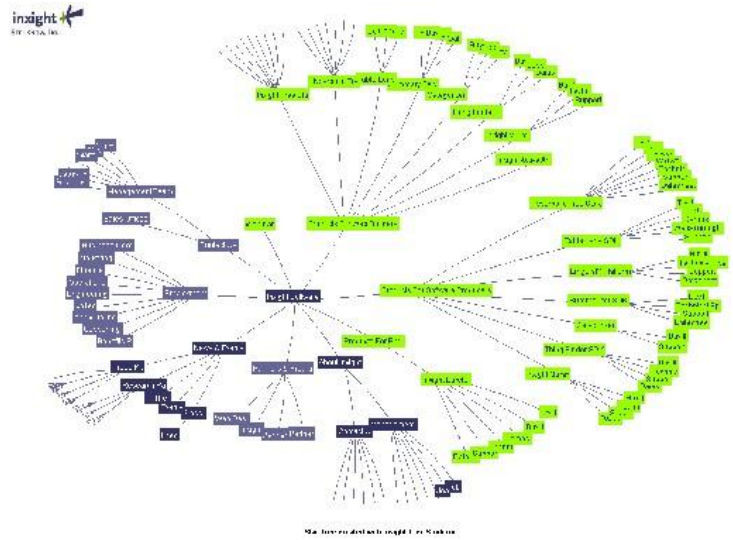

Fig. 11　Hyperbolic Browser による Focus + Context 可視化 [20]

1-3-4　フラクタルの概念を用いた可視化

フラクタルの概念を直接的に情報可視化に取り入れた例は希少である. Koike らの Fractal Views(Fig. 12)は我々が確認できた唯一の事例である[14]. Fractal Views では多段階層構造に自己相似的構造が多く現れることに着目しフラクタル的重要度の概念を導入して可視化されるオブジェクトのカリングを行うことで表示される情報量をほぼ一定に保つアイデアが提案されている. 当該手法はフラクタルの概念を導入することでマルチ解像度可視化が実現可能であることを明確に示した例であるといえる. 本研究ではより具体的にフラクタル図形の自己相似構造を用いたマルチスケール・マルチ解像度の可視化の実現方法を提案する.

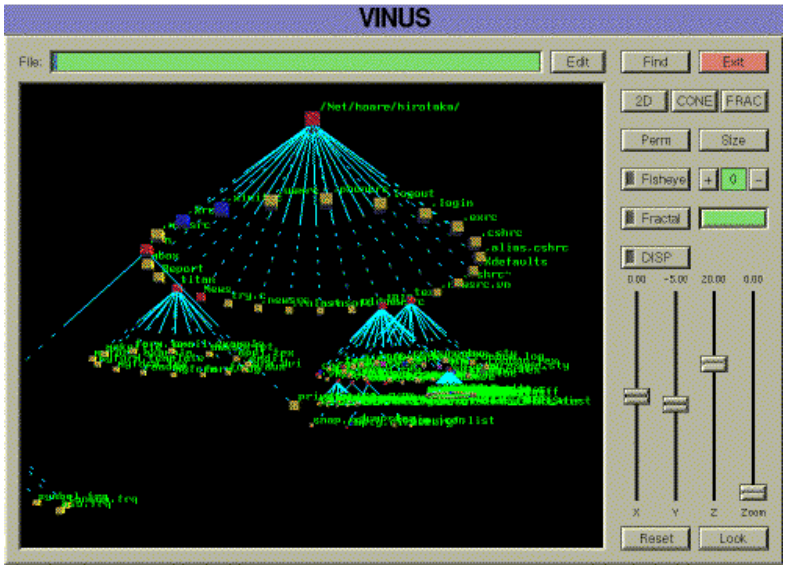

Fig. 12　Fractal Views による可視化 [14]

1-4　既存手法に関する考察

1-4-1　科学的可視化と情報可視化の比較

様々な可視化手法を調査する中で科学的可視化・情報可視化双方の傾向とその相違を見出すことができた. 既に述べた内容と重複する部分もあるがここにそれらを列記する. 本研究における提案手法を開発するにあたりこれらの相違点を考慮することは大変参考になった. 特に代表的な相違と考える 4 点を以下に述べる.

相違点のひとつは情報可視化の多様性である. 科学的可視化がその対象を物理量に限定しているのに対し情報可視化が対象とするのは有象無象あらゆる種類の情報である. 近年におけるコンピュータの処理能力向上やインターネットの普及・発達はデータの大規模化のみならず多様化にも寄与した. その結果可視化対象となる様々な情報が氾濫し結果として情報可視化の必要性を高めた. また同様のことが可視化手法についても言える. 科学的可視化がいくつかの限られた手法を洗練させてきたのに対し情報可視化では特徴も構造も多種多様な情報を可視化するために無数の可視化手法が提案されてきた[25].

2 つ目の相違点として挙げられるのは科学的可視化の専門性の高さである. 前述した通り物理現象の解明や自動車などのプロダクトデザインの評価などを目的とする科学的可視化を行う主体は専門性の高い研究者や研究職に属する場合がほとんどである. 従って可視化手法もそのような研究を専門とするユーザーに向けたものとなっており使いこなすためにある程度の科学知識とトレーニングを必要とするものが多い. 一方で情報可視化には一般ユーザーによる利用を想定したものも見受けられる. 検索エンジン

と組み合わせた Web サービスという形式で提供されているものが多く，身近な例を挙げれば現実の地図上に様々な情報をオーバーラップさせて表示する Google Map(Fig.13)などもその一種であると言える[31].

Fig. 13　Google Map [31]

３つ目の特徴としてインタラクションに対する考え方の相違がある．科学的可視化におけるインタラクションは可視化像の調節が主目的である．例えばボリュームレンダリングや３次元的な流線の可視化においては視点の移動・回転操作により可視化領域を変更し，データの特徴を捉えた画像を生成することがユーザー操作によるインタラクションの目的である．もちろん情報可視化にも同種のインタラクションは存在する．そのうえで情報可視化の場合可視化結果にアクセスすることによってより詳細な情報を得るという目的のインタラクションも多く見られる．これは可視化されたアイコンをマウスでクリックするなどしてオブジェクトを指定し，そのノードに関してその時点では可視化結果に現れていない詳細情報を取得するといったものである．あるいは情報を取得するだけでなく対象オブジェクトに対して何らかの操作を実行する，ファイルシステムの可視化を例に取るとアイコンをクリックすることで当該ディレクトリがエクスプローラー上で表示されたり，当該ファイルを規定のアプリケーションで実行されたりするなどといった操作が可視化結果を用いることで可能となる．すなわちここでは可視化結果そのものを一種の入力インターフェイスとして用いている．先に挙げた通り Windows を始めとする近年のオペレーティングシステムが備える GUI によるデスクトップ画面とそれに対するマウス操作はこの種の可視化・インタラクションの好例である．

４つ目の相違点は空間の用い方である．ここではより違いが明確となる３次元可視

化を例に取る．ボリュームレンダリングや流線描画，Scatter plot などの科学的可視化手法では可視化空間はほぼ例外なく現実の物理空間と同様の３次元距離空間（距離を３軸にとる空間）である．これは科学的可視化が「人間の目に不可視な物理量の空間分布や連続性をもし見ることができたとしたらどのように見えるか」という思想に基づいているためである．一方で情報可視化では３次元空間における軸は必ずしも物理的距離を意味するとは限らない．様々な種類のデータを対象とする情報可視化では価格・人口などの様々な定量化可能な指標を各軸にとることも行われる．こうした本来性質の異なる指標により構成された３次元空間は非等方的な仮想的空間であり情報空間（information space）と呼ばれることもある．このような仮想情報空間を用いた３次元情報可視化は３次元科学的可視化手法を情報可視化に取り入れたものというよりもむしろ一般的に用いられる２次元グラフを３次元に拡張したものだと考えるべきである．

科学的可視化と情報可視化を結びつける試みもまた近年行われている．我々は前年度に階層的情報可視化手法である平安京ビューを用いて原子力発電所の稼動シミュレーションにおける冷却材の流量をモニタリングするシステムを構築した．冷却材の流量は物理量であるがそのモニタリングに平安京ビューを活用することで科学技術分野における情報可視化手法の適用を試みた．本研究では主に科学的可視化の対象であるボリュームデータに対し，情報可視化的発想を取り入れた独自の可視化手法を提案・適用する．

１-４-２　２次元可視化と３次元可視化の比較

本項では２次元可視化手法と３次元可視化手法それぞれの持つ特徴・長所・短所について考察する．科学的可視化では現実世界における物理量分布が主な可視化対象でありその分布は本来３次元的である．従ってその分布をありのままに再現するためには３次元可視化手法を用いる必要がある．ボリュームレンダリングや３次元空間における流線・Scatter Plot などはその例である．しかしながら可視化結果を表示するデバイスは一般に２次元的な平面ディスプレイであるため描画にあたって３次元空間からスクリーン空間への投影（Projection）を行わなければならない．すなわち得られる可視化像は物理量の３次元的分布をある視点から眺めた場合の可視化像にすぎず，その投影の際に視点依存性と遮蔽（Occlusion）という２つの問題が生じる．ボリュームレンダリングではレイキャスティングによりボリュームデータに格納されたデータ値を２次元スクリーンへ投影するが，視点から光線（Ray）を放つレイキャスティングは視点依存手法であり，視点の位置と方向によって得られる可視化像は大きく変化する．従ってボリュームデータに記録された３次元的形状やデータ値の分布・連続性を正しく把握するためには視点移動・回転といった操作により様々な角度から対象を眺める必要がある．また，ボリュームレンダリングは半透明表示を用いた内部構造の可視化が可能ではあるが，場合によっては視点の反対側やボリュー

ム内部に存在するデータが可視化結果に現れにくいという遮蔽の問題も発生する．この問題を解決するためには視点を移動だけでなく可視化に関わるいくつかのパラメータを調節しながら再度レンダリングを繰り返すことも必要となる．流線を用いた３次元可視化についても同様の問題が発生する．特に多数の流線を描画する際には複数の流線同士が交差しオーバーラップしてしまい流れ場の正確な把握が困難になるという問題が多々発生する．この問題を軽減するために視点に応じて流線描画を最適化する Image based な手法が提案されているものの３次元可視化に付随する視点依存性がもたらすコストの問題は依然として残る[8]．一方２次元可視化手法を用いた場合，ある 2 次元断面上のデータ値を平面のディスプレイに投影するため原理的に遮蔽は発生しない．また視点の移動・回転を行う必要がない．そのため一般に２次元可視化は３次元可視化よりも認知負荷が低くヒューマンフレンドリーな手法である．ただし可視化されるデータは 3 次元データ全体のうちある一断面上のデータ値にすぎないため一枚の画像のみで物理量の 3 次元的分布を確実に把握することはできない．

情報可視化は仮想的な情報空間上での可視化であるから，３次元可視化手法を使う利点は 2 次元可視化よりも１つ多くの軸を指標として活用できる点にある[17][25][28]．また透視投影変換を用いることで視点から遠くにあるアイコンはスクリーン上で小さく表示されるので近景のアイコンがより強調され，遠景にはサイズの小さなアイコンを大量に並べることができる．このように３次元情報可視化は大量のアイコンを３種のパラメータからなる３次元情報空間に配置することで２次元可視化では認識することのできなかった特徴を浮き彫りにすることを目的とする手法である．ただし対象とするデータが大規模化し表示するアイコンが増加するに従って３次元可視化特有の視点依存性と遮蔽の問題が顕著になってくる．前述の Cone Tree などの３次元可視化手法は大量のアイコンが描画される際互いに重なり合ってしまい半透明表示を用いても前後関係の把握が困難となる．また表示デバイスはあくまで２次元のディスプレイでありマウスによる操作も２次元的であるためこれらのデバイスを利用してユーザーが３次元空間上に描画された特定のアイコンを選択する際のインタラクション・コストは比較的大きい．一方で Tree Map のような２次元可視化手法は２次元ディスプレイ上で平面的表示を行うため原理的に遮蔽が発生せず視点依存性もない．またアイコンの選択もマウスによる２次元的な操作で十分である．一般に３次元可視化手法に比べ空間充填率が高いことも特徴である．すなわち２次元可視化は３次元可視化と比較して，可視性・アクセシビリティ・空間充填率の点で優れる．Smallman らは情報探索のしやすさという観点から 2 次元表示と 3 次元表示を比較した[26]．その結果，2 次元表示の方が情報を素早く探索できるうえミスも少ないということが報告されている．彼らの論文によれば情報探索の観点からは 3 次元表示は重要ではなく，効率良くデザインされた 2 次元表示を用いるほうが有用である．

２次元可視化と３次元可視化を併用する試みも多く行われてきた．Tory らは表示手法により 3 次元空間における相対的な位置関係の把握しやすさがどのように異なるかをユーザ

ーテストにより比較評価した[23]. その中で 2 次元可視化図と 3 次元可視化図を同一画面内に表示する ExoVis という新しい手法を提案し評価している(Fig. 14).

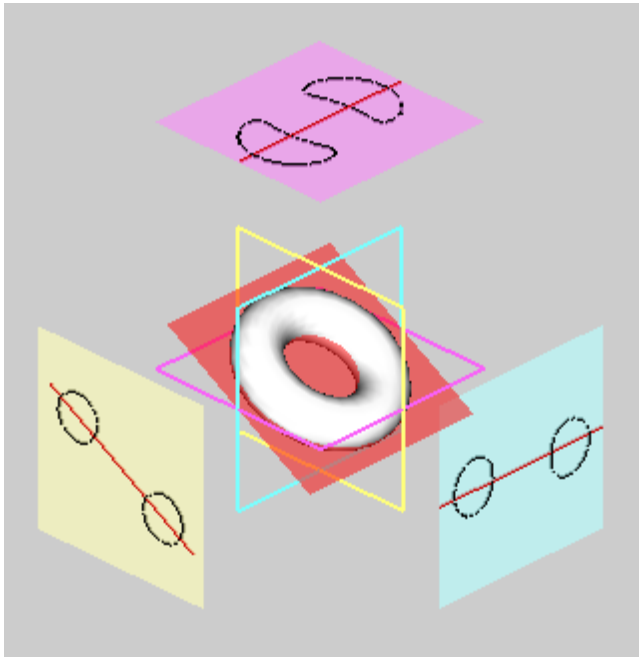

Fig. 14　Tory らの ExoVis [23]

彼らの行ったユーザーテストの結果としては 3 次元表示に影表示などの Depth Cue を加えたものが相対位置関係のおおまかな把握には最も有効であったと報告されている. ただし 3 次元表示は距離や座標を高精度で識別・指定するようなタスクには不向きであった. ExoVis のような 2 次元表示と 3 次元表示を組み合わせた手法は正確な位置関係の把握や座標指定において有効であるとされた.

Piringer らは 2 次元 Scatterplot（散布図）と 3 次元の Scatterplot を組み合わせることで遮蔽の問題を軽減する手法を提案した[24]. ユーザーが 2 次元 Scatterplot 画面に対して行った操作は 3 次元 Scatterplot 画面に反映される. すなわち 2 次元的な操作が 3 次元的な表示へとフィードバックされるのである. 彼らはこの手法にさらに Focus+Context の概念を取り入れることで大規模なデータの空間分布把握を容易にした(Fig. 15).

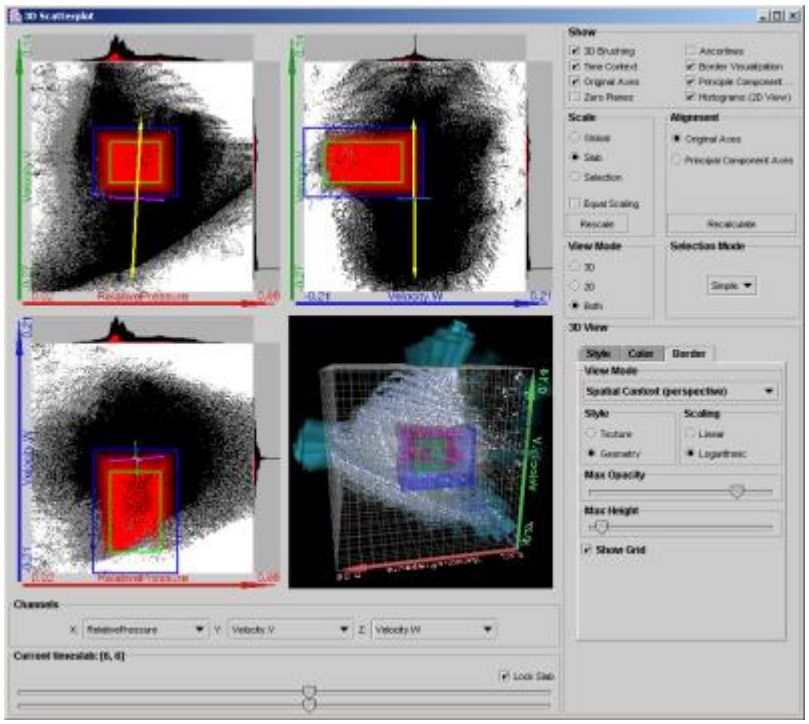

Fig. 15　Piringer らの開発した 2D+3D Scatterplot システム [24]

1-5　本研究の位置付け

ボリュームデータに適用可能な科学的可視化手法は限られており，さらに可視化精度や遮蔽性・視点依存性などの面から単体の手法による可視化結果のみでデータの特性すなわち現象として何が起こっているかを判断することは危険である．したがって求められているのはボリュームデータの多角的可視化を実現する手段であり，そのためには従来の可視化手法とは異なる観点からボリュームデータ可視化・閲覧を実現する手法を開発する必要性がある．そこで本研究では情報可視化的発想を取り入れた新しい可視化手法を提案・開発する[1][2][3][4]．またそのような手法は 3 次元可視化の問題点である視点依存性・遮蔽性を克服するものでなければならない．我々は提案手法を視点依存性・遮蔽性のない 2 次元可視化とすることでこれらの欠点を克服し，既存の 3 次元可視化手法と組み合わせた際に多角的なデータ可視化を実現することを目指す．さらにフラクタル図形を可視化に活用することでボリュームデータのマルチレゾリューションな可視化を実現する．次章ではこの提案手法の詳細について解説する．

第２章　手法

２-１　方針

本章では我々の提案手法である情報可視化的発想を取り入れた新しいボリュームデータ可視化手法を解説する[1][2][3][4]. ここでは手法の基礎となる概念や手順の説明に重点を置き，アプリケーションとしての実装および実際のデータへの適用は次章にて行う．本手法を開発するにあたり前章における既存の科学的可視化・情報可視化両手法の特徴および２次元可視化・３次元可視化両手法の長所短所に関する議論を踏まえ開発方針を定めた.

1) 情報可視化的発想を取り入れた新しいボリュームデータ可視化手法の開発
2) 提案手法は遮蔽・視点依存性のない２次元可視化手法
3) ボリュームデータの一断面のみでなく全域を２次元的に表示可能な手法の開発
4) ボリュームデータ構造を階層データとみなし階層的情報可視化を行う
5) フラクタル図形を活用したマルチスケール可視化手法の開発
6) ２次元可視化結果をボリューム全域の地図として活用
7) ２次元可視化結果を入力インターフェースとして活用
8) 提案する情報可視化的手法と既存の科学的可視化手法との相互運用
9) 提案する２次元ベースの表示手法と既存の３次元手法とのコラボレーション

以上の方針に基づき新たな可視化手法を開発した．本章では 1)～5)の方針を踏まえた新しい可視化手法を提案し，次章では 6)～9)の方針を踏まえた実装を行う.

２-２　概念

本手法を構築するために必要となる概念を次に紹介する．まずは可視化対象となるボリュームデータに階層性を定義しそこに Octree 構造を見出す必要がある．また本手法はフラクタル図形を用いた新しい可視化手法であるためシェルピンスキーのカーペットと呼ばれるフラクタル図形についてその概念および実体を理解する必要がある.

２-２-１　ボリュームデータ

まずは本手法の可視化対象となるボリュームデータについて説明する．ボリュームデータは温度・圧力・速度といった物理量（スカラー・ベクター量）の３次元空間における分布を記録したものであり，主な可視化手法としてボリュームレンダリングや流線がある．連続空間における物理量の分布を有限のデータ量で表現するために空間にサンプリング点

を離散的に配置しそれらの点における物理量をサンプリングし記録する．数値シミュレーションではこのサンプリング点により構成される格子をそのまま計算格子として用いる場合が多い．このサンプリング点を密に配置するほど記録されるデータの精度が向上し高解像度のボリュームデータとなるが一方でデータサイズは増加する．一般的な X-Y-Z 直交座標系を用いて軸に沿ってサンプリング点を配置する場合精度向上のため３軸方向のサンプリング点密度をそれぞれ２倍にするとデータ量は８倍となってしまいデータサイズが爆発的に増加するという傾向がある．このようにサンプリング点を座標系に沿って規則正しく並べたデータ構造は構造格子と呼ばれる．X-Y-Z 直交座標系における構造格子の場合，データ値を格納した各サンプリング点を 6 面体とみなすことができこれがボクセル呼ばれる．一方でサンプリング点および計算格子が座標系に沿って並んでいない場合これを非構造格子と呼ぶ．代表的なものに六面体・四面体・三角柱などからなる非構造格子が存在する．サンプリング点間隔および計算格子の配列が均等でなくともよいためあらかじめ高い精度・解像度が必要と思われる部位にサンプリング点を密に配置し，計算時間とデータサイズの増加を抑えつつ必要なデータ精度を得ることが可能である．提案手法では６面体構造格子のみを対象とするが，非構造格子からなるデータであっても軸に沿って適切なサンプリングを行うことで構造格子へと変換することが可能である．ただし変換後のデータの解像度はサンプリング間隔に依存する．

２-２-２　ボクセルの階層性

本提案手法は６面体ボクセルの均等な配列からなるボリュームデータを対象とする．その理由はボクセル構造に階層性を見出すことができるためである．物理量の分布をより正確に表現するためにはサンプリング点を密に配置したデータ―すなわち細かいボクセルを用いる必要がある．そこでボクセルの再分割という考え方を導入することでボクセルの解像度を次のように定義する．初期状態を単一のボクセルとする．これを解像度０のボクセルとする．各ボクセルを X-Y-Z 各方向に２等分し８分割することによって解像度が１上昇することとする．この分割操作をボクセルの再分割と呼ぶこととし，この再分割を繰り返すことによって高解像度のボクセルを生成するものとする．この手続きの様子を以下の Fig. 16 に示す。

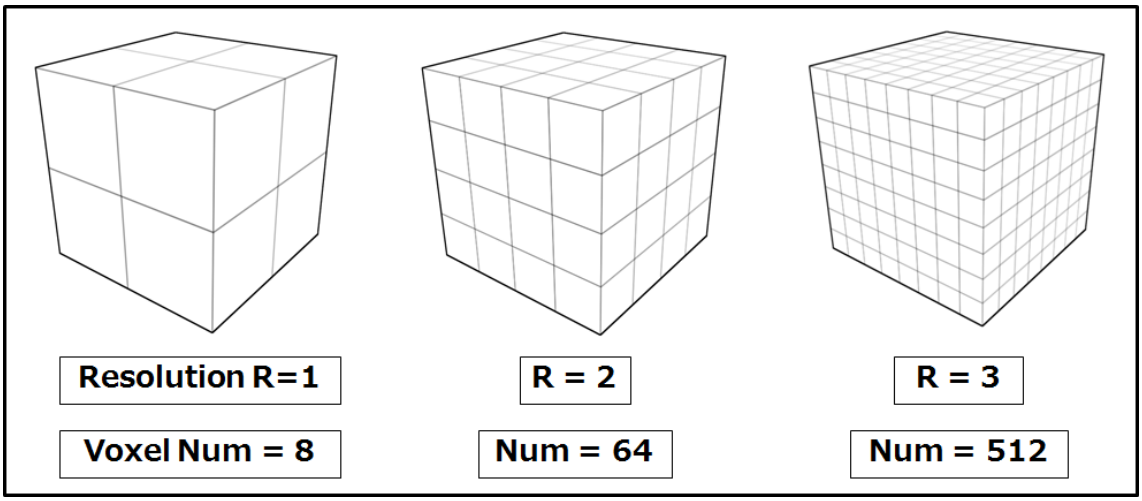

Fig. 16　R＝１・２・３のボリュームデータ

このとき解像度 1 のボリュームデータには 8 個のボクセルが, 解像度 2 のデータには 64 個のボクセルが，解像度 3 のデータには 512 個のボクセルが含まれる．一般化すると，ある解像度Rのボリュームデータには 8^R 個のボクセルが含まれる．従ってこのボクセルの再分割によるデータ構造は典型的な８分木（Octree）構造となる．

本手法において解像度Rのボリュームデータを構成する各ボクセルのことを解像度Rのボクセルと呼称する.解像度 R のボクセルの各辺の長さは解像度 R+1 のボクセルの 2 倍(体積にして 8 倍）である.

２-２-３　シェルピンスキーのカーペット

続いてシェルピンスキーのカーペット（Sierpiński carpet）（以下ＳＣと略記）と呼ばれるフラクタル図形について述べる．この図形は Wacław Sierpiński によって 1946 年に提唱された図形であり以下のような再帰的なアルゴリズム(Fig. 17)によって描画可能な２次元フラクタル図形である.

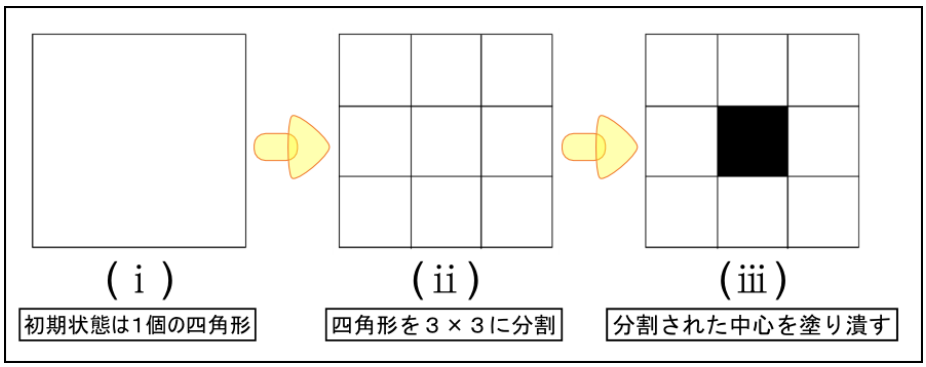

Fig. 17　ＳＣの描画アルゴリズム

(i)　初期状態は単一の矩形とする
(ii)　各矩形を縦横 3 等分し 9 分割する
(iii)　分割した 9 つの矩形のうち真ん中の矩形を取り除く
(iv)　(ⅱ)～(ⅲ) の手順を残った矩形に再帰的に適用する

提案手法では(ⅱ)～(ⅲ) のプロセスをボクセルの再分割と対応させる意味でＳＣの再分割と呼ぶこととする. 本来ＳＣはこの再分割プロセスを無限に適用した概念上の図形であり有限回の操作では完璧なＳＣを描画することはできないが, 本手法においては再分割を有限回適用した図形についてもＳＣと呼称することとする. ここでボクセルの場合と同様にＳＣの解像度を定義する. (ⅰ)の状態を解像度０のＳＣと定義し，再分割プロセスを１回適用するごとにＳＣの解像度が１上昇するものとする.

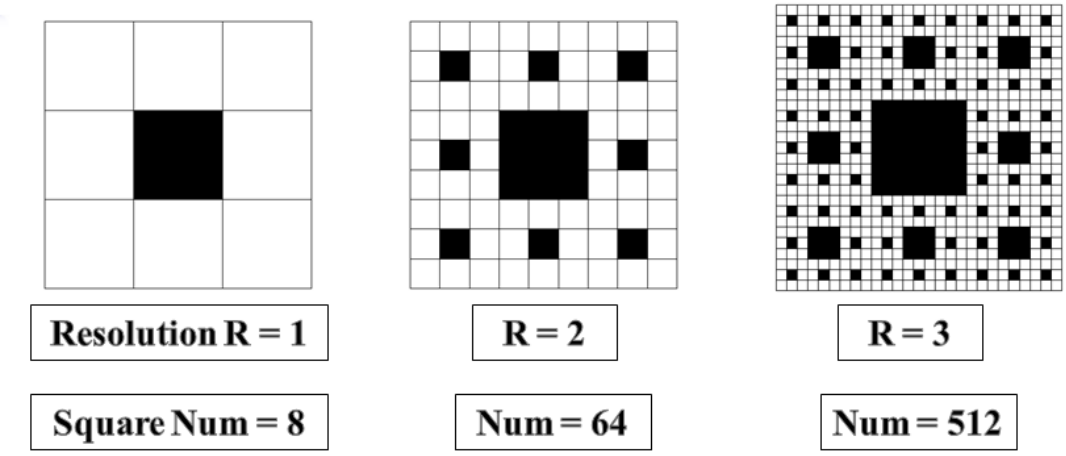

Fig. 18　Ｒ＝１・２・３のＳＣ

この分割アルゴリズムに従うとき，解像度 1 のＳＣには 8 個の矩形が，解像度 2 のＳＣには 64 個，解像度 3 のＳＣには 512 個の矩形が存在する．一般化すると解像度ＲのＳＣには 8^R 個の矩形が存在することになる.

我々はこの 8^R という数字の一致に着目した．解像度Ｒのボリュームデータに含まれるボクセルの個数と，同じく解像度ＲのＳＣに含まれる矩形の個数は一致する．そして再分割ボクセルからなるボリュームデータの Octree 構造をフラクタル図形の自己相似構造と結びつけることによって，３次元的なボクセルデータを２次元平面上に展開することが可能であると考えた．次項ではこの展開手法について説明する.

2-3　ボクセルの２次元展開

本手法では解像度Ｒのボリュームデータに存在する 8^R 個のボクセルを解像度 R のＳＣに含まれる 8^R 個の矩形領域に２次元的に展開し，当該ボクセルのデータ値を色として表現することで可視化が完了する．とはいえ 3 次元形状をもったボクセルを 2 次元の矩形に展開するという発想は本手法特有の新規なものであり，まずはその概念を説明する必要がある．前項にてボクセル構造を階層的に捉え，そこに Octree 構造を見出した．ボクセルの 2 次元展開が実現可能であることはその構造，すなわち Octree 構造が SC 上に展開可能であることによって保証される．それではなぜ Octree 構造は SC 上に展開可能であると言えるのか，次に示す Fig. 19 はその理由を明確に示している．

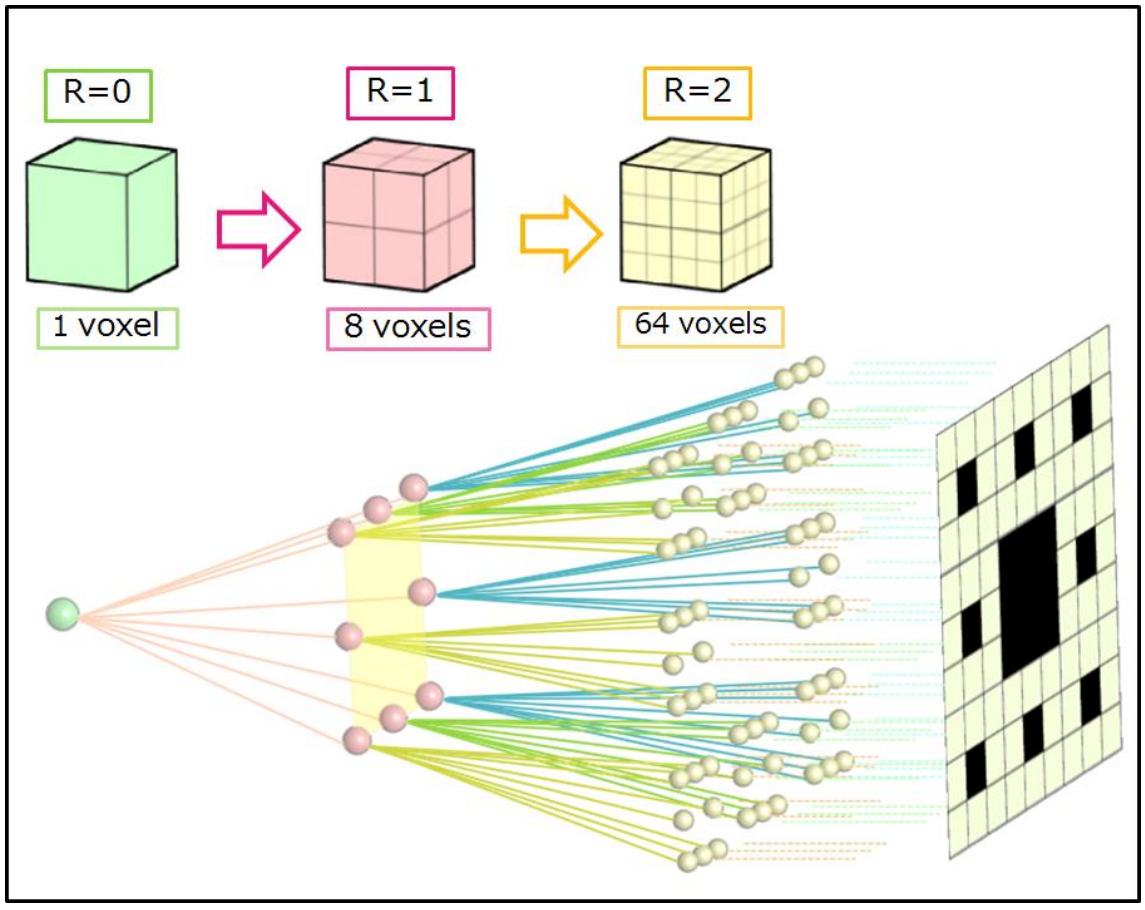

Fig. 19　ボクセルは Octree 構造であり Octree 構造は SC に展開可能である

このように Octree 構造の各末端ノードは SC の各矩形に 1 対 1 に展開可能である．それが可能である理由は上図からもわかるように SC の中にも Octree 構造を見出すことが可能だからである．ボクセルの Octree 構造と SC の Octree 構造を結びつけることによって初めてボクセルの 2 次元展開が可能となる．しかもこの展開は双方向性をもち，

逆に SC の各矩形もまた Octree の各ノードすなわちボクセルに転写可能であるとも言える．ここではボクセルと矩形の 1 対 1 の対応が成立している．ここにボリュームデータの各ボクセルが SC の矩形に展開可能（対応付け可能）であることが示された．

しかしながら Octree という概念としての構造が SC に転写可能であることを示しただけでは可視化手法として用いるには不十分である．Octree の各ノードが SC のどの矩形に展開されるのかという対応関係が具体的に定義されて初めて実用上の価値をもつ．この対応関係が規則性をもって規定されなければどのボクセルがどの矩形に展開されるかを演繹によって求めることができない．次段ではこの展開ルールを定める．

２-３-１　ボクセルの基本展開ルール

さてどのボクセルがどの矩形に対応するかを定義するための展開ルールを定めなければならない．本手法では R=1 のボリュームデータの各ボクセルと R=1 の SC の各矩形の対応関係を記述した基本展開ルールをまず定義し，より高解像度のデータに対してはこの R=1 において定めた展開ルールを再帰的に適用することで任意の解像度のボリュームデータを等しい解像度のＳＣ上に展開することを可能とする．

さてまずは R=1 における基本展開ルールを定める必要がある．このルールはボクセルと矩形が 1 対 1 の対応している限り任意であり様々なパターンが考えられる．性格には 8 個のボクセルと 8 個の矩形のマッチングパターンであるから基本展開ルールの候補は8! = 40320通り存在する．この中からどの展開ルールを採択するかによって展開の結果すなわち可視化結果は変化する．ゆえに合理的な基本ルールを選択することは大変重要である．我々はこの基本展開ルールを決定するにあたって以下の Fig. 20・Fig. 21 に示す２種のルールを最終的な候補とした．

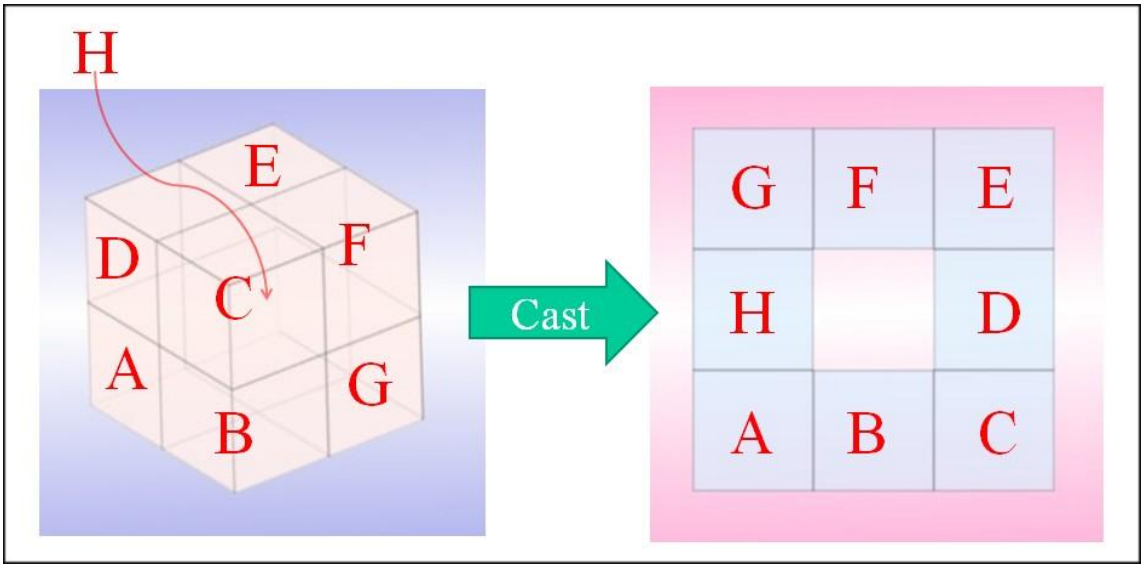

Fig. 20　Ｒ＝１の展開ルール候補［Ｉ］

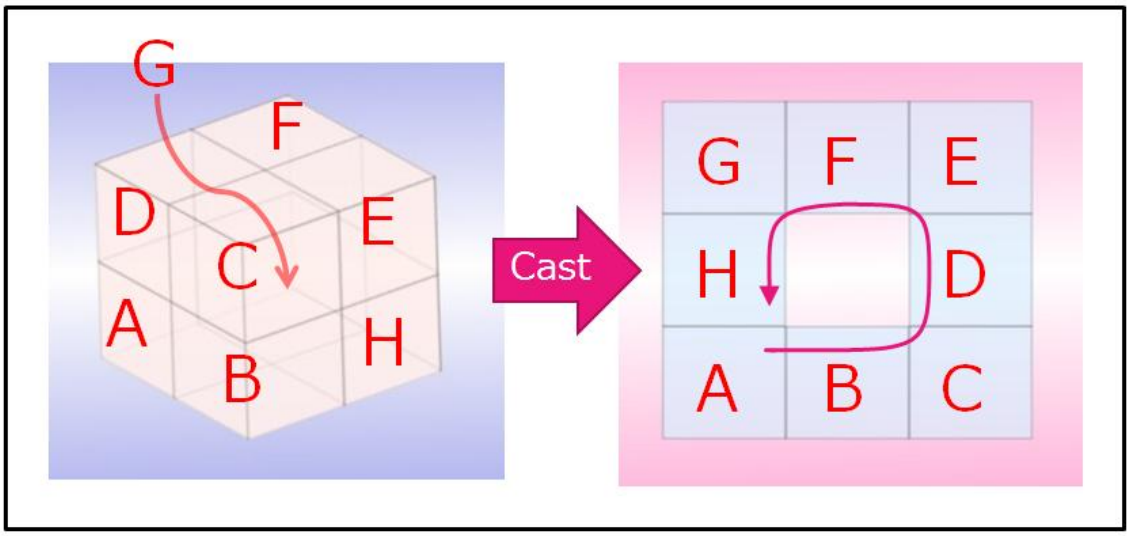

Fig. 21　R＝1 の展開ルール候補［Ⅱ］

上の対応表の見方を説明する. はじめに各ボクセル･矩形に A～H までの名称をつけた. 同じ名前のボクセルと矩形が対応することを表している. 例えばボクセル A は矩形 A に対応し, ボクセル A に格納されたデータ値は矩形 A の位置に色として可視化される. 続いて 2 つのルール[Ⅰ]・[Ⅱ]の特徴と相違点を説明する. どちらも左下のボクセル・矩形を A に割り当て対応させた. また SC 側では矩形 A から反時計周りに A~H の名称を割り当てた. ここまでは A・B 双方に共通する特徴である.

ルール[Ⅰ]ではボクセル同士の隣接関係を SC 展開後も極力保つことを心がけている. この方針をとる理由は隣接関係を保つことで 3 次元的に値の連続した領域は結果的に連続した矩形に展開され, ボリュームデータにおけるクラスタ構造を展開後においても 2 次元的なクラスタとして可視化することが達成されうると考えたためである. そしてこれを実現するためにルール[Ⅰ]では 8 個のボクセルを一筆書きの要領で 1 周できるように名前を付けている（A→B→C→D→E→F→G→H→A）. SC 側でも反時計周り順に矩形の名前をつけたため同様に一筆書きの要領で 1 周できるようになっている. このときボクセルの隣接関係のうち最低でも 2 つが展開後も保たれることが保証される. また SC 上で隣接する矩形同士は対応するボクセル同士も隣接することが保証される. ルール[Ⅰ]の SC において矩形 A と隣接する矩形は B と H であるが展開前のボクセルにおいてもボクセル A・B・H はそれぞれ隣接関係にある. ただしボクセル A のもうひとつの隣接ボクセルであるボクセル D は SC 上では矩形 A と離れた位置に展開されてしまう. R=1 の SC 上ではある矩形に隣接する矩形は 2 個のみであるため R=1 ボクセルでの隣接関係のうちひとつは損なわれることになってしまう. すなわちこの基本展開ルールを用いても隣接関係を完璧に保つことはできない.

ルール[Ⅱ]を定めるにあたり採用した方針はボクセル同士の対角関係がSC上において完全に保たれるようなルールを考案するというものである. R=1のボクセルを見たとき各ボクセルにとってボリューム中心の点に対して点対象な位置（対角位置）には別のボクセルが存在する. 例えばFig. 21においてボクセルAの対角位置にあるボクセルはボクセルEである. このような対角関係は全部で4対存在する（A⇔E, B⇔F, C⇔G, D⇔H）. 一方でR=1のSCの矩形にもこのような対角関係が存在する. SCは2次元図形であるのでこちらはSCの中心に対して2次元的な点対称の関係である. 例えばFig. 21において矩形Aの対角位置にある矩形は矩形Eである. このような対角関係はSC上でも4対存在する. そこでR=1のボリュームデータにおけるボクセル同士の対角関係がR=1のSCにおける矩形同士の対角関係として展開後も保たれるように展開ルールを定めたものが上に挙げた基本展開ルール[Ⅱ]である.

我々は上述の２つのルールを比較検討した. ルール[Ⅰ]は近傍関係をできる限り保とうとするルールであるがその効果は完全ではない. 一方でルール[Ⅱ]は対角関係すなわち「互いに遠く離れている」という関係を展開後も保つものでありその効果は完全である. また対角関係は距離的に離れているという位置関係に限らず3次元空間において正反対の位置にあるという位相関係を表す指標でもある. こうした理由から我々は展開ルール[Ⅱ]のほうがより効果的であると考え, [Ⅱ]を基本展開ルールとして採用することに決定した. 以下の論文中において特に断りが無いかぎり基本展開ルールとして Fig. 21にあげたルール[Ⅱ]を用いている. 前述のようにこのルールを採用するとSC上で対角位置にある矩形同士は 3 次元ボリュームにおいても対角関係にあることが保証される. 現段階までの議論ではこの性質は R=1 についてしか保証されないが, 次段以降に述べる議論によりR≧2のより高解像度なデータを2次元展開する場合においてもこの性質が完全に保たれることが保証される. その理由は高解像度データに対してはこの展開ルールが再帰的適用されるためであるが, その詳細については次段以降に述べる.

２-３-２　高解像度なデータのための2次元展開ルール

より高解像度のデータ(R≧2)をSCに展開するためには解像度に応じた展開ルールを定める必要がある. しかし R が増加するに従ってボクセル数が膨大になると前段のように個々のボクセル･矩形に対して人の手で対応関係を設定することは実質不可能である. そこで高解像度の 2 次元展開ルール基本展開ルールから自動的に導出できるような方法を考案する. 実はこれはボクセル・SC 両方に再分割を施してゆくことで実現可能である. この再分割の際に R=1 で定めた展開ルールを再帰的に適用することで各ボクセルをSCの矩形に対応付けることができる.

例えばR=2のボクセルからなるデータをR=2のＳＣに展開する場合を考える. これはFig. 21に示したR=1のボクセル･ＳＣ双方に再分割を施すことによって達成される. まずFig. 21のボクセルAに再分割を施す. このとき同時にSCの矩形Aにも再分割を

適用する．Fig 2-6 のボクセル A は 8 個の R=2 ボクセルに分割され，SC の矩形 A もまた 8 個のサブスクエア（と 1 個のブランク領域）へと分割させる．ここで 8 個のボクセルから 8 個の矩形へのマッピングであれば最初に定めた基本展開ルールが適用可能である．ここに基本展開ルールを適用すれば 8 個のサブボクセルがそれぞれどのサブボクセルへ展開されるのかという対応関係を定めることができる．次の Fig. 22 はここまでの過程を図示したものである．

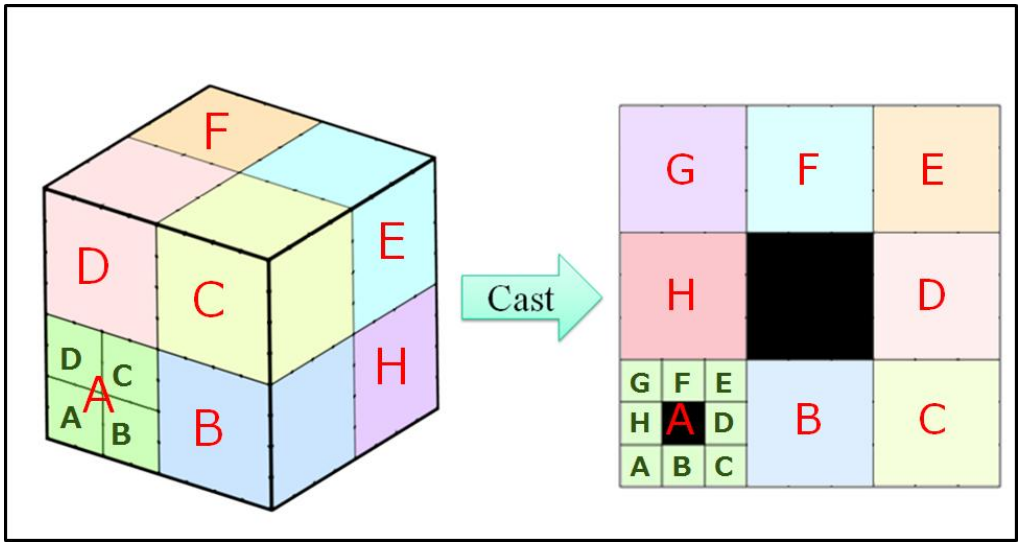

Fig. 22　R ＝ 1 のあるボクセルを再分割

同様の処理を R=1 の全ボクセル・全矩形に適用することで R=2 のボリュームデータに含まれる 64 個のボクセルを R=2 のＳＣの 64 個の矩形と 1 対 1 に対応させることができ，ここに R=2 の場合の 2 次元展開ルールが得られた(Fig. 23).

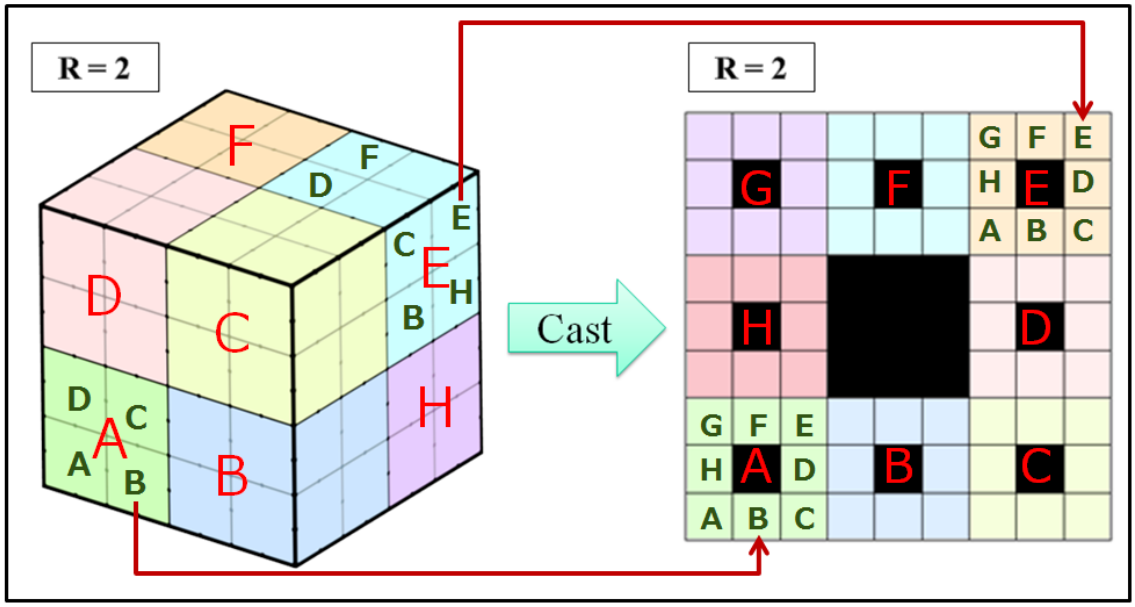

Fig. 23　R＝２の展開ルール（部分）

より高解像度のデータに関しても同様に基本展開ルールを再帰的に適用していくことによって展開ルールが導出可能である．従って R=1 における基本展開ルールさえ定義しておけば任意の解像度のボリュームデータを同じ解像度のＳＣ上に展開することができる．

２-４　本手法のメリット

ここまでに挙げた階層構造・フラクタル図形・2 次元展開といった概念群を組み合わせたときいくつかのユニークな特徴を見出すことができる．本項ではそれらの特徴を可視化におけるメリットとして活用するための指針を示す．特に SC の自己相似性および分割の局所性によってマルチレゾリューション可視化が実現されるという点は本手法を可視化手法として活用するにあたり大きなメリットとなる．さらに高解像度の SC では多数のブランク領域が生じるというデメリットをこれらのブランク領域を効果的に活用することでメリットへ転化させることが可能である．

２-４-１　マルチレゾリューション可視化

前項の Fig. 22 は再分割処理が局所的・選択的に適用可能であることを示している．なぜならこの Fig. 22 のようにボクセル A と矩形 A のみに再分割を施すことも可能だからである．この特性を活かすことで複数の解像度のボクセルデータを共存させたマルチレゾリューション可視化が実現できる．例えば次の Fig. 24 の左のような，R=1, 2, 3 の大きさのボクセルが共存したボリュームデータがあるものとする．赤色・黄色・青色のボクセルがそれぞれ R=1, 2, 3 のボクセルに対応し，それぞれ１回，２回，３回の再分割

処理が施された領域である．このような解像度の異なるボクセルが共存したボリュームデータであっても再分割処理の局所性により Fig. 24 の右に示すように１枚のＳＣ上でボリューム全域を一覧可視化することが可能である．この２次元展開結果は前述したボクセルおよびＳＣの再分割処理を局所的に適用していくことによって自動的に得られる．

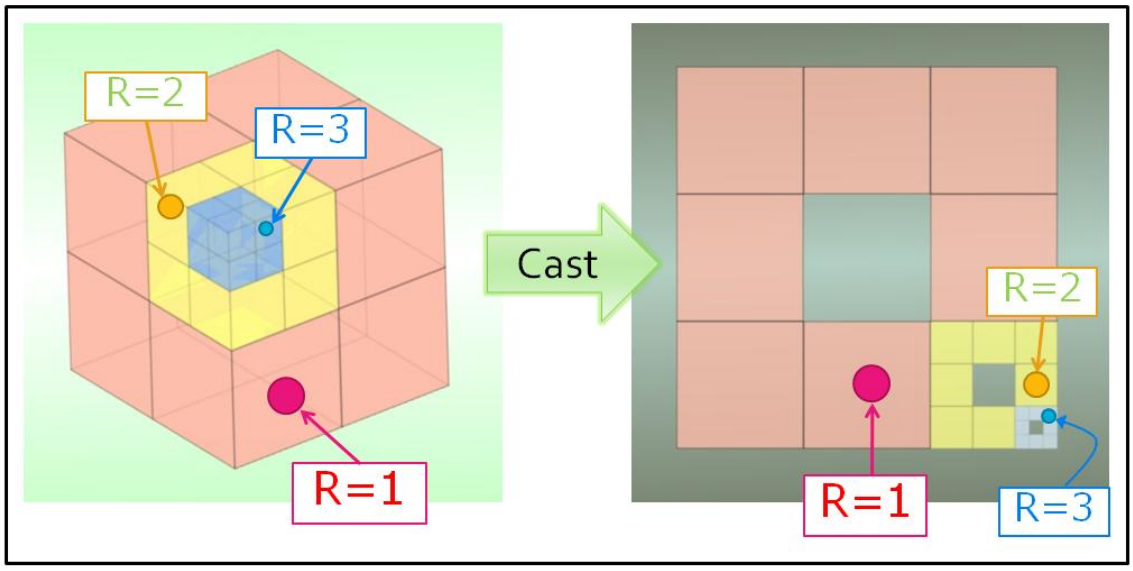

Fig. 24　マルチレゾリューション可視化

このマルチレゾリューション特性を活用すると，局所的に詳細度制御を行った LOD（Level-of-Detail）可視化を実現可能である．格納されたデータ値が均一な分布をもつボクセル領域についてはそれ以上再分割を行わず，均一でない領域や特徴的な値を含む領域についてのみ再分割処理を適用していくことで，重要度の高い情報部分を高解像度で表示し，重要度の低い部分を低解像度で表示するマルチレゾリューションな可視化が実現できる．LOD は主にグラフィクスの分野で用いられる手法でありカメラに近い部位のポリゴンを重要度の高い部位と考え細かく分割して描画し，カメラから遠い部位のポリゴンは重要度の低い部位と考えて分割せずに描画する．このように重要度に応じて精細度のレベルをコントロールすることで描画負荷を軽減することがグラフィクスにおける LOD の狙いである．同様の考え方をテクスチャに用いた場合は MipMap という．この LOD を可視化の観点から見た場合，重要な部位の情報をより多く表示し，重要度の低い部位は情報量を抑えて表示することで可視化主体が可視化結果を解釈する際の認知負荷を軽減するという効果がある．そしてそれは本手法の場合 SC の分割度合い・ボクセルの分割度合いを局所的に制御することによって達成される．これはいわゆる深さの均一でない Octree 構造にあたる．次の Fig. 25 は深さの不均一な Octree であって

も SC に展開可能であることを示している.

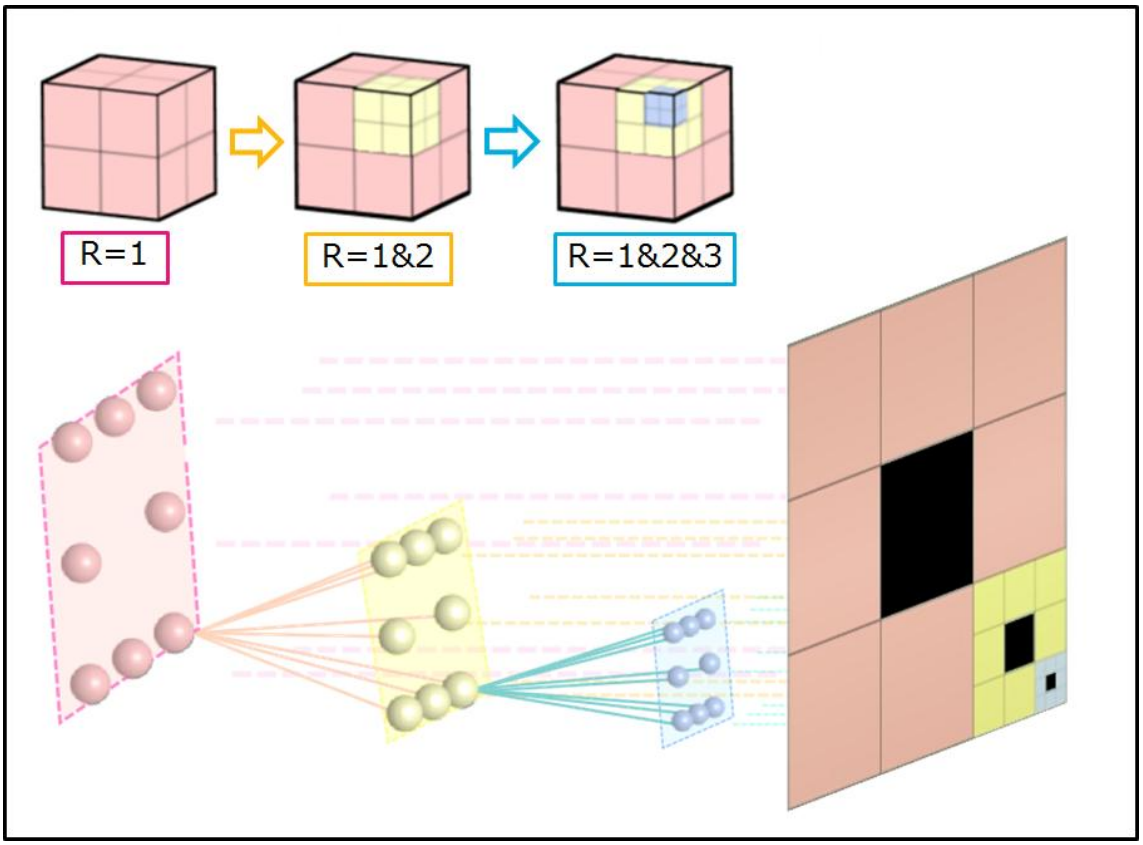

Fig. 25　深さの不均一な Octree の SC への展開

階層の深さはボクセルの解像度であり, ボクセルの解像度はそのままボクセルの細かさを表すのでこのマルチレゾリューション可視化はすなわちマルチスケール可視化を意味する.

2-4-2　2次元展開の法則性

基本展開ルールとして R=1 におけるボクセルの対角関係を SC 上で維持するようなルールを採択したが，高解像度のデータを 2 次元展開する際にこの展開ルールが再帰的に適用されるとボクセルはどのような法則をもって SC 上に展開されるのか. これは大変興味深い問題である. 当面の間 R=2 のデータに限って議論を進める. 先ほどの R=2 のボリュームデータを 2 次元展開した図を以下に再掲する.

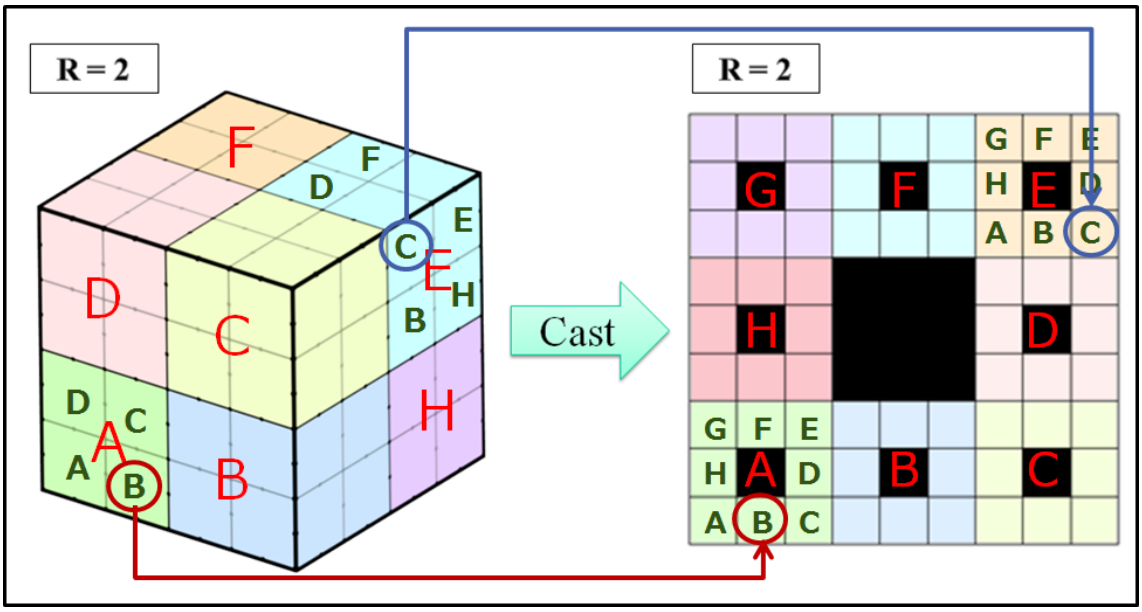

Fig. 26　R＝２の展開ルール（再掲）

２-４-２-１　包含関係の維持

ここでボクセル｛AB｝に注目する．ボクセル｛AB｝とはR=2のボクセルであって，R=1においてボクセルAに属し（上の赤文字のAの領域），R=2においてボクセルBの位置に相当するボクセルのことである．これはFig. 26左図で言うところの赤丸で囲まれたボクセルである．このボクセルの展開先は Fig. 26 右図の赤丸で囲んだ矩形であり，この矩形はR=1におけるAの領域（Fig. 26右図左下の赤文字Aの領域）内部のさらにR=2においてBの位置にある矩形(左下角)に位置している．この矩形を同様の表記法を用いて矩形{AB}と呼ぶことにする．つまりボクセル｛AB｝は矩形｛AB｝に展開される．同様にボクセル｛EC｝（青丸で囲んだボクセル）は矩形｛EC｝（同じく青丸の矩形）に展開される．

さてこのR=2の展開においてボクセルの包含関係が展開後も矩形の包含関係となって維持されていることに注目したい．ボクセル｛AB｝はR=1におけるボクセル｛A｝に内包されるボクセルであるが，その 2 次元展開後の矩形｛AB｝はやはり矩形｛A｝に内包される矩形である．別の例としてボクセル｛E｝に内包されるボクセル｛EC｝はやはり矩形｛E｝に内包される矩形｛EC｝へと展開される．このようにボクセルの内包関係はそのまま SC における矩形の内包関係となって展開結果に現れてくる．この理由として Fig 2-7 に示したようにボクセルおよびＳＣの再分割が局所的であることに注目したい．再分割が局所性をもつおかげでボクセル構造における包含関係が 2 次元展開後も保たれるというメリットがもたらされているのである．このような包含関係はより高解像度のボリュームデータを展開した場合にも保たれる．すなわちボクセルの Octree 構造の親子関係がそのままＳＣ上に包含関係として投影されているのである．

2-4-2-2　Corner-to-Corner マッピング

ボリュームデータの角に位置するボクセルの展開位置に注目する．対角関係を保つ基本展開ルールを用いたことによって，これらのボクセルの展開先にはどのような規則性が現れるのであろうか．以下の Fig. 27 に R=2 のボクセル｛AA｝およびボクセル｛EE｝の展開先を示す．

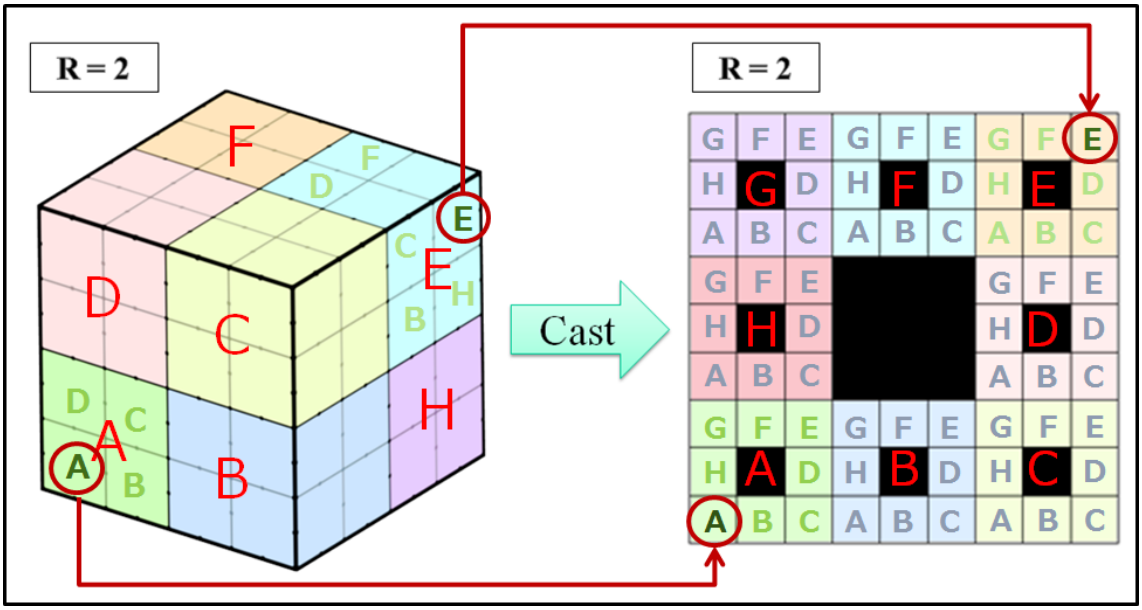

Fig. 27　コーナーに位置するボクセル｛AA｝およびボクセル｛EE｝の展開先

ここでボクセル｛AA｝（Fig. 27 左図で言うところの左下の赤丸で囲まれたボクセル）に注目する．このボクセルはボリュームデータの左下角に位置するボクセルであるが SC 上でもこれに対応する矩形が R=1 で A の領域（Fig. 27 右図左下の赤文字 A の領域）内部のさらに R=2 において A の位置にある矩形（左下角）に位置している．この矩形を同様の表記法を用いて矩形｛AA｝と呼ぶことにする．つまりボクセル｛AA｝は矩形｛AA｝に展開される．これは基本展開ルールが 2 度再帰的に適用された結果であることに注意する．まず一度目の展開（R=0 から R=1 への分割に伴う展開，Fig. 27 を参照）ではボクセル｛A｝（Fig. 27 左図において左下の緑色の領域）が SC の矩形｛A｝（Fig. 27 右図において左下の緑色の領域）に展開される．さらにこのボクセル｛A｝に再分割を適用する．そうすることでボクセル｛A｝は 8 個のボクセル｛AA｝・｛AB｝・｛AC｝・｛AD｝・｛AE｝・｛AF｝・｛AG｝・｛AH｝に分割され，2 度目の基本展開ルール適用によりそれぞれ対応する矩形へ展開される．ボクセル｛AA｝は 2 度の展開で『左下→さらにその左下』と展開されるため結果的に SC 左下角の矩形へと展開されるのである．

それでは｛AA｝以外の角に位置するボクセル｛BB｝・｛CC｝・｛DD｝・｛EE｝・｛FF｝・｛GG｝・

{HH}はそれぞれどの矩形位置へ展開されるのであろうか．これを示したのが次の Fig. 28 である．

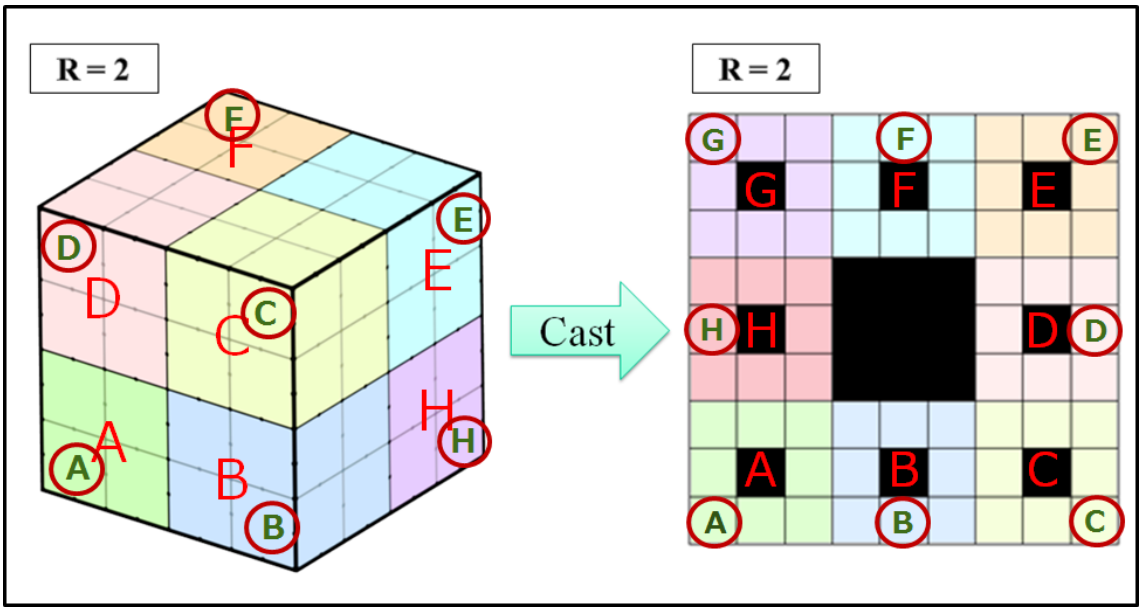

Fig. 28　角に位置するボクセルの展開先矩形位置

角に位置するボクセルとその展開先の矩形には赤丸で印をつけた．さて上図を見ればコーナーに位置するボクセルの展開先にはある規則性があることが見て取れる．対角関係を保つような基本展開ルールを用いた場合，ボリュームデータの角に位置するボクセルは SC の周縁部の矩形へと展開される．つまりコーナーにあるボクセルは展開後には SC のコーナーかもしくは各辺の中点に位置する．この Corner-to-Corner のマッピングが今回採用した展開ルールを多段適用した際の大きな特徴でありこの傾向はより高解像度のデータに対して基本展開ルールがさらに再帰的に適用されても変わることはない．

2-4-2-3　Inner-to-Inner マッピング

それでは外縁ではなく中心部のボクセルはどうであろうか．R=2 のボリュームデータにおいて中心部のボクセルとは {AE}・{BF}・{CG}・{DH}・{EA}・{FB}・{GC}・{HD} のことを指す．これらのボクセルはボリューム表面に現れておらず，ボリュームの中心部付近に存在するボクセルである．これらのボクセルの展開先は次の Fig. 29 のようになった．

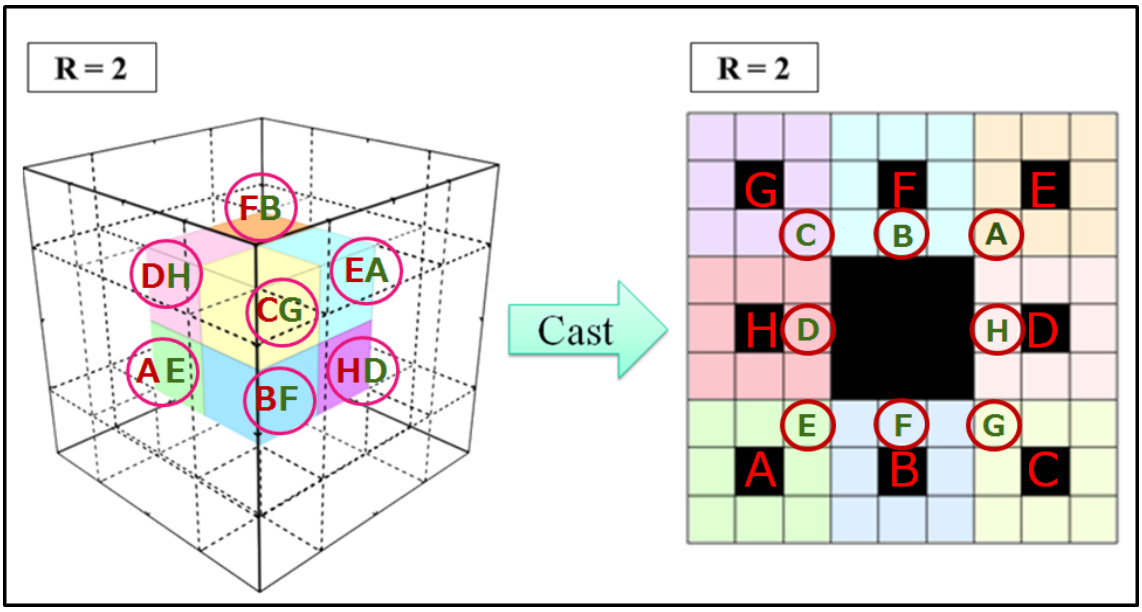

Fig. 29　中心部に位置するボクセルの展開先矩形位置

中心部のボクセルとその展開先の規則性は上の図から一目瞭然である．対角関係を保つような基本展開ルールを用いた場合，ボリューム中心部に位置するボクセルは SC 上でも内側の矩形へと展開される特性を有する．この Inner-to-Inner マッピングの特性は展開ルールをさらに多数回再帰的に適用しても不変である．

参考までに R=3 の展開図を以下 Fig. 30 に示す．ボリュームデータのコーナーに位置するボクセルに対応する矩形は赤色，中心部に位置するボクセルに対応する矩形は青色で示している．

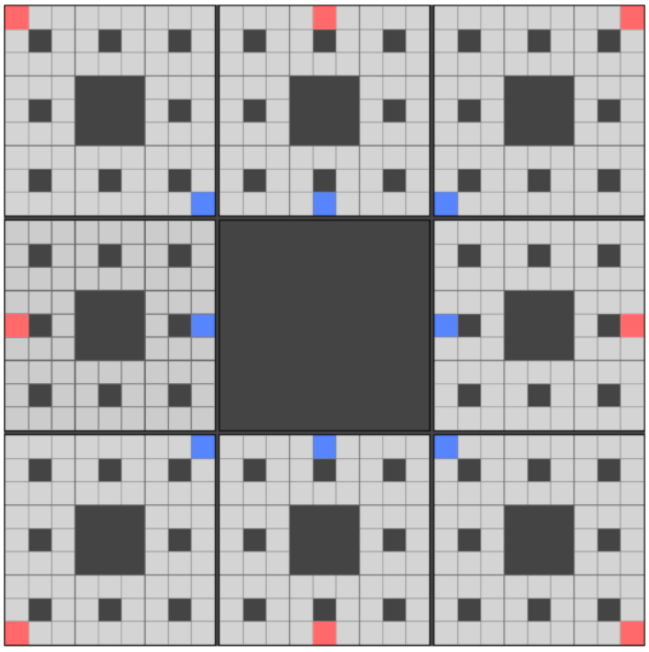

Fig. 30　Corner-to-Corner Mapping（赤）と Inner-to-Inner Maapping（青）（R=3）

２-４-２-４　対角関係の維持

ここで Fig. 27 のボクセル｛AA｝と｛EE｝の位置関係に注目する．この２つのボクセルはボリュームデータにおいて互いに対角関係に位置する．そしてその展開先の矩形においても，矩形｛AA｝と矩形｛EE｝は互いに対角関係を保ったままである．すなわち基本展開ルールにおいて対角関係が保たれている場合には，2 度の 2 次元展開の適用を経た後でもボリュームにおけるボクセルの対角関係は保たれるということである．Fig. 27 のその他の対角関係（｛BB｝と｛FF｝や｛CC｝と｛GG｝など）もしくは Fig. 29 に見られるあらゆる対角関係もまた 2 次元展開後も維持されていることに注目したい．そしてこの対応関係の維持は基本展開ルールが何度適用されようと維持される．

このように展開ルールの再帰的適用は高次元の 2 次元マッピングに様々な規則性をもたらすことが明らかになった．

２-４-３　ブランク領域の活用

ＳＣの描画アルゴリズムのステップ(iii)によって，高解像度のＳＣには多数のブランク領域が存在する．このブランク領域に相当する矩形の個数は全体の 1/9 であるがその面積は解像度により異なる．R=1 の場合ＳＣ全体の約 11.1%，R=2 の場合は 22.0%，R=3 では 29.8%であり，無限回の再分割を施した理想的なＳＣでは全体の 100%がブランク領域であるという結果に収束する．これはＳＣのフラクタル次元が約 1.8928（< 2）であるためである．本来このように多数のブランク領域が生じることはデメリットである

といえる. しかしこのブランク領域に補助的な情報を可視化することによって逆に本手法のメリットとして活用することが可能であると考えた. 幾何学的な観点から見るとブランク領域は分割によって生じた 8 個の矩形の中央に位置し, 言い換えれば周囲をこれら 8 個のボクセルに囲まれている. 本段ではこのブランク領域に周囲の 8 個の矩形に対応するボクセル値の平均値を格納し可視化する手法を提案する. さらに分割によって損なわれるボクセルの隣接性を補うためにボリューム中央の領域を SC 中央のブランク領域に展開可視化する手法も提案する.

2-4-3-1 平均値の可視化

1 つ目の活用法として分割する前段階のボクセル値をブランク領域に可視化する方法がある. これは再分割ステップ(ii)による 9 分割によって生じる中央のブランク領域にはその周辺の 8 個の矩形に格納された値の平均値が可視化されるということである. このブランク領域に対してそれ以上の再分割処理を行う必要はない. このような手法をとることで例えばデータ全体の平均値は中央のブランク領域の色を見るだけで直ちに認識可能である. さらにボリュームデータを様々なスケールで見た場合の可視化結果が 1 枚の S C 内によって表現されているため本手法のもつマルチスケール可視化の特徴をさらに効果的に利用可能である(Fig. 31).

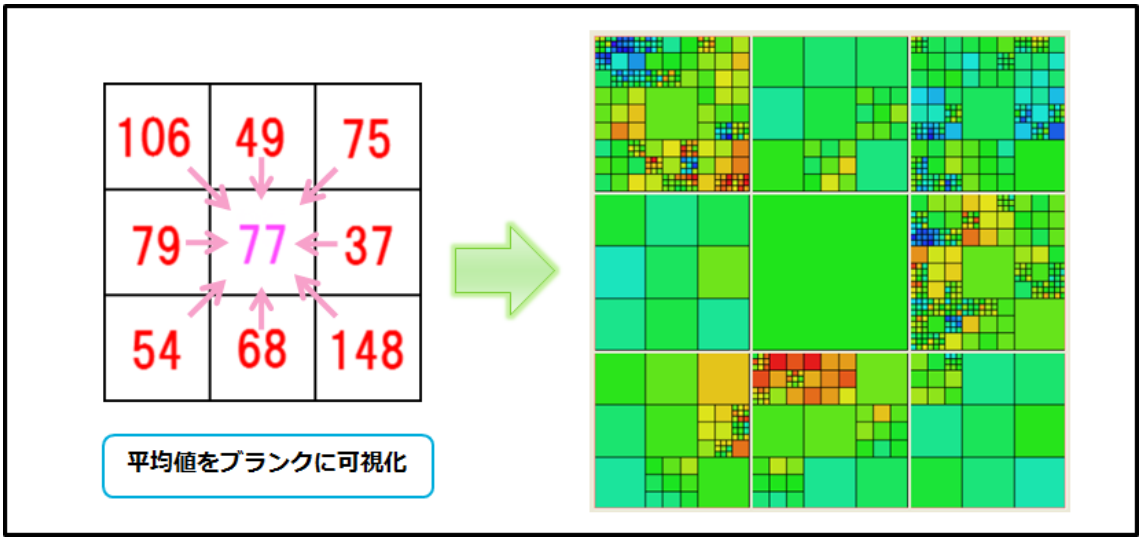

Fig. 31　平均値の可視化

2-4-3-2 中心領域の可視化

本研究で提案する 2 次元展開の欠点として挙げられる点に, ボクセル分割の中心近傍に位置するボクセルは 3 次元的には近接位置にあるにも関わらず 2 次元展開後は互いに

離れた矩形に展開されてしまうという問題がある. ブランク領域に分割中心付近の領域のみを対象とした展開を行うことでこの欠点を軽減する. 具体例として Fig 2-17 の左図のような 64 個のボクセルからなる R=2 のボリュームデータを考える. このときボリュームデータの中心部にはボリューム表面に現れていないボクセルが 8 個存在する. この 8 個のボクセルの位置は互いに近接関係にあるものの，２次元展開後は互いに離れた矩形へと展開されてしまう（Fig. 32　右図）.

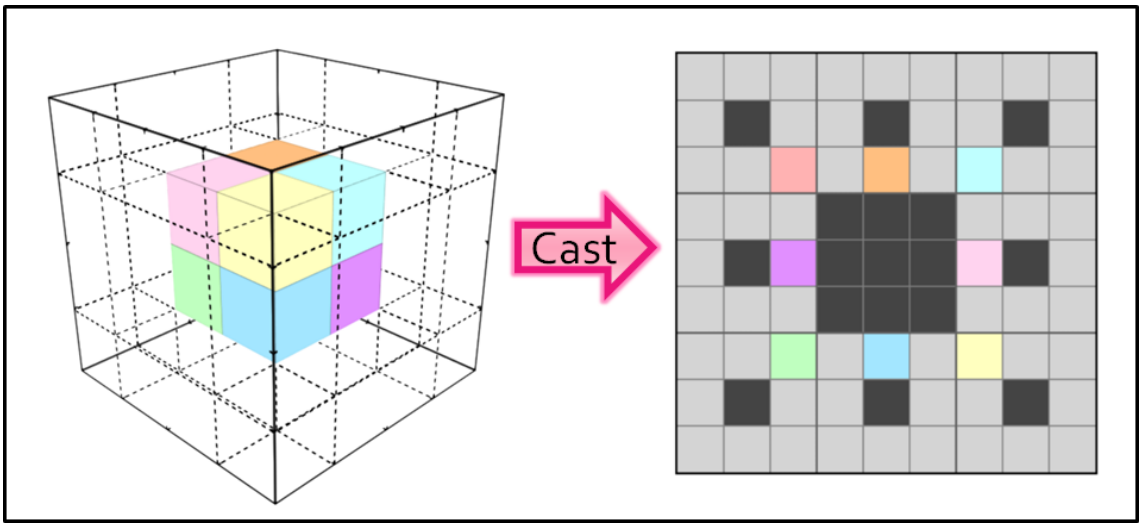

Fig. 32　展開後の中央ボクセルの分布（Ｒ＝２）

そこでこれら 8 個のボクセル群を R=1 のボリュームデータとみなし，基本展開ルールを適用することによってＳＣの中央のブランク領域(R=1 のブランク領域と呼ぶこととする）へと展開する．その様子を Fig. 33 に示す.

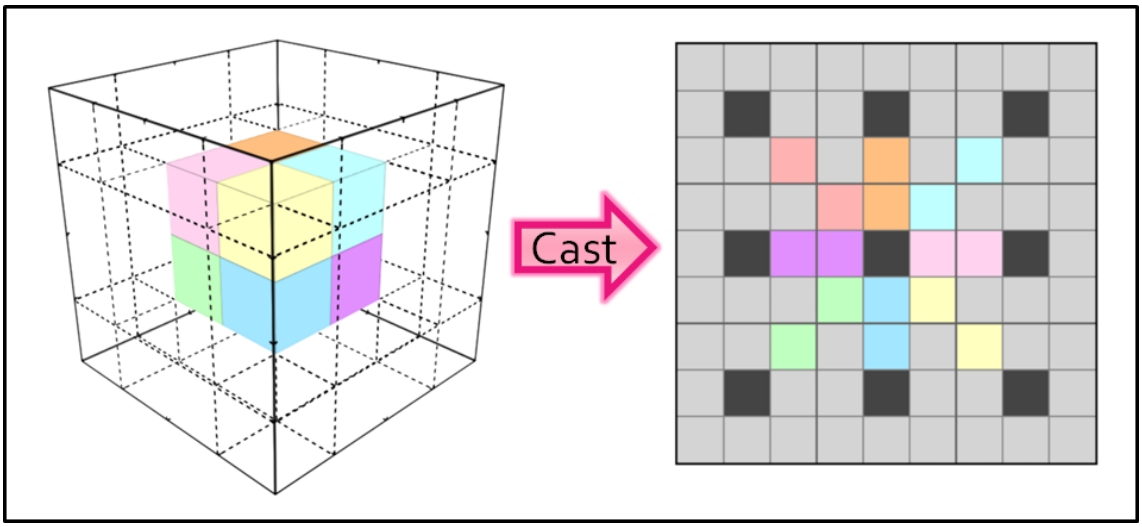

Fig. 33　中央領域の展開　Ｒ＝２

さて 512 個のボクセルからなる R=3 のボリュームデータは R= 2 のボリュームデータ（ただし各ボクセルは R=3 のボクセルであり，スケールは 1/2）が 8 個結合したものであると考えることができる．そのように考えれば先程の中央領域の展開ルールは R=3 のボリュームデータにも適用することができる．この場合 R=2 のブランク領域まで展開が行われ，その結果は Fig. 34 のようになる．

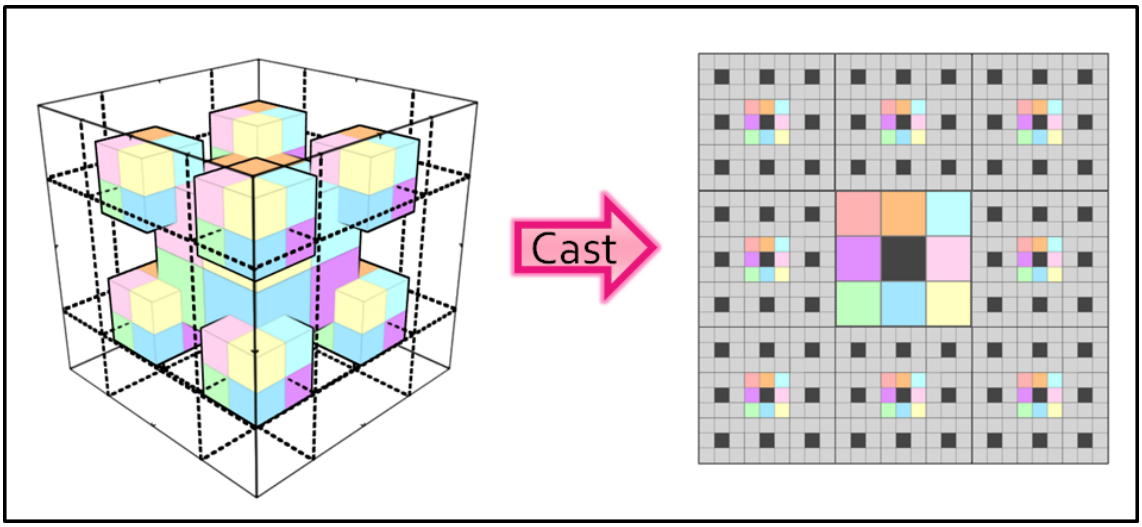

Fig. 34　中央領域の展開　R＝3

以上のように考えることで任意の解像度のボリュームデータに対して中央領域の展開を実行することが可能であり，本来２次元展開により破壊されてしまう近傍関係を保ったまま可視化を実現することができる．この手法はデータのもつ局所的特長を抽出するうえで有用であると考える．また，ボリューム中心のボクセルがＳＣにおいても中心領域に展開されるという Center-to-Center のマッチングが実現されているためデータ中心部の特徴を可視化主体が直感的な認識しやすくなるといったメリットが考えられる．

２-４-４　その他のメリット

前述のようにブランク領域を補助的情報の可視化に活用すると SC 全域が満遍なく可視化に利用されることになる．こうした空間充填率の高さは 2 次元可視化特有のものでありこれも本手法の特徴のひとつである．本手法の可視化像が Tree Map などの２次元情報可視化手法のものに類似している点は注目に値する．どちらも単一の矩形を初期状態として空間分割を施すという描画アルゴリズムによって高い空間充填率が達成されていることが共通点である．

さらに本来３次元的なボクセル構造を２次元に展開したためボリューム全域を遮蔽

なく概観することが可能となっている．このような大規模データの概観表示は情報可視化の一般的な特徴であり本手法が情報可視化的発想に基づいた手法であることの表れである．また，視点依存性がなく１枚の画像上で全ボクセルの情報が個別に可視化されているため複数のデータを並べて比較することも容易であると考えられる．これは視線に沿って伝達関数の積分を行うボリュームレンダリングと対照的な特徴である．その結果特徴的な値をもつ領域が色のコントラストにより一目で認識できるようになっている．ボリュームレンダリングでは視線上に並んだ各サンプリング点の寄与は積分されているため各ボクセル値は個別に可視化されない．従ってボリュームレンダリングの可視化結果は１枚では解釈の一意性をもたない．そのため視点を変更しながら様々な角度からボリュームデータを眺め，再レンダリングする必要がある．一方で本手法では各矩形がボクセルと１対１に対応しているためその解釈に一意性を期待することができる．

２-５　本手法のデメリット

本手法にもいくつかの欠点が存在する．そのひとつは空間の連続性が保たれないことである．2 次元展開を用いる場合，フラクタル展開により本来隣接した領域が SC の矩形上に離散的に展開される．そのためボリュームデータが内部に 3 次元的に値の連続した領域を保持していても 2 次元展開後はその連続性が保たれないという問題が生じる．これを説明するために上で定めた基本展開ルールを再掲する．

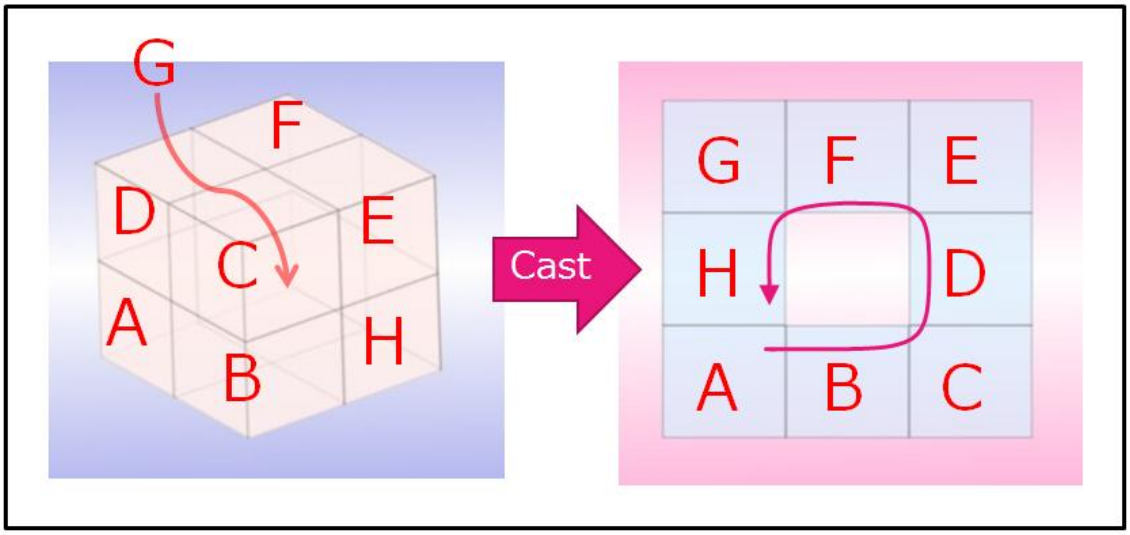

Fig. 35　R＝１の展開ルール［Ⅱ］（再掲）

上図右よりボクセル A の隣接ボクセルとしてはボクセル B・D・G がある．しかし 2 次元展開後の矩形 A と隣接する矩形は B と H であり矩形 D・G は A と隣接関係にない．展開ルールは 1 対 1 の対応が保たれる限り自由に設定して構わないがどのような基本展開ル

ールを用いてもボクセルの隣接関係を展開後も完全に保つことは実際には不可能である．このようにR=1の2次元展開ルールにおいてさえボクセルの隣接関係は保たれていない．したがって高解像度のボリュームデータを展開した場合にもこの問題は必然的に発生する．この問題は必然であると同時に不可避である．その根本の原因は2次元と3次元における位相構造の違いにある．このことを3次元から2次元への展開という類似のロジックであるポリゴンUV展開における連続性の破壊を例にとり考察する．

ポリゴンのUV展開は3次元ソリッドサーフェイスであるポリゴン表面に平面テクスチャを貼るために，多角形（一般には3角形・4角形）ポリゴンを2次元平面テクスチャ上に展開する手法である．この際テクスチャをポリゴン表面に継ぎ目なく貼り付けるためにはポリゴンモデルの3角形･4角形面の連続性をできる限り保ったまま展開することが要求される．しかしながら閉じた3次元ポリゴンモデルはたとえ凸包（convex）であっても各面の連続性を完璧に保ったままUV展開することは一般に不可能である．UV展開は3次元座標（x, y, z）から2次元座標（u, v）への展開とはいえ厳密にはモデル表面（2次元）から2次元平面への展開である．にもかかわらず要素の連続性は破壊されてしまうことは避けられず，ここにその原因が3次元構造と2次元構造の位相的性質の相違にあることが明確となる．ボクセルの場合内部構造を含むため1つのボクセルの隣接ボクセルの数は最大で6個であり，一方でSCの矩形は最大でも4個の隣接矩形しか持ちえない．このように3次元構造を2次元平面に展開する際に連続性が損なわれる問題は不可避であると言える．

別の欠点としてSCのある矩形がどの3次元領域に対応するかという関係が直感的には認識困難であるという問題がある．基本展開ルールそれ自体は単純であるが，それが再帰的に適用されることで展開ルールは複雑化し，ある矩形に対応するボクセルの3次元座標を直感的に把握することは難しくなる．前述したようにいくつかの特徴は基本展開ルールが再帰的に適用されても維持されるが，これらの特徴は任意の矩形とボクセルの位置関係を人間が即座に導出・理解することを可能にするような類のものではない．

これら2つの大きな欠点を解決するためにSCによる2次元展開を用いた可視化に加え，空間連続性を保ったまま可視化可能な既存の3次元可視化手法を併用する必要があると考えた．また，情報可視化の観点から考えて2次元可視化結果はそのまま入力インターフェースとしての利用が可能である．これは2次元可視化と3次元可視化のコラボレーションが先に挙げた欠点の克服以上のメリットをもたらす可能性を示唆している．その内容・実装・可視化例については続く第3章で議論する．

第３章　３次元可視化とのコラボレーション

３-１　方針

前章では階層的ボクセルからなるボリュームデータを２次元フラクタル図形であるＳＣ上に展開する手法を提案した．本手法を考察した結果２次元展開による可視化ではボリュームデータに格納された３次元形状や値の連続性の把握が困難であるという欠点があることが明らかになった．この欠点を軽減するために２次元可視化手法である本手法と一般的な３次元ボリュームデータ可視化手法とのコラボレーションを実現する．これは同時に複数の手法による多角的な可視化を実現することも意図している．本章では提案手法が提供するボリューム全域を概観可能な２次元表示と，詳細表示・形状把握に適した既存の３次元可視化を連動させたインタラクティブな可視化システムを構築する．これは情報可視化において既にホット・トピックである２次元可視化と３次元可視化のコラボレーションを科学的可視化に取り入れることに相当する．このコラボレーションを実現するにあたり２次元可視化結果をユーザー入力インターフェイスとして活用することによって２次元可視化手法・３次元可視化手法間のインタラクションを可能にする．可視化アプリケーションの開発には Visual Studio 2008 Express Edition を開発環境とし，言語として C# 3.0，グラフィクス API として DirectX 9.0c，シェーダ言語として HLSL（High Level Shader Language）を用いた．可視化結果の描画およびデータ値の色変換に GPU を使役することで可視化パフォーマンスを高めた．次に示す Fig. 36 は我々の開発した可視化アプリケーションのスクリーンショットである．

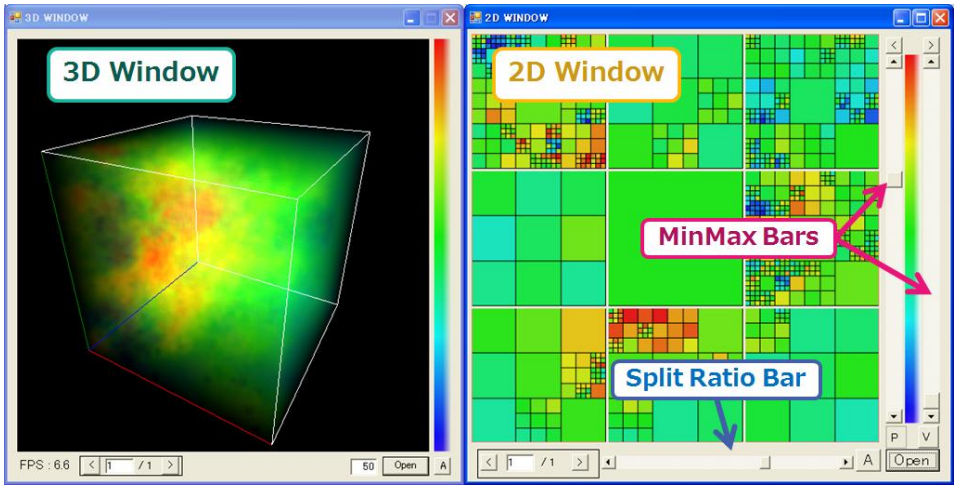

Fig. 36　可視化アプリケーションのスクリーンショット

このアプリケーションは左側に 3D Window，右側に 2D Window という 2 つの可視化ウィンドウを備える．2D Window の右端には可視化結果の表示をコントロールするための 2 つの垂直スクロールバーがあり，これらを MinMax Bar と呼ぶ．2 本の MinMax Bar の間に挟まれたグラデーションテクスチャがスカラー値の色変換に用いられる 1 次元テクスチャである．また，Octree の分割の細かさを変更するための水平スクロールバーが下端に備えられており，これを Split Control Bar と呼ぶ．

3-2　スカラーデータの可視化

スカラーボリュームデータはボリュームデータの各ボクセルにスカラー値を格納したデータであり，温度・密度・圧力などの物理量分布を記録したデータがこれにあたる．本段ではこのスカラーデータに対し提案手法である 2 次元展開と 3 次元可視化手法であるボリュームレンダリングを組み合わせた可視化を実現する．

ここでは 3 次元流れ場における圧力分布の時系列データを可視化対象として用いた．このデータは自動車モデル周辺の流れ場を CFD により計算したシミュレーションデータから自動車モデル後方部にあたる領域の流れ場の圧力分布をボリュームデータとして抽出したものである．下の Fig. 37 は CFD 計算結果全域を Scatterplot で可視化したものであり，今回可視化対象とする領域は白枠で囲まれた立方領域である．

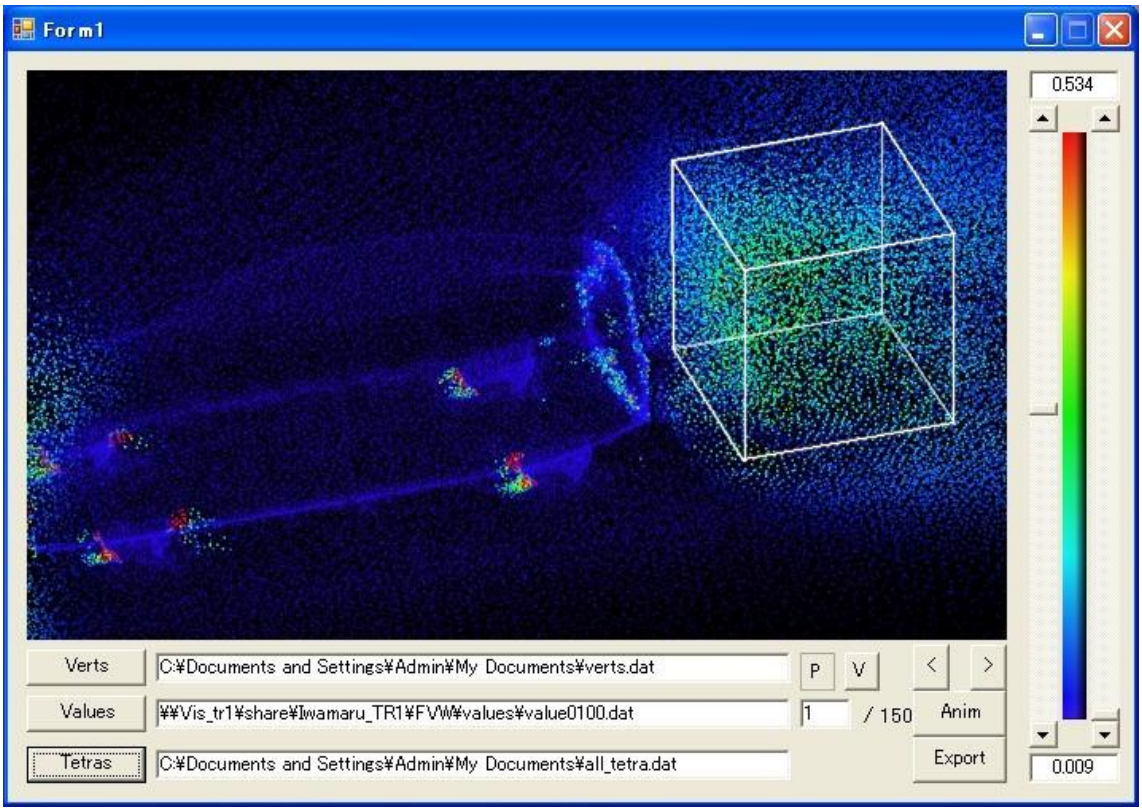

Fig. 37　今回可視化対象とした領域

この領域におけるデータ値分布を各軸方向に 32 個のボクセルからなる R=5 のボリュームデータとして抽出し入力データとした.

3-2-1　2次元展開

ボクセルの 2 次元展開は内包されるデータ値の分散値に基づいて行った．例えば入力データの解像度が R=5 であった場合このボリュームデータの最大解像度は 5 である(R_{max}=5)．始めに R=0 の単一のボクセルから始める．この R=0 のボクセルは 32×32×32＝32768 個のボクセルを代表し，その内部には 32768 個の R=5 のボクセルが内包されていると考えることができる．従ってこの R=0 のボクセルに格納されるデータ値は 32768 個の R=5 のボクセルに格納された値の平均値である.一般に解像度 R の各ボクセルには $8^{R_{max}-R}$ 個の R_{max} ボクセルが内包されている．本手法においてあるボクセルの平均値とはそのボクセルが内包する R_{max} のボクセル値の平均値を指す．また，あるボクセルの分散値σ^2とはそのボクセルの内包する全 R_{max} ボクセルの値の分散値であり次式(3)で表される.

$$\sigma^2 = \frac{1}{Num}\sum_{i=1}^{Num}(x_{ave} - x_i)^2 \qquad (3)$$

ここで Num はそのボクセルが内包する R_{max} ボクセルの総数（$8^{R_{max}-R}$ 個），x_iは内包する

各 R_{max} ボクセルに格納された値，x_{ave} はそのボクセルに格納された平均値を表す．本手法ではボクセルの分散値をそのボクセルを分割するかどうかの指標として用いる．分散値が一定値以下のボクセルは内部のデータ値分布が均一であるとみなしそれ以上の再分割を行わない．一方ある閾値よりも高い分散値をもつボクセルには再分割を施すことでより高解像度の可視化を行う．今回の可視化では閾値を定めるにあたり R=0 のボクセルにおける分散値$\sigma^2{}_{ref}$を参考値とした．ある解像度 R のボクセルの分割判断に用いる閾値thは次の式(4)により定義される．

$$th = a^R \sigma^2{}_{ref} \qquad (4)$$

ここでaは分割の細かさをコントロールするための係数である．aが小さいほどthの値が小さくなりより細かく再分割が行われる．このaの調節には Fig. 36 の Split Control Bar を用いる．

具体的な再分割の手順は以下の通りである．まず R=0 のボリュームデータから始める．このデータは R=0 のボクセル 1 個を持ち 8^{Rmax} 個のボクセルが内包されている．このボクセルの分散値を計算しその値を$\sigma^2{}_{ref}$とする．この R=0 のボクセルに再分割を施し 8 個の R=1 のボクセルに分割する．それらの各 R=1 ボクセルについて分散値を計算し閾値thと比較する．分散値がthよりも大きい場合そのボクセルに対しさらに再分割を施す．新たに分割されたボクセルに対して同様に分散値算出・比較・分割の手順を繰り返す．以上の手続きを分割後のボクセルの解像度が R_{max} に達するか，もはや分割するボクセルが存在しなくなるまで繰り返す．以上のように再帰的なプロセスによってボクセル分散値に基づいた Octree を構築し，これを前章で述べた展開ルールに従って SC 上に 2 次元展開する．ユーザーが Split Control Bar を操作し，aが変更された際には Octree を再構築する．段階的に分割が進む様子を以下の Fig. 38 に示す．

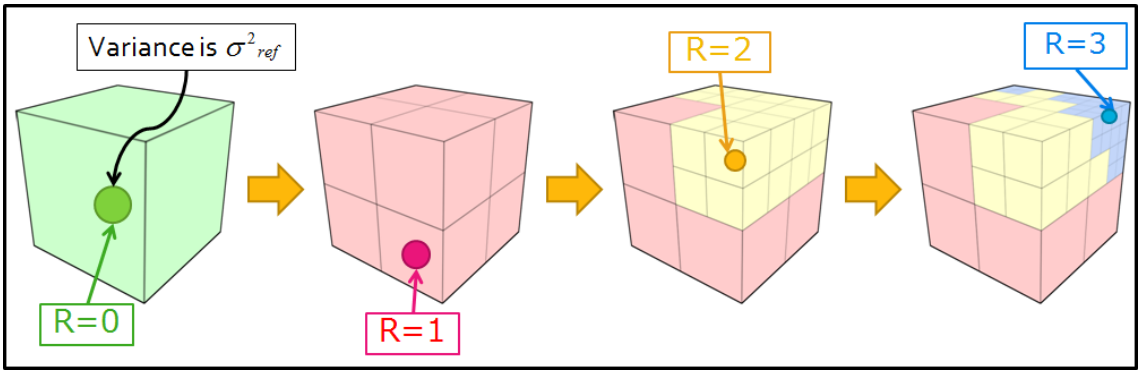

Fig. 38　分散値に基づいたボクセル分割の様子

3-2-2　スカラー値から色への変換

本手法では各ボクセルは SC 上の 1 個の矩形と対応しており，各ボクセルに格納されたスカラー値を矩形の色に変換することで可視化する．この色変換処理を Colorization と呼ぶ．変換には 1 次元のテクスチャを用いた．次のテクスチャサンプリング関数（Texture Sampling Function）T はテクスチャ座標 u を指定することでスカラー値 v を色 c に変換する関数である．

$$c = T(u(v)) \tag{5}$$

ユーザーによる可視化表示結果のコントロールを可能にするために，可視化に関わる 2 種類の閾値 v_{max}，v_{min} を導入する．v_{max} よりも大きな値はテクスチャの最大値に相当する色に変換され（u>1），v_{min} よりも小さな値はテクスチャの最小値に相当する色に変換される（u<0）．この 2 値はユーザーによって調節可能な変数である．調節には Fig. 36 の Min Max Bar を用いる．次式(6)はスカラー値の色変換式（Colorization Function）を表す．

$$c = T\left(\frac{v - v_{min}}{v_{max} - v_{min}}\right) \tag{6}$$

この Colorization 処理は HLSL の Pixel Shader 2.0 によって計算される．

3-2-3　ボリュームレンダリング

スカラーデータの 2 次元・3 次元協調可視化を実現するにあたりダイレクト・ボリュームレンダリング（Direct Volume Rendering）を実装した．より高いパフォーマンスを発揮するため HLSL による GPU プログラミングを行い，GPU 処理による Direct Volume Rendering を実装した．近年の GPU は並列処理に特化した計算パイプラインを備えているため Direct Volume Rendering 等の各ピクセルの処理が独立した計算を高効率に実行可能である．計算アルゴリズムには最も一般的に用いられるレイ・キャスティング（Ray Casting）法を採用した．Ray Casting におけるサンプリング点は均等間隔配置とし，Adaptive Ray Sampling は行っていない．サンプリング点におけるスカラー値は近傍ボクセル値の線形補完値とした．DirectX 9.0c では Volume Texture をサポートしているので，Volume Texture の Sampling Filter Type を Linear とすればテクスチャサンプリング時に自動的に線形補完されたスカラー値を取得可能である．スカラー値から色への変換には 2 次元可視化に用いたものと同じ 1 次元テクスチャを用いた．こうして取得したテクスチャ色にアルファ値と光の減衰を考慮した重み付けをしたうえで足し合わせ，最終的なピクセル色とする．

３-２-４　機能と操作

ボクセル展開による 2 次元可視化手法と 3 次元可視化手法であるボリュームレンダリングの相互運用を実現するためユーザー操作を伴う幾つかの機能を実装した.

A)　再分割の細かさの変更

既に述べたように Split Control Bar を操作してaの値を変更し，再分割の閾値thを調節することで矩形分割の細かさをコントロールすることができる.

B)　可視化パラメータ変更による表示結果の調節

これも既に述べた内容であるが Min Max Bar を操作して v_{max}，v_{min} の値を変更することによって可視化結果の色コントラストを調節することができる．この v_{max}，v_{min} の変更はボリュームレンダリングにも同時に反映され, 2 次元可視化と 3 次元可視化の各表示結果に一貫性をもたせる.

C)　可視化領域の直接的な選択

ボリュームレンダリングにおいて領域の一部のみを抽出して可視化したい場合がある．しかし抽出すべき領域を発見することが困難であることに加え，注目したい領域を特定できたとしても 3 次元可視化特有のアクセシビリティの欠如から 3 次元領域を直接的に指定することが困難であるという２つの問題が生じる．この問題を解決するために我々は 2 次元図形であるＳＣを入力インターフェイスとして活用する．本手法では 3 次元空間のある領域を占めるボクセルとＳＣ上の矩形が１対１に対応しているため，ＳＣ上のある矩形を選択することで対応するボクセル，すなわち 3 次元領域を指定することができる．我々はこの特性を利用して，指定した矩形に対応する 3 次元領域のみをボリュームレンダリングの対象領域として設定する機能と実装した. この機能は前述の 2 つの問題を同時に解決する. まず, 2 次元可視化結果にはボリュームデータに記録されたスカラー値が矩形色として遮蔽なく可視化されているため，ユーザーはＳＣ上でボリューム全体を概観的に眺め，特徴的なデータ値分布をもつ領域を容易に発見可能である．そうして特定した特徴領域をマウス操作で選択することで直接的にボリュームレンダリング対象領域を指定することが可能である(Fig. 39).

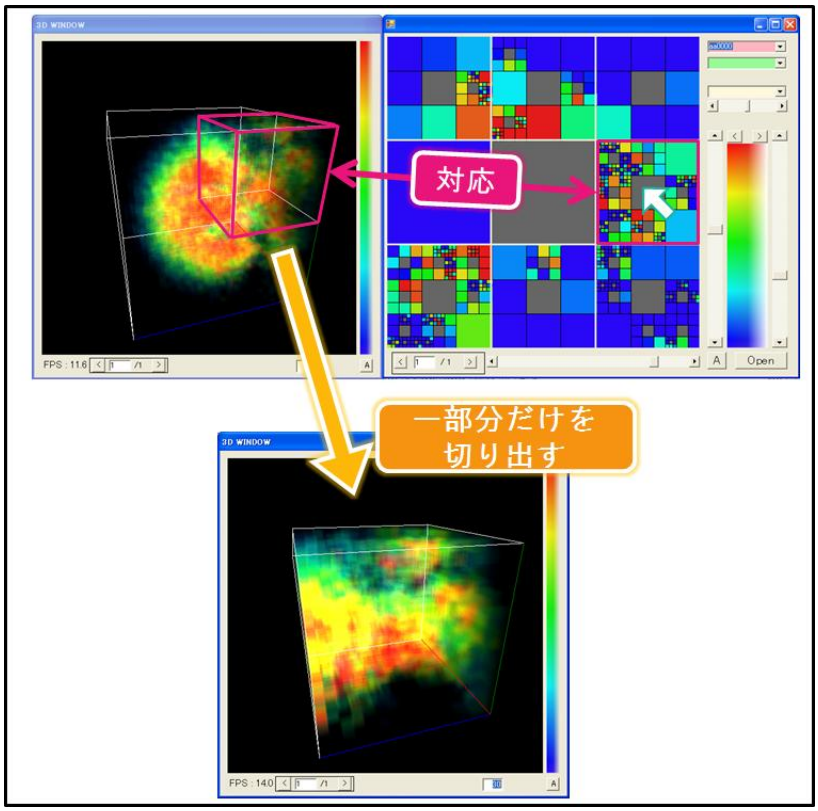

Fig. 39　可視化領域の直接的な選択

D)　指定したボクセル値の減算

C)で述べた領域選択機能は矩形を選択することで対応する 3 次元領域を取得するものであったが，この機能は対応するボクセルに格納された値を取得するために用いることもできる．すなわちボクセル格納値に対する直接的なアクセスを提供する．我々は指定したボクセル格納値をデータ全体から減算する機能を実装した．この機能の目的は，指定したボクセル値を Colorization の基準値（u(v) = 0 に相当する値）に設定することで，その基準値よりも高い値をもつ領域のみを抽出し可視化することにある．具体的には Colorization Function を以下のように修正する．

$$c = T\left(\frac{v - v_{sub} - v_{min}}{v_{max} - v_{min}}\right) \qquad (7)$$

ここで v_{sub} は指定した減算値である．この減算値の指定は対象とする矩形を右クリックで選択後現れるメニューより Subtract this value を選択することで行う．このとき v_{sub} より小さい値をもつボクセル値は T(0)に相当する色が割り当てられる．この色について不透明度を 0 に設定することで v_{sub} 以下の値を格納したボクセルは可視化結果に現れなくなり，v_{sub} より大きい値をもつ領域のみを可視化することが可能となる．このように任意の矩形

（ボクセル）を指定し，そのボクセル格納値を基準とした可視化を 2 次元マウス操作のみで実現できることがこの機能の利点である. 以下 Fig. 40 にこの機能と B)を併用した可視化例を示す.

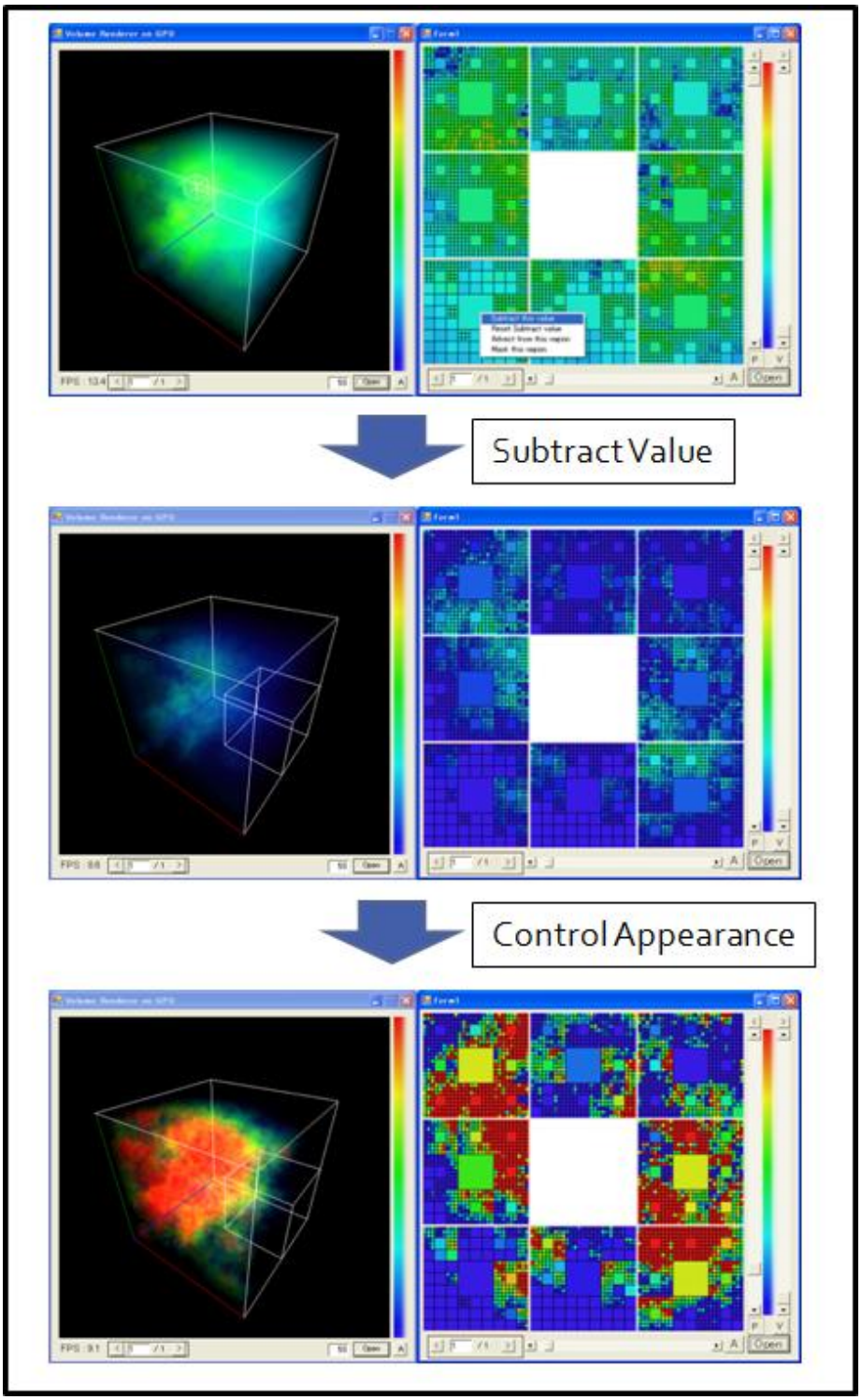

Fig. 40　減算処理を用いた可視化結果の調節

E)　時系列データのアニメーション表示

本手法ではボリュームデータに対し遮蔽のない 2 次元可視化が可能である．このような可視化は時系列データのモニタリングに適すると考えた．ボリュームレンダリングのような視点依存・遮蔽の発生する可視化手法を用いた場合視点およびパラメータを頻繁に変更しながらデータを眺める必要があるため時系列データの閲覧に要するインタラクション・コストは大きい．インタラクション・コストはアプリケーションの操作や表示変更に際して生じるコストの総称であり，Information Visualization におけるインタラクション・コストの定義および議論は Lam の研究に詳しい[30].

また，視点変更を伴う可視化では複数の時点のデータ同士の比較が容易ではない．一方で視点依存性の無い本手法は static な視点からボリューム全域を一覧することが可能であり，そのためデータの比較も容易である．

時系列データの可視化を可能にするために我々のシステムは複数データの読み込みをサポートしている．読み込んだボリュームデータの総数は 2D Window 下部に数字として表示され，現在表示しているデータの番号がその左側に表示されている．例えば全部で N 個のデータが読み込まれており，現在可視化されているデータがその n 番目のデータに当たる場合，表示は n／N となる．その表示の左右には矢印ボタンが備えられており，このボタンをクリックすることで 1 つ前のデータや次のデータへ表示を切り替えることができる．ウィンドウで一度に可視化できるデータの数は 1 つのみであるため，複数データ比較のためには逐一ボタンを操作して表示を切り替え確認することになる(Fig. 41).

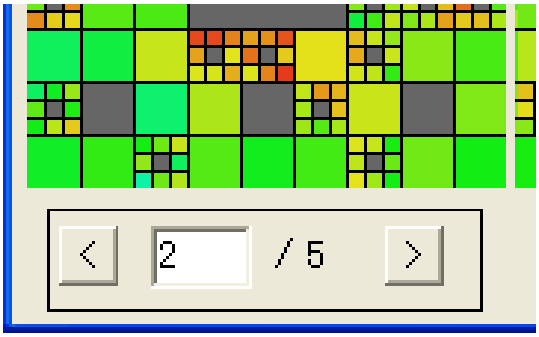

Fig. 41　複数データの読み込み表示およびデータ番号切り替えボタン

以下 Fig. 42 に時系列データの可視化例を挙げる．

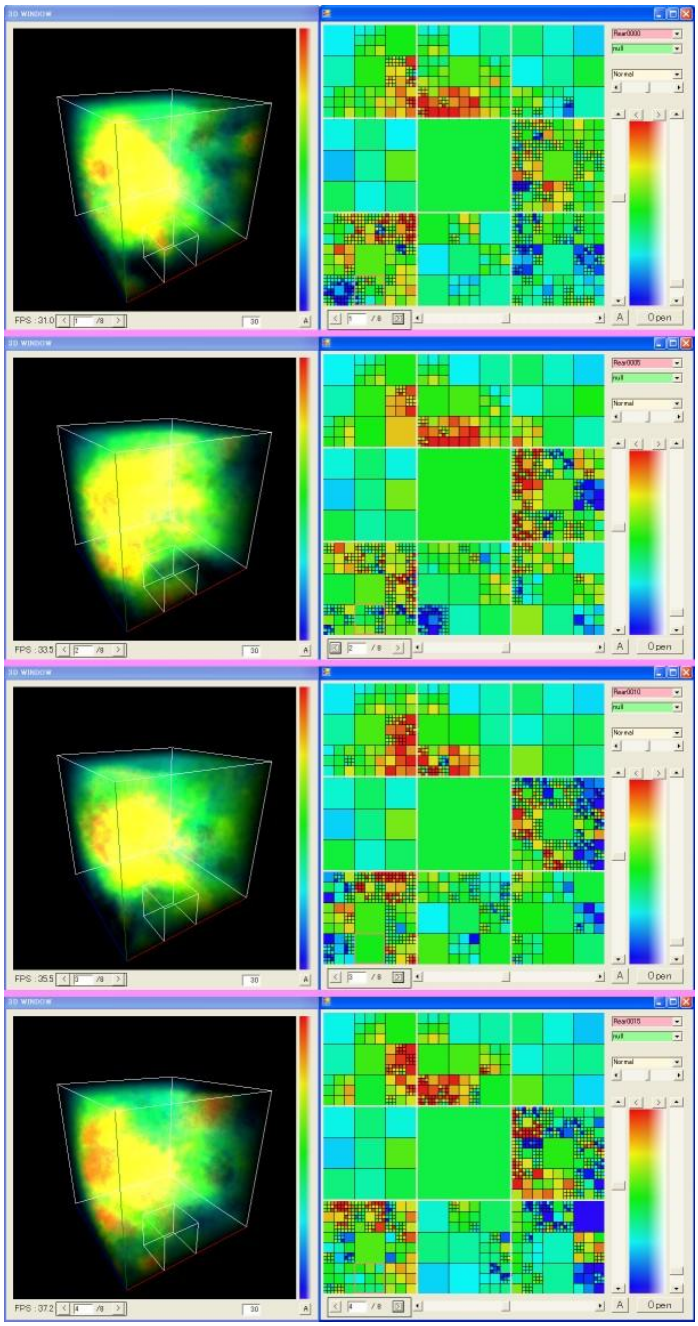

Fig. 42　時系列データの比較

視点依存性のない 2 次元展開可視化では同一ボクセルにおける物理量の時系列変化を静的に観察することができる．一方で 3 次元空間連続性が保たれたボリュームレンダリングは物理量の空間的な移動の様子を認識する目的に適する．

F)　ズームイン・ズームアウト機能

提案手法であるボクセル 2 次元展開の抱える問題点として高解像度データを dot-to-dot で

表示しようとすると膨大な画面表示領域が必要となってしまうという問題がある．解像度 R のボリュームデータを dot-to-dot で表示するためには少なくとも 3^R pixel×3^R pixel の画面領域が必要であり，一般的なディスプレイ解像度を鑑みるに R=6（729 pixel×729 pixel）程度の表示サイズが上限である．そこで我々は特定の領域にズームインして可視化対象領域を限定することでこの表示解像度制限を解決した．この機能は 2D Window においてズームインする矩形領域を右クリックで選択し，メニューから Zoom into this region を選択することで利用できる．本手法の特徴であるボクセル・矩形の対応関係より，表示領域をある矩形に限定することは可視化対象をあるボクセル（およびその内部に属するボクセル）に限定することに等価である．このズームイン機能の活用例を以下の Fig. 43 に示す．

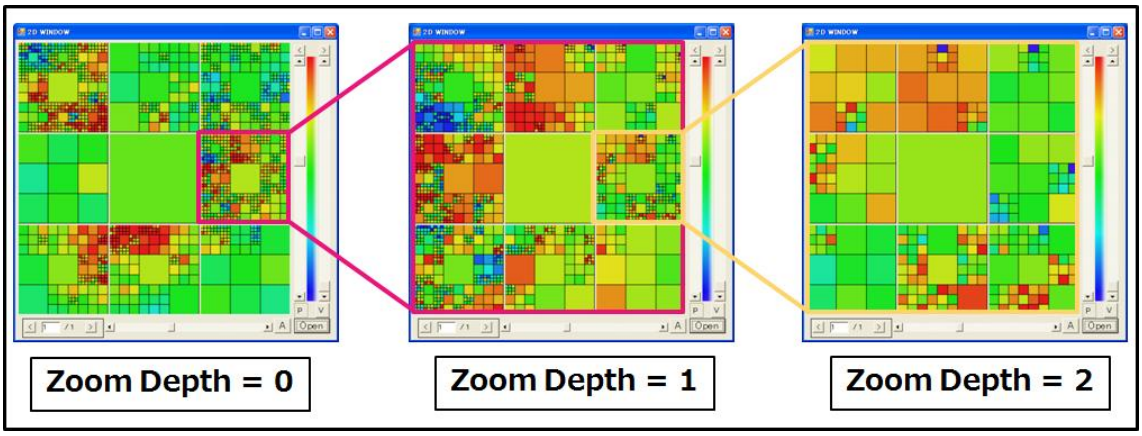

Fig. 43　ズームイン機能

同様に特定領域からひとつ上の階層へズームアウトする機能も右クリックメニューから利用できる（Zoom out from this region）．このズームイン・ズームアウト機能はＳＣの自己相似性―すなわち拡大・縮小しても幾何学形状が変わらないという特性を有効活用した機能である．

現在どのスケールでボリュームデータを閲覧しているかを表す指標として View Resolution という概念を導入し，R_{view} と表す．始めにボリューム全域を見渡しているときの View Resolution を 0 として（R_{view}=0），そこから 1 階層ズームインして可視化領域をある R=1 のボクセル内部に限定したときの R_{view}=1 とする．以下同様に 1 階層ズームインすることによって R_{view} が 1 増加すると考える．また，1 階層ズームアウトすれば R_{view} は 1 減少する．

R=10 などの高解像度ボリュームデータを可視化する際，一般的なディスプレイの画面領域では全ボクセルを dot-to-dot で表示することはできない．従って 2D Window サイズに応じた解像度のボクセルまでを描画することにする．この描画解像度を Draw Resolution という指標を導入して R_{draw} で表す．本研究の例では 2D Window の描画領域を(486 pixel×486 pixel)とし，R_{draw}=4 とした．このとき R_{view}=0 にてボリューム全域を表示する場合，

R=4 のボクセルまでが可視化され，各 R=4 ボクセルは 2 pixel×2 pixel の矩形として描画される．さらに，ズームイン機能を用いて R_{view}=1 へ 1 階層ズームインした際にはより 1 階層下の R=5 ボクセルまで描画する．すなわち View Resolution が R_{view} であるとき，可視化結果には $R=R_{view}+R_{draw}$ のボクセルまでが描画されるということになる．このようにしてズームイン機能とマルチレゾリューション可視化を組み合わせることで有限の画面領域内で高解像度ボリュームデータの可視化を可能にした．本来であれば高解像度のデータになる程可視化に伴う負荷は高くなるが本手法では可視化領域にズームインしていくことで描画量はほぼ一定に抑えられる．フラクタルの特性により表示情報量を一定に保つアイディアは Koike らの Fractal Views に，ズーミングを取り入れたマルチレゾリューション可視化は Microsoft Deep Zoom などの先行例を参考にした．

3-3　ベクターデータの可視化

ベクターボリュームデータに対しても同様に 2D・3D 協調可視化を実現する．入力データにはスカラーデータの場合と同一の CFD 出力データから自動車モデル後方部にあたる領域の流れ場の流速分布を R=5 の時系列ベクターボリュームデータとして抽出したものを用いた．ある位置における流速はその位置に仮想的な流体粒子を配置したときの単位時間あたりの移動量であり，このような物理量の可視化には流線を用いた描画が効果的である．従ってここではベクターデータに対する 3 次元可視化手法として流線を用いた．

3-3-1　２次元展開

ベクターデータを格納したボクセルの再分割においても，格納値の分散値を指標として用いる．スカラーボクセルの 2 次元展開の項でも述べたように，ある解像度 R のボクセルは 8^{Rmax-R} 個の R_{max} ボクセルを内包する．従ってこのボクセルに格納される値はこれら 8^{Rmax-R} 個のベクター値の平均ベクトルである．次にあるボクセルの分散値を次のように計算する．

$$\sigma^2 = \frac{1}{Num}\sum_{i=1}^{Num}\|x_{ave} - x_i\|^2 \qquad (8)$$

ここで Num はそのボクセルが内包する R_{max} ボクセルの総数（8^{Rmax-R} 個），x_iは内包する各 R_{max} ボクセルに格納されたベクター値，x_{ave} はそのボクセルに格納された平均ベクターを表す．以下スカラーデータの場合と同様に R=0 における分散値を$\sigma^2{}_{ref}$ とし，その値に従って閾値thを計算し再分割処理および 2 次元展開を実行する．

3-3-2　ベクター値から色への変換

各ボクセルに格納されたベクター値を対応する矩形の色として可視化することで 2D Window での可視化が完了する．スカラーデータではこの色変換に 1 次元テクスチャを用

いたがベクターデータの場合はより直接的にベクターの成分{u, v, w}をそれぞれ独立に矩形色{R, G, B}に変換する．具体的には以下の Colorization Function を用いる．

$$c = (0.5, 0.5, 0.5) + \left(\frac{u}{2.0 \cdot u_{max}}, \frac{v}{2.0 \cdot v_{max}}, \frac{w}{2.0 \cdot w_{max}}\right) \qquad (9)$$

ここで u_{max}, v_{max}, w_{max} はそれぞれ可視化結果表示の赤，緑，青成分をコントロールするための変数である．もしベクター値の u 成分が u_{max} よりも大きければ変換後の矩形色の赤成分は 1.0 である．一方で u 成分が-u_{max} よりも小さければ赤成分は 0.0 となる．このような Colorization を行えば色分布がそのまま速度分布を表すためボリュームの全体的・局所的速度分布を浮き彫りにすることが可能である．以下の Fig. 44 に可視化例を示す．この Fig.3-9（右）のようにＳＣ全域がピンク(r=1.0, g=0.5, b=0.5)で表現されている場合これは流れ場全域の特性を表し，全体的に X 軸正の方向の流れが支配的であることが分かる．

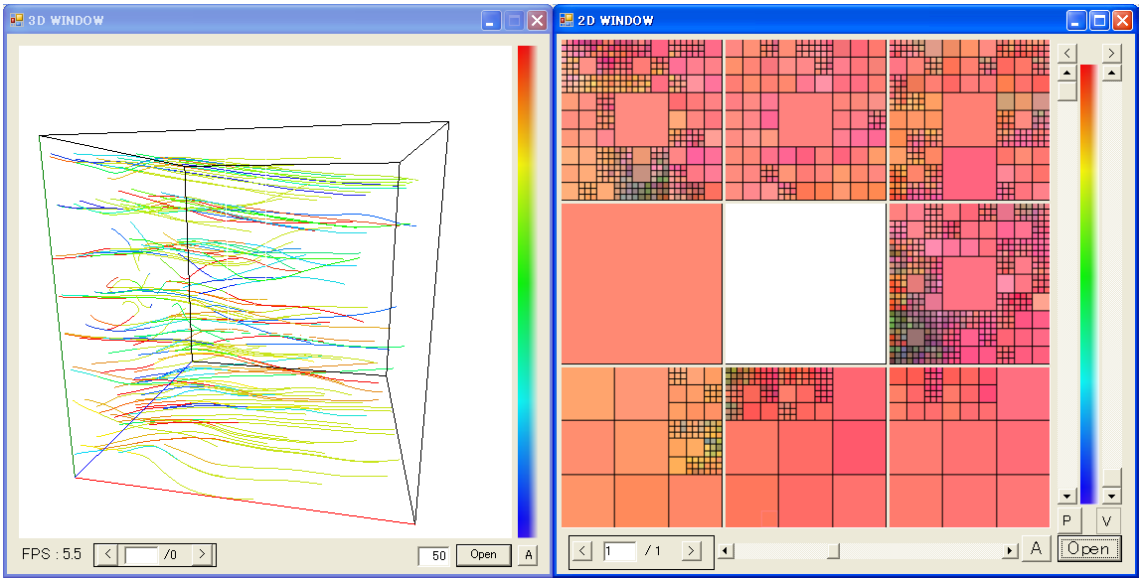

Fig. 44　ベクターデータの可視化結果

3-3-3　流線描画

3D Window では 3 次元的に流線を描画することでベクトル場の特性を可視化する．流線の描画は空間上のある点に始点を設け（Seeding），その点から周辺の速度場に沿って仮想的な粒子を移流させることによって描画可能である．この移流は始点から前方および後方に行うことができる．後方に移流させる場合には正負の符号が逆の速度場を仮定し，前方に移流させる場合と同様に微小時間間隔ごとに粒子座標を更新してゆけばよい．この流線の描画にも DirectX 9.0c API を利用した．

３-３-４　機能と操作

2 次元展開と流線描画を相互運用したインタラクティブな可視化を実現するためにいくつかの有用な機能を実装した. スカラーデータ可視化の項で挙げた機能のうち A), B), E), F) の機能はベクターデータ可視化においても同様に利用可能である．ここではそれに加えベクター可視化のためのいくつかの機能について説明する.

G) 指定した領域に始点を与え流線を描画

流線描画のための Seeding を半自動的に行うアプリケーションは多いが本手法ではＳＣを入力インターフェイスに用いた手動での Seeding を可能にする．これはＳＣ上である矩形領域を指定することで，対応するボクセル領域に対し Seeding を行うというものである．この機能は右クリックメニューから Advect From This Region を選択することで利用できる．この様子を Fig. 45 に示す.

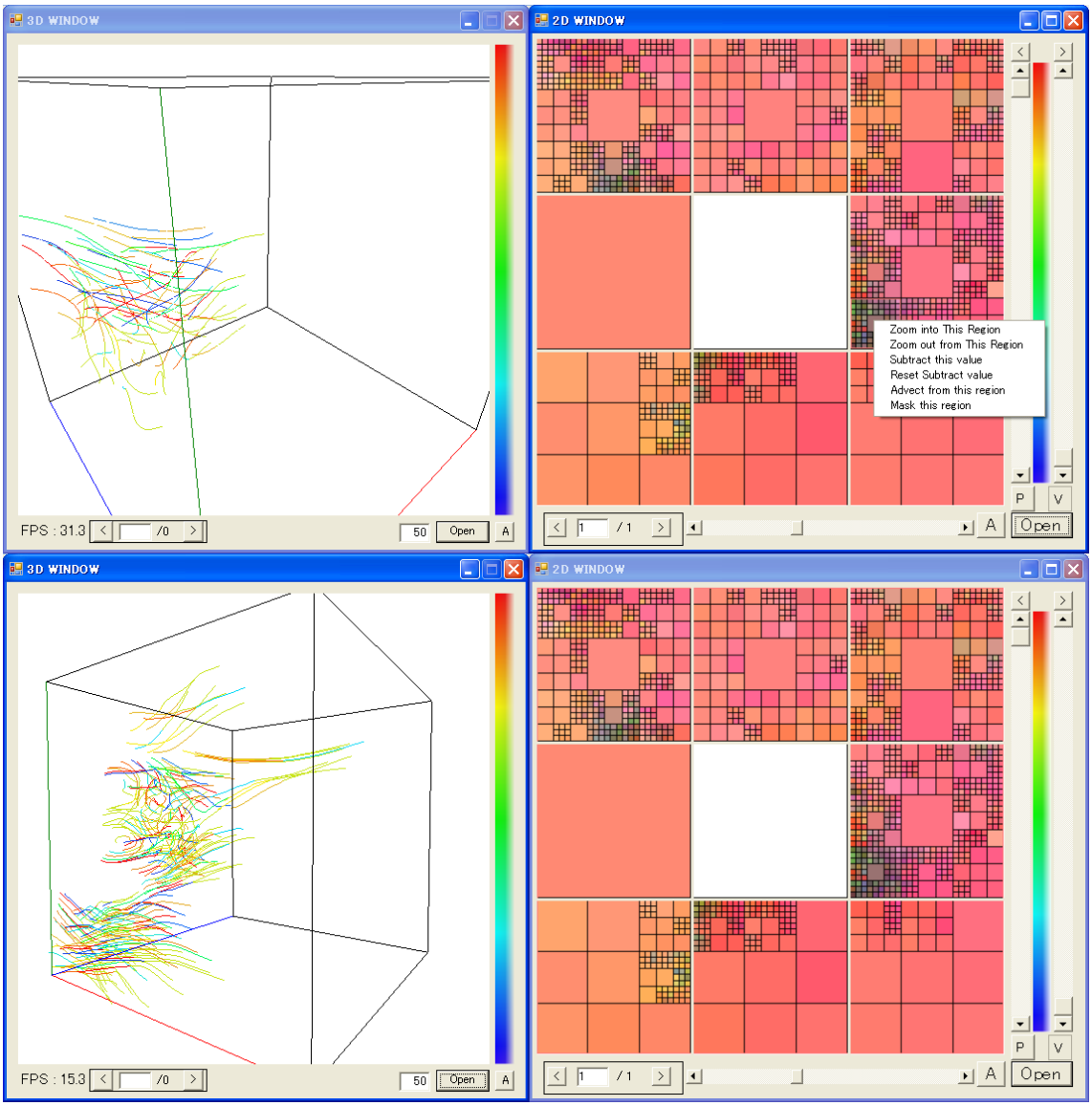

Fig. 45　流線の seeding を行うことで特徴領域の流れを可視化した例

このようにユーザーが Seeding したい領域を直接指定することでユーザーの意図に沿った流れ場の可視化が実現できると考えた.

2 次元展開による可視化結果はこの Seeding 対象領域を発見するために活用できる．例えばより細かく分割されている領域や周囲とは明らかに速度ベクトルの向きが異なる領域は 2 次元展開により遮蔽なく可視化されているため発見が容易である．この注目領域の発見をより実用的にするため，ベクトルボリュームデータが入力された際には渦度の絶対値をスカラーボリュームデータとして保持し，このデータの可視化も行えるようにした(Fig. 46)．一般に渦度の絶対値の高い領域は流れ場が複雑であることが多く流線可視化の対象になりやすい.

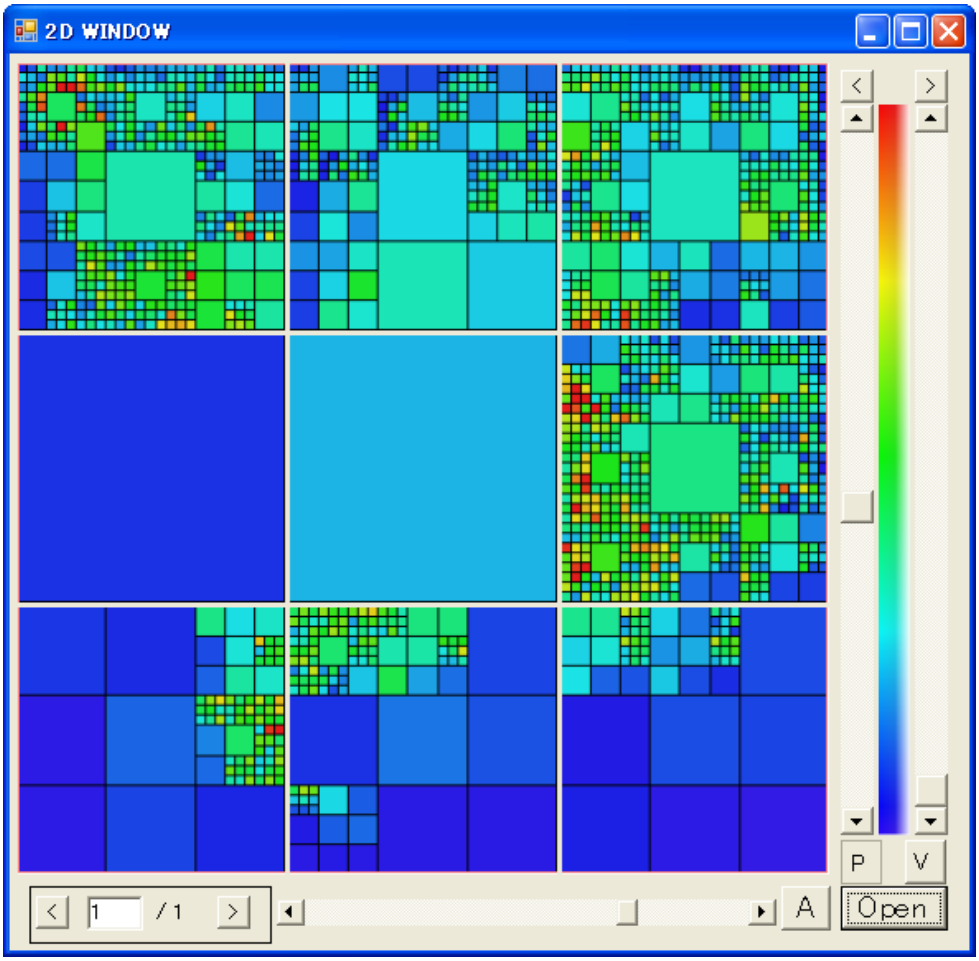

Fig. 46　渦度マップ

第４章　考察

ここでは 3 章の内容を踏まえた考察を行う．

４-１　可視化手法およびアプリケーションに関する考察

3 章では 2 章で提案した新しい可視化手法を実際に可視化システムとして構築した．また 2 次元展開可視化手法と既存の 3 次元可視化手法のコラボレーションを実現した．このコラボレーションのためには両手法間のインタラクションが不可欠である．また，ユーザーと可視化アプリケーション間のインタラクション機能も実現する必要がある．そのため我々は独自の可視化システムを構築しインタラクティブな可視化のための様々な機能を実装した．ここでは我々が開発した可視化システムにおいて 2 章冒頭に挙げた新しい可視化手法の開発に係る方針 1)～9)がいかに盛り込まれているかについて考察する．

1)　情報可視化的発想を取り入れた新しいボリュームデータ可視化手法の開発

空間分割による高密度充填，マルチレゾシューション可視化など情報可視化で用いられる発想を多く取り入れた．Focus+Context に関しては 3D Window を Focus Window として用いることで実現可能である．これには第 3 章で示した機能のうち**(C) 可視化領域の直接的な選択**を利用する．Focus 部位をボリュームレンダリングで可視化し，Context は 2D Window で SC による全域可視化により把握できる．

2)　提案手法は遮蔽・視点依存性のない２次元可視化手法

本来 3 次元構造であるボリュームデータを 2 次元平面に展開することで遮蔽性を無くし，視点依存性の無い２次元可視化を実現した．一般に 2 次元可視化は 3 次元可視化に比べ人間が可視化結果の認識に要する負荷が低い．本手法では回転等の視点変更操作が不要であるため複数のデータを比較する際のインタラクションコストも低減される．また描画に要する負荷もボリュームレンダリングと比較して低い．なぜなら 2 次元展開処理は単純にボクセルの座標変換を行っているに過ぎないためである．従ってノートＰＣなどの比較的低スペックなコンピュータであっても 2 次元展開による可視化手法を利用することが可能である．

3)　ボリュームデータの一断面のみでなく全域を２次元的に表示可能な手法の開発

ボクセルを 2 次元平面に展開することでボリューム全域にわたり情報を遮蔽なく閲覧することが可能となった．これはボリュームデータの全体像を外観的に把握する上で重要な特徴である．

4) ボリュームデータ構造を階層データとみなし階層的情報可視化を行う

ボクセル分割により Octree を構築し，ボリュームデータに階層データ構造をもたらした．各階層のボクセルはより下位の階層のボクセルを内包すると考える．したがって各階層のボクセルには下位ボクセルに格納されたデータ値の平均値が格納され，これにより階層的可視化が実現可能となる．

5) フラクタル図形を活用したマルチスケール可視化手法の開発

Octree を 2 次元に展開するにあたりフラクタル図形であるＳＣを活用．ＳＣの自己相似性と Octree の階層構造を結びつけることでマルチスケール可視化を実現した．

6) ２次元可視化結果をボリューム全域の地図として活用

Colorization Function によりボクセル値を色として表現した．これを 3)の全域表示と組み合わせることにより 2 次元可視化結果はボリューム全域のデータ値分布を表現する地図とみなすことができる．9)に述べる 2 次元・3 次元コラボリューション可視化ではこの地図を 3D 表示領域を指定するためのガイドマップとして用いることができる．この遮蔽の無い全域表示により特徴的な値を持つ領域が一目で把握でき，その部位をターゲットにより詳細な可視化を行うといった段階的手続きが促される．このように 2 次元展開による可視化を取り入れることで効率的な可視化ワークフローを構築できる可能性が生まれた．

7) ２次元可視化結果を入力インターフェースとして活用

ボリュームデータを構成するボクセルとＳＣ上の矩形は 1 対 1 に対応しているためＳＣ上で任意の矩形を選択することによって対応するボクセル，すなわち 3 次元領域を指定することができる．このように 2 次元可視化結果を入力インターフェースとして利用することによって 3 次元ボリュームおよびボクセル値へのダイレクトなアクセス・操作が可能となった．本システムではこのインターフェースを活用したいくつかの機能を提供している．

8) 提案する情報可視化的手法と既存の科学的可視化手法との相互運用

本来科学的可視化の適用分野であるボリュームデータに情報可視化的発想を取り入れた提案手法を適用することで科学的可視化分野における情報可視化手法の適用を試みた．また，ボリュームレンダリング・流線といった既存の 3 次元科学的可視化手法と提案手法を組み合わせ，相互運用する試みを行った．複数の特性の異なる可視化手法を併用することでデータの特徴を多角的な観点から捉えることが可能となった．

9) 提案する２次元ベースの表示手法と既存の３次元手法とのコラボレーション

既存の 3 次元科学的可視化手法と 2 次元可視化手法である 2 次元展開によるコラボレーション可視化システムを構築した．見せ方の異なる手法を併用することでデータの特徴領域・重要部位の見落としが低減されることが期待できる．

以上のように提案手法には 1)～9)の各項目が全て盛り込まれており，当初の方針に沿った可視化手法・アプリケーションを構築することができた．

本手法によりもたらされるメリット・実現された可視化機能は上に述べた通りである．一方で本手法にもいくつかの欠点が存在する．2 次元展開のみを用いる場合，フラクタル展開により本来隣接した領域が SC の矩形上に離散的に展開される．そのため実際には 3 次元的に連続な値の分布があっても 2 次元展開後はその連続性が保たれないという問題がある．元来 R=1 の 2 次元展開ルールにおいてさえボクセルの隣接関係は保たれていないのであるからこの問題は必然的に発生する．しかしこの問題は必然であると同時に不可避である．その理由を 3 次元から 2 次元への展開という類似のロジックであるポリゴン UV 展開における連続性の破壊を例にとり考察する．

ポリゴンの UV 展開は 3 次元ソリッドサーフェイスであるポリゴン表面に平面テクスチャを貼るために，多角形（一般には 3 角形・4 角形）ポリゴンを 2 次元平面テクスチャ上に展開する手法である．この際テクスチャをポリゴン表面に継ぎ目なく貼り付けるためにはポリゴンモデルの 3 角形・4 角形面の連続性をできる限り保ったまま展開することが要求される．しかしながら閉じた 3 次元ポリゴンモデルはたとえ凸包（convex）であっても各面の連続性を完璧に保ったまま UV 展開することは一般に不可能である．UV 展開は 3 次元座標（x, y, z）から 2 次元座標（u, v）への展開とはいえ厳密にはモデル表面（2 次元）から 2 次元平面への展開である．にもかかわらず要素の連続性は破壊されてしまうことは避けられず，ここにその原因が 3 次元構造と 2 次元構造の位相的性質の相違にあることが明確となる．ボクセルの場合内部構造を含むため 1 つのボクセルの隣接ボクセルの数は最大で 6 個であり，一方で SC の矩形は最大でも 4 個の隣接矩形しか持ちえない．このように 3 次元構造を 2 次元平面に展開する際に連続性が損なわれる問題は不可避であると言える．

別の欠点として SC のある矩形がどの 3 次元領域に対応するかという関係が直感的には認識困難であるという問題がある．基本展開ルールは単純であるが，それを再帰的に施すことで展開ルールは複雑化し，直感的に把握することは難しくなる．

これら 2 つの大きな欠点を解決するために SC による 2 次元展開を用いた可視化に加え，空間連続性を保ったまま可視化可能な既存の 3 次元可視化手法を併用する必要があると考えた．また，情報可視化の観点から考えて 2 次元可視化結果はそのまま入力インターフェースとしての利用が可能である．これは 2 次元可視化と 3 次元可視化のコラボレーションが先に挙げた欠点の克服以上のメリットをもたらす可能性を示唆している．従って 3 章ではインタラクティブなコラボレーション可視化システムの内容・実装・可視化例について実際にシステムを開発することで具体的に議論した．

4-2　スカラーデータ可視化に関する考察

スカラーデータの可視化ではSCを用いた2次元展開手法とボリュームレンダリングを連動させることで異なる2つの観点からボリュームデータを閲覧することを可能にした. また, 可視化に関わるパラメータにも共通性をもたせ, Min Max Bar を操作することによって双方の可視化結果が連動して変化するような実装を行った. また SC をインターフェースとして用いたいくつかの機能を実装しインタラクティブな可視化を実現した.

まず v_{max}, v_{min} による可視化像の調節について考察する. この2つのパラメータは Min Max Bar により調節可能であり, 可視化像の色コントラストを調節することを目的としている. あるボリュームデータを入力した際のデフォルトの可視化像（v_{max}=1.0, v_{min} = 0.0）と調節後の可視化像を Fig. 47・Fig. 48 に示す.

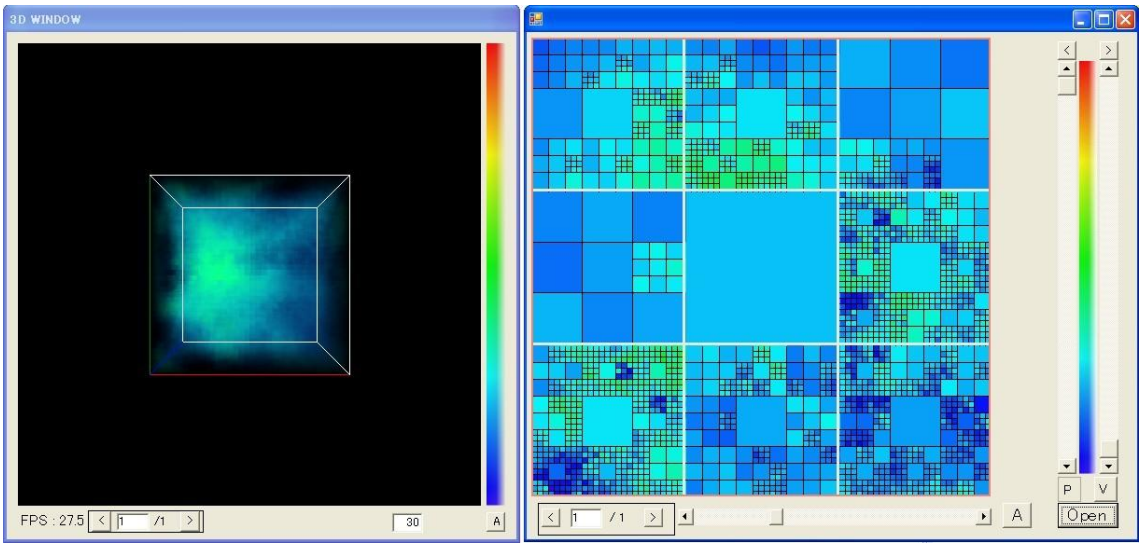

Fig. 47　デフォルトのスカラーデータ可視化像

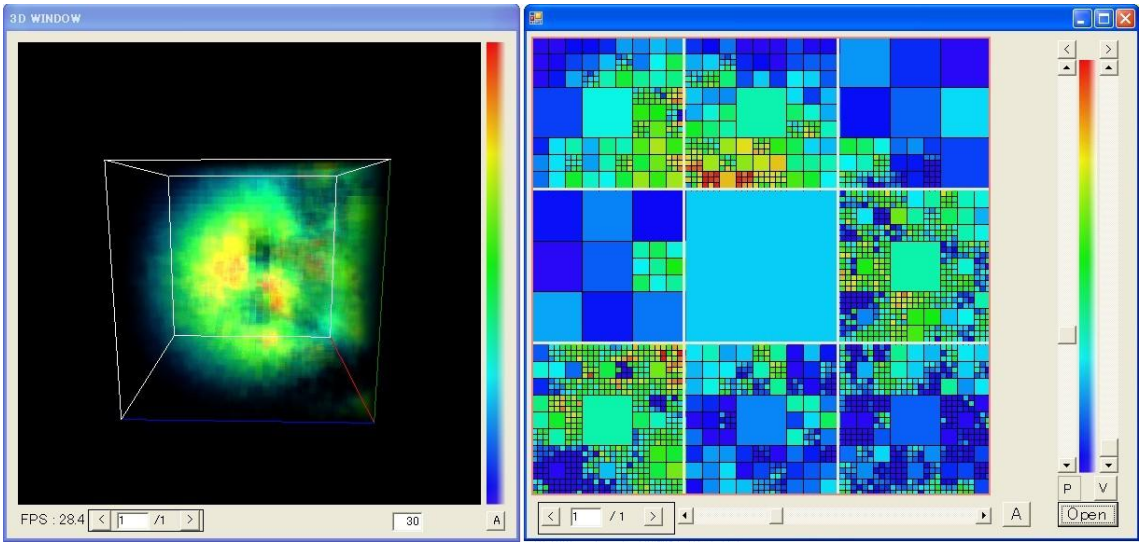

Fig. 48　vmax, vmin を用いて調節した可視化像

調節前の可視化像はテクスチャのダイナミックレンジを使い切っておらず全体的に低コントラストな画像となっている．調節後は v_{max} を下げることでデータ値の高いボクセル・矩形にテクスチャの赤領域を割り当て，v_{min} を上げることでデータ値の低い領域を不可視にした．このように可視化像を調節することができる．

ボリュームデータ同士の比較が容易に行えることは重要である．2 つのボリュームデータを入力し，同じ v_{max}, v_{min} の値で可視化してみる．下の Fig. 49 は 2 つのボリュームデータ 1 ・ 2 を可視化したものである．

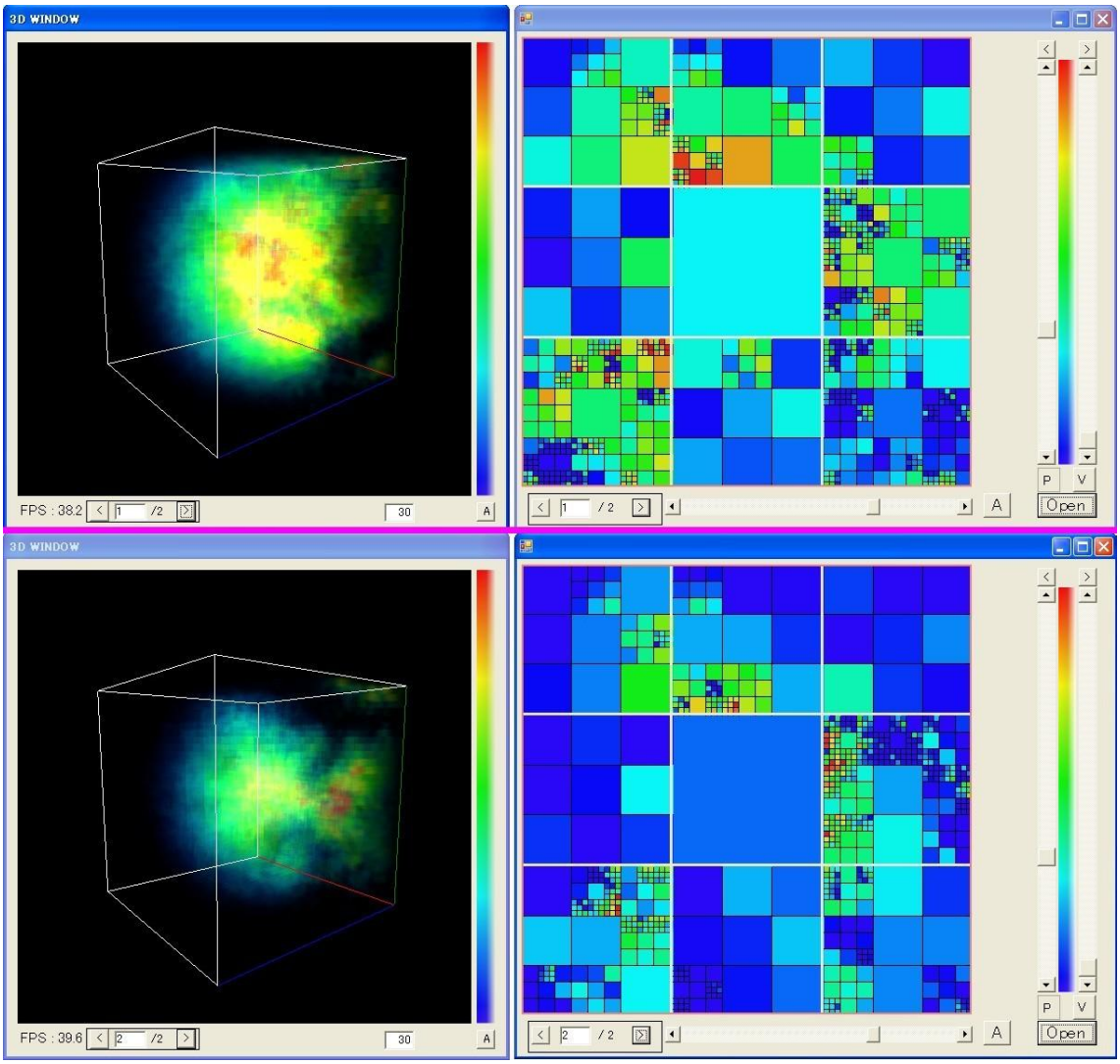

Fig. 49　2 つのボリュームデータの比較　上：データ 1　下：データ 2

ボリューム全体の平均値がデータ 1 の方が大きいことは 2D Window の中央同士を比較すれば一目瞭然である. 両データで大きな値をもつ部位が異なることも 2D Window 同士を比較することでわかる. その部位が 3 次元的空間においてどの位置を占めるのかを把握するためには 3D Window 同士を比較する必要がある.

ボリュームデータ内で局所的もしくは全体的な比較を行いたい場合もある. この場合には Subtract Value 機能を利用する. これは右クリックメニューから利用できる. 例えば次の Fig. 50 は全体の平均値を v_{sub} に指定することで, 平均より高い値をもつ領域のみを可視化結果に反映させることができる.

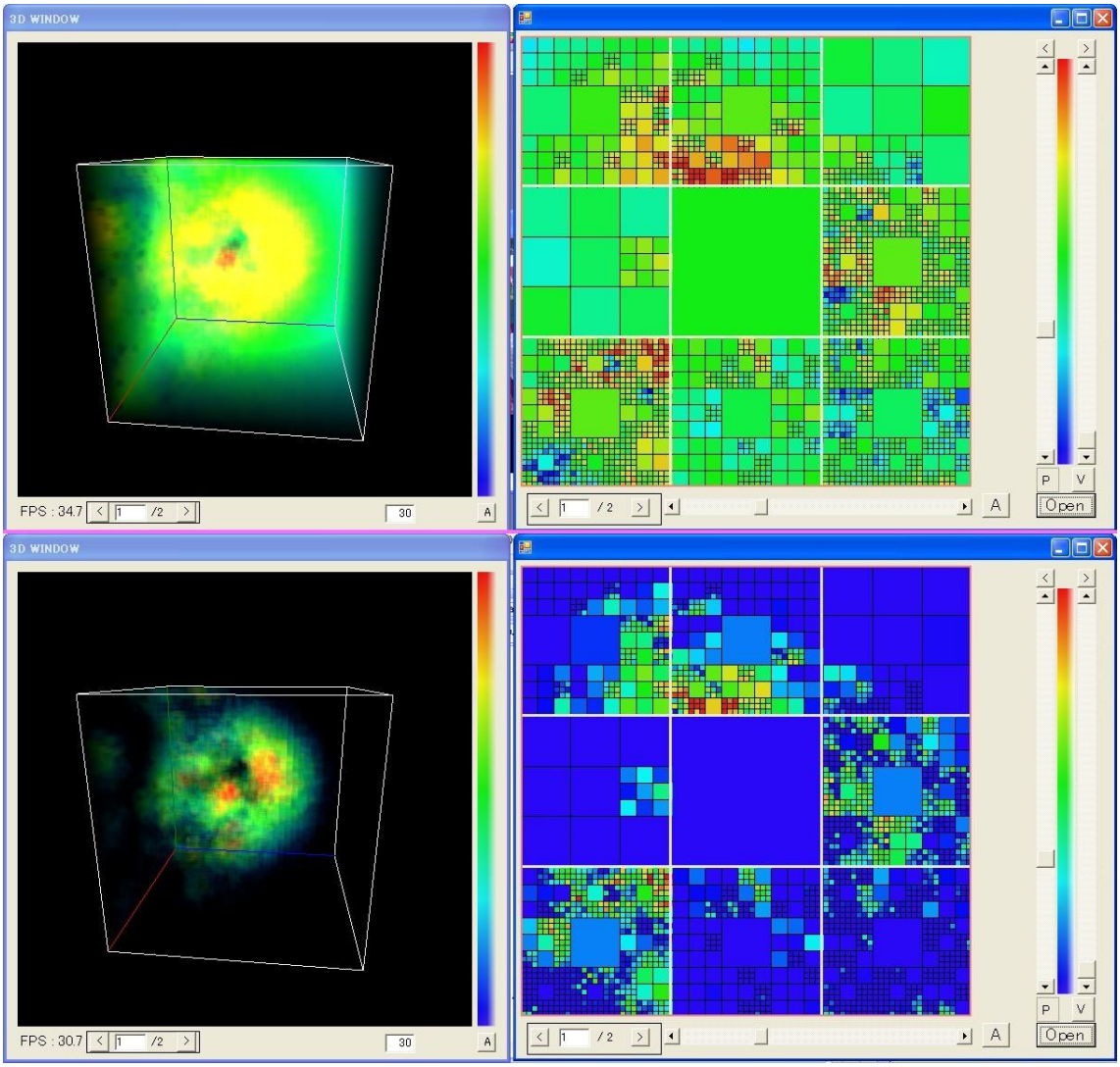

Fig. 50　ボクセル値の減算

2D Window の可視化結果は平均値よりも高い値をもつボクセルがどの程度存在し，どのように分布しているかを表す．ただしそれらのボクセルが構成する 3 次元的な実像は 3D Window によってしか把握できない．

2D Window の矩形は個々のボクセルを表現する．2D Window に表現された個々のボクセルを 3 次元空間に並べ直し，ある視点から眺めれば 3D Window の表示になる．2 次元展開による表示はボリュームデータの地図であり，1 種のカタログである．それはボリュームデータがどのような構成要素によって構成されているかを表しており，矩形色は相当する値をもったボクセルの実在を可視化主体に告げる．2 次元展開によって可視化主体はそれらのボクセルの存在を確信をもって認知することができるが 3 次元的な構造・形状を把握することはできない．

4-3　ベクターデータ可視化に関する考察

ベクターデータの可視化では CFD により得られた速度場データを用いた．速度場は位置座標を時間微分したものであるので，3 次元可視化手法には流線を用いた．また，SC をインターフェースとして活用し，任意の領域に流線の始点を seeding することを可能にした．ここでは seeding を施す領域（特徴領域）を発見するための地図として 2D Window が活用されている．

2D Window によって特徴領域を発見させるためには，その発見を促進させるような可視化結果を提示する必要がある．すなわち SC の分割と Colorization である．このうち SC の分割はスカラーデータの場合と同様に分散値を用いた．一方で 3 次元ベクトルの Colorization には 3 次元テクスチャを用いる必要がある．1 次元テクスチャを 3 つ用意する方針も考慮したが各軸成分の色が排他でなければならず，この方針は棄却した．本研究では速度ベクトル成分をそのまま矩形色の RGB 色に変換し，各軸の独立性を保った(Fig. 51)．

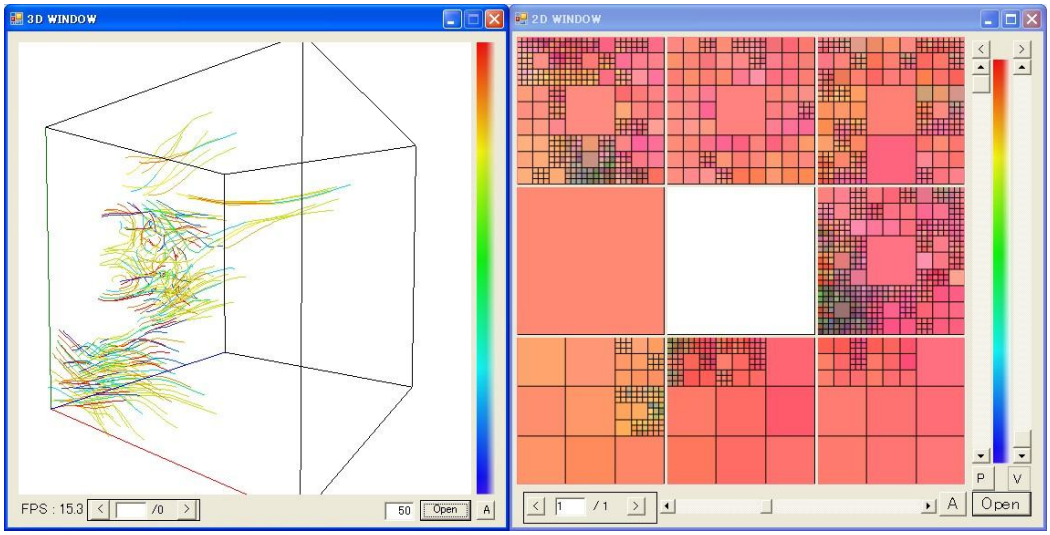

Fig. 51　ベクターデータの可視化

この Colorization の欠点は速度ベクトル同士の関係性を色に落とし込むことが困難な点である．例えばあるベクトル v とその逆向きのベクトル-v があるとき，色情報のみからそれらが逆方向かつ長さの等しいベクトルであることを認識することは非常に困難である（Colorization 結果としては v が (r, g, b) = (0.5 + r, 0.5 + g, 0.5 + g) として可視化されるなら，-v は (r, g, b) = (0.5 - r, 0.5 - g, 0.5 - g) となる）．同様に互いに垂直なベクトル，方向は異なるが長さの等しいベクトルなども容易に判別できない．ただし長さ・方向双方が似たベクトル同士は類似の色に変換されるため判別可能である．今回は RGB 色空間を用いたが HSV・CMY 等の色空間も利用可能であり，理想的にはこれらを比較検討して用いることが望ましい．

2D Window で任意のボクセルを指定することで seeding を行い，3D Window に流線を描画する．Seeding は指定したボクセル領域内にランダムな始点を配置する．本研究の実装では描画した流線の評価は行っていない．流線による可視化をより効果的なものにするためには似たような流線を複数描画することを避けて認知負荷減らす，流線同士のオーバーラッピング・インターセクションによる認知負荷を低減するといった工夫が必要となる．また seeding 領域を自動的に抽出し流線描画を自動化する研究も行われている．こうした高度な流線描画アルゴリズムを導入することでより効率的な可視化が可能となるであろう．

4-4　その他考察

4-4-1　本手法における制限について

この項では本手法における制限事項（Limitations）について説明する．本研究で提案した SC によるボクセル 2 次元展開の制限のひとつに，対象とするボリュームデータを $2^n \times 2^n \times 2^n$ ボクセル（n は自然数）のものに限定しているという点がある．この制限による生じる問題は大きく分けて 2 つある．まずは自由なボクセル数をとることができない点．もし n=3 で解像度が足りなければ n=4 のボリュームデータを用いざるを得ず，8 倍のデータ量が要求される．データ量の増加はストレージを圧迫するだけでなく占有メモリ量・データ処理速度・描画速度などに影響を及ぼす．2 つ目の問題点は各軸方向のボクセル数が等しい必要があるという点である．ただしこれは必ずしも可視化されるボリュームの各軸方向の長さが等しくなければならないという意味ではない．なぜなら各ボクセルの各辺の長さは軸ごとに異なっていても構わないからである．すなわち非立方体領域を可視化するためには領域形状と相似な直方体ボクセルを用いればよい．ただしボリューム領域がある軸方向にだけ極端に長い・もしくは短い場合には単一の SC に展開すると直感的な理解を阻害する．そのような場合，できるだけ立方体形状に近い複数のボリュームに分割し，複数の SC を用いてそれぞれのボリュームを可視化することが必要となる(Fig. 52)．

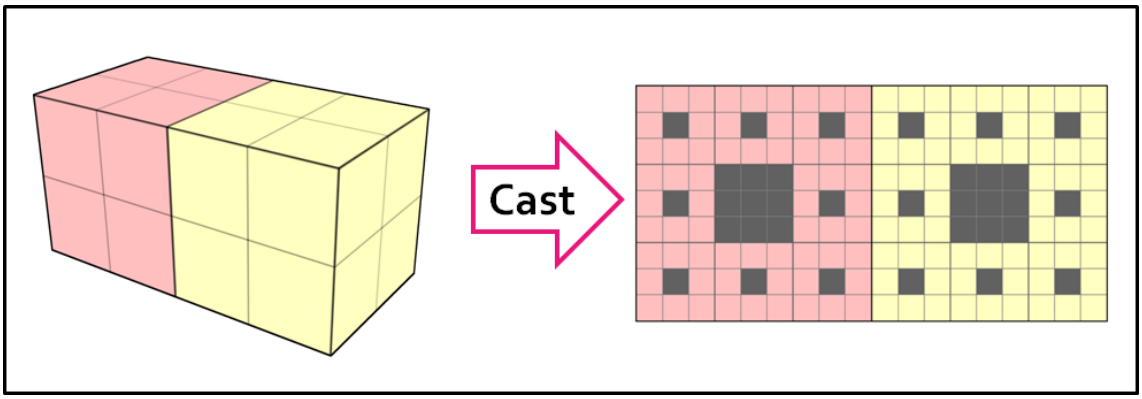

Fig. 52　一方向に長い領域を分割し，複数の SC を用いて 2 次元展開を行った例

４-４-２　可視化パフォーマンスについて

本研究で構築した可視化アプリケーションのパフォーマンスについて考察する．アプリケーションでは 2D Window と 3D Window の 2 つのウィンドウにそれぞれ異なる可視化手法による表示を行う．スカラーデータを可視化する場合 3D Window にはボリュームレンダリングによる可視化を行う．ボリュームレンダリングの計算負荷は描画ピクセル数とサンプリング数に比例する．本研究では HLSL を用いて GPU ボリュームレンダリングを実装することでパフォーマンスの高いレンダリングを実現した．NVIDIA GeForce7900GS を使用した可視化において 500 pixel×500 pixel の 3D Window にサンプル数 30 samples / pixel で 54.0 fps 以上のフレームレートを達成しており実用上問題のないパフォーマンスであると言える．一般に投影する ray の本数および ray 1 本あたりのサンプル数が一定であれば，ボリュームレンダリングのパフォーマンスはボリュームのボクセル数が増加しても低下することはない．ただしサンプル数が一定なままではサンプリングされないボクセルが発生するためボリュームデータのデティールが可視化結果に表れずレンダリングの品質が低下する．この問題を避けるにはサンプル数を増加させればよいが当然その分だけレンダリング時間は増加する．これはピクセル数についても同様である．高品質なレンダリングを行うためには 1 ピクセルを細かなサブピクセルに分割し，各ピクセルに対し複数本の ray を投射する（スーパーサンプリング）か，もしくはレンダリング領域を大きくする必要がある．どちらの場合も計算時間は増加する．このように高解像度データのレンダリングにおいて品質を一定に保つためには計算時間の増加は避けられない．

一方で 2 次元展開可視化の場合計算プロセスは事前処理（preprocess）と描画処理（drawing process）の 2 つに分けられる．事前処理はボリュームデータを読み込み，Octree を構築し，そのノード情報を頂点情報として GPU の頂点バッファに転送するところまでである．描画処理は転送された頂点情報に基づいて矩形を描画する処理である．近年の GPU の処理能力の向上によりこの描画処理にかかる時間はきわめて短く，ほとんどの場合その描画時間は無視できるほどである．一方で事前処理は CPU 側で行う処理と CPU から GPU へのデータ転送処理が含まれるため比較的計算負荷の高い処理となっている．さらにこの事前処理の計算負荷はボリュームデータの解像度に依存する．なぜならボクセル分割に伴う Octree 構築には平均値および分散値の算出を何度も行わねばならず，ボクセル量が増加するにしたがってこの処理は高負荷なものとなるからである．ただし Split Ratio Bar が操作され分割細かさをコントロールするパラメーターaが変更されない限り Octree を再構築する必要はない．Min Max Bar の数値変更に関わる色計算は GPU 側で行われるためこのバーの操作によって頂点情報を GPU の VertexBuffer に再転送する必要も生じない．まとめると計算負荷のかかる Octree 構築を行う必要性が生じるのは Split Ratio Bar が操作された場合および新たなボリュームデータが読み込まれたときに限る．ここまでに採用した

実装を用いて処理速度の面で不自由なく可視化が行える解像度は R=6 程度までである.

第５章　評価

５-１　評価方針

ここでは提案手法である 2 次元展開による可視化手法と既存の 3 次元可視化手法の比較に焦点をあて，ユーザーテストを行うことで互いの特性・相違点・有効性を検証する．検証項目は

実験１）全体像の把握
実験２）局所特性の把握
実験３）特徴値の把握

の 3 点とし，各項目を独立して評価するために CFD から得られたデータではなく各項目の評価に即したテストデータを人為的に作成し用いる．

実験形式は次の通りである．まず入力データは全て R=6 のスカラーデータである．実験のためにデータ値分布の異なるボリュームデータを 20 例作成した．実験内容はこの 20 例のデータから試行ごとに 4 例を抽出し，指定した可視化手法による可視化結果を元にボリュームデータを比較するというものである．用いた可視化手法は 3 種類ありそれぞれ

- VR　（ボリュームレンダリング）
- SC Ⅰ（ブランク領域に何も可視化しない SC）
- SCⅡ（ブランク領域には平均値を可視化した SC）

である．次の Fig. 53 にこれら各手法による比較可視化の様子を示す。

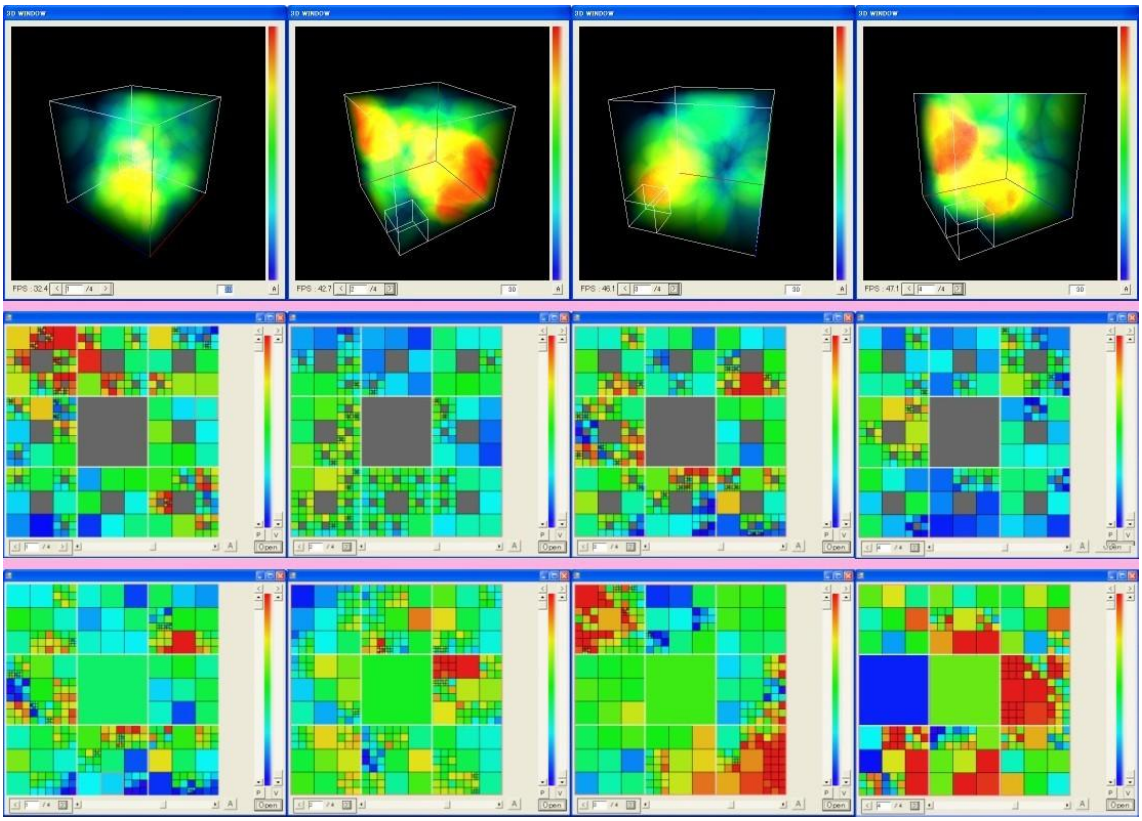

Fig. 53　上から可視化手法 VR, SC Ⅰ, SC Ⅱ を用いた比較

本実験の目的は各手法の定性的な評価である．具体的には，手法として VR を用いてデータ比較を行う場合と SC を用いて比較を行う場合で正確さや回答に要する時間がどう異なるかを実験により明らかにすることが目的である．そのため一度の比較試行には単一の可視化手法のみを用い，複数手法の協調可視化は行わない．可視化手法として VR, SC Ⅰ, SC Ⅱ を用いた場合のそれぞれ 3 通りの比較試行を行うことになる．各実験は 10 人の大学院生を被験者として計 90 回行った（3 手法×30 回）．

被験者は 4 例のボリュームデータの可視化結果を操作・比較検討し，それらの並べ替えを行う．並べ替えの基準となる指標は実験ごとに異なる．これを比較指標と呼ぶ．例えば実験 1 ではボリューム全域の平均値が比較指標である．操作は Min Max Bar・Split Ratio Bar およびボリュームレンダリングのカメラ操作に限定する．被験者の回答を正解と照らし合わせ採点する．各実験の前に被験者に対して 1 分程度の簡単な説明を行った．

配点方法を次のように定める．配点は比較指標が最大のデータ・最小のデータが正しく回答されているかに関して行う．比較指標の値が最大のデータの順位を正しく回答できた場合には 0.5 点を，比較指標値が最小のデータを正答できた場合にも 0.5 点を与え，合計 1 点を満点とする．ただし，比較指標値が最大のデータを誤って 2 位に順位付けした場合にも 0.25 点を，比較指標値が最小のデータを 3 位に順位付けした場合にも 0.25 点を与えるものとする．

回答用紙は Table 1 のような形式となっており，各空欄にデータ１～４の番号を比較指標値降順に回答する．

Table 1　ユーザーテスト回答用紙

	最大 ⟷ 最小			
Q1				

例えば，正しい答えが Table2 であるような設問に対し，Table 3 のように回答した場合を想定する．

Table 2　ある例題の正答

	最大 ⟷ 最小			
Q1	4	1	2	3

Table 3　ユーザーの回答の１例

	最大 ⟷ 最小			
Q1	1	4	2	3

このときユーザーは最大値の欄に回答すべきデータ番号（４番）を 2 位の欄に回答しているため最大値の回答に対して 0.25 点が与えられる．さらに最小値のデータ番号は正答しているので，最小値の回答に対し 0.5 点が付与される．したがってこの回答に対する配点は 0.75 点となる．次段以降に各実験の詳細および結果を記す．

5-2　実験１　全体像の把握

この実験における比較指標はデータ全域の平均値である．被験者は各データの可視化結果を閲覧することで，データ全域の平均値を目視で見積もり大小比較を行う．各データを見積もった平均値の大きいものから順に並べ替えてもらい，その正答率（点数）および回答に要した時間を調べる．

実験結果は Table 4 のようになった．

Table 4　実験 1 の結果

実験 1	VR	SC Ⅰ	SC Ⅱ
平均点	0.819	0.857	0.889
平均回答時間（s）	62.6	61.2	48.0

Table4 の結果をプロットしたものが下の Fig. 54 である.

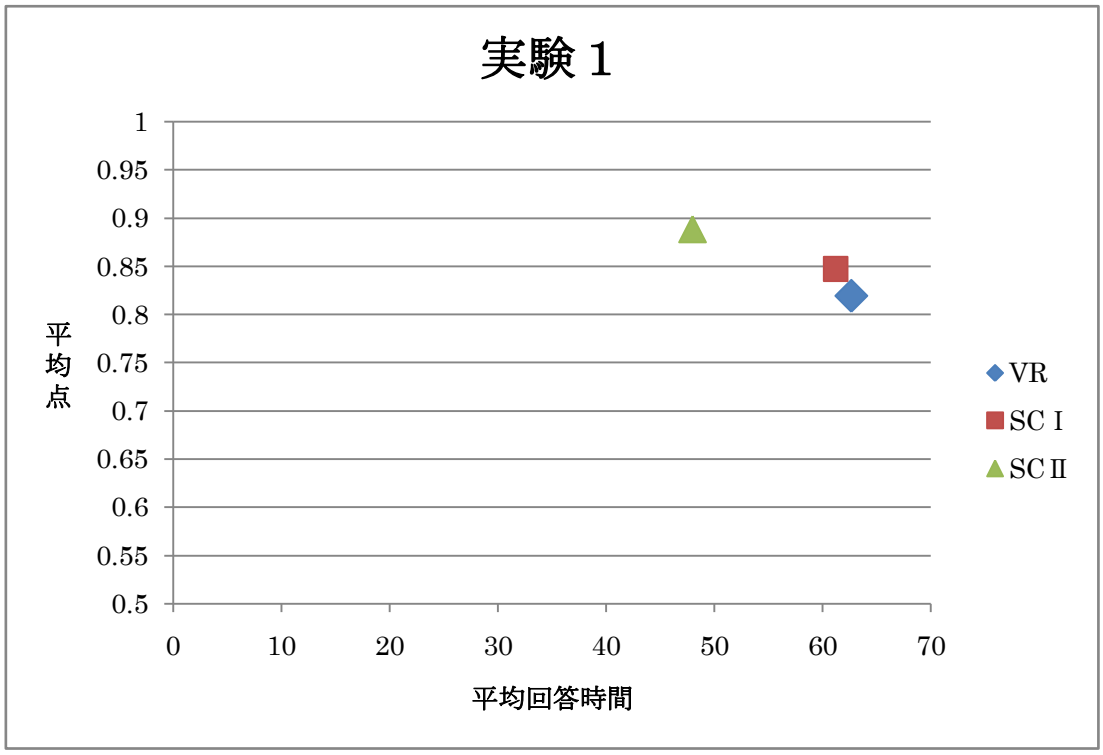

Fig. 54　実験 1 の結果を平均回答時間と平均点を軸としてプロットしたグラフ

平均点・平均回答時間共に SC Ⅱ, SC Ⅰ, VR の順に良い結果となった. SC Ⅱの平均点が最も高い理由はこの手法においては全体の平均値が中心のブランク領域にそのまま可視化されているためである. このため Min Max Bar を操作して適切なパラメータを設定すれば平均値の近いボリュームデータ同士であっても目視で比較することは十分に可能となる. このとき可視化に用いるグラデーションテクスチャの色解像度が十分である必要がある. SC Ⅰ の平均点が VR よりも高い主な理由は視点依存性の無いこと・遮蔽が存在しないことのいずれかもしくは両方であると考えられる. ただしボリュームレンダリングにおいても視点を固定して各データを比較すればスタティックな視点からの比較は実現可能である. ただし視点から反対側に存在するボクセルや内部のボクセルが可視化結果に現れにくいケースには上記の 2 点の理由により静的な比較では不十分となる. その一方で単一の試行を見た場合には SC Ⅰ を用いた場合の得点が VR よりも低い結果となった試行も相当数存在

する．このような回答結果が生じる理由として次のようなものが考えられる．ボリュームデータが比較的分散の大きいデータ，すなわちボリューム内にデータ値の十分高い領域と十分低い領域が双方とも存在する場合を考える．SC I　による可視化では Fig. 55 のような可視化結果となる．

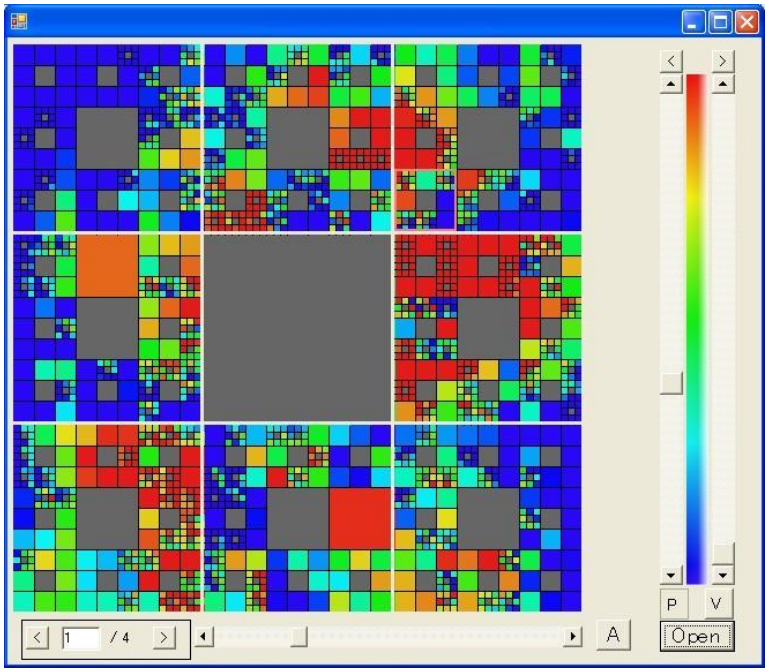

Fig. 55　大きな色コントラストを含む可視化像

このように 2D Window 全域にわたって大きな数値幅をもった分散（正確には色分布）が存在する場合，目視でそれらの平均値を推測する際に値が十分大きい領域（赤色部）と十分小さい領域（青色部）のどちらかの影響で判断にバイアスがかけられ，平均値を過大評価もしくは過小評価してしまう可能性が否定できない．SC を用いた可視化ではこのように極端な特徴値をもつ局所領域の存在が全体像の把握にバイアスをかけてしまうケースが発生しやすいと考えられる．可視化 Window 内に大きな分散をもって散らばった情報を目視により認識・集約し平均値を推測するというタスクの認知コストの高さを物語っていると言える．

一方でボリュームレンダリングを用いた場合にはレイキャスティングによる積分がこの見かけ上の問題を軽減してくれる．ボリュームレンダリングによってあるピクセルに現れる色はそのピクセルを通る視線上のサンプリング点における伝達関数の値を積分したものである．従って視線方向の数値の大きなばらつきは可視化結果に現れず，被験者がその問題を意識することもなくなる．言い換えれば視線方向の積分によってバイアスの引き金になる情報がフィルタリングされていることになる．さらに視点位置・方向を変更することでバイアスの影響を受けにくいと思われる可視化像を得て，それら同士を比較するという

方法をとることが可能である．結果的として認識負荷は低減され，視点操作とパラメータ操作両方を要するというインタラクションコストの高さにも関わらず SC I の場合と大差ない時間で回答が可能となる．

平均点としては最も低いボリュームレンダリングであるが，その点数は 0.819 点となっており平均値を推察・比較する目的においてはボリュームレンダリングの正答率は決して低いとは言えない．本実験のために実装したボリュームレンダラが比較的単純なものであることを考慮すれば，ボリュームレンダラを商用品質程度まで改良すればこの順位が変動することは十分に考えられる．

5-3　実験 2　局所特性の把握

実験 2 ではボリューム内部の一領域（3D Window では 3 次元領域，2D Window では 2 次元領域）が指定されている．この指定領域におけるデータ値の平均値がここでの比較指標である．被験者は各データに対し指定された領域における平均値を目視で見積もり大小比較を行う．指定した領域はボリューム内部に存在する R=2 のボクセルにあたる領域である．被験者には見積もった平均値の大きいものから順に並べ替えてもらい，その正答率および回答に要した時間を調べる．実験 1 と同様に 3 手法による比較実験を行った．

比較対象となる領域は下の Fig. 56 に赤枠で示された部位である．被験者はこの領域内のボクセル平均値をボリュームデータ間で比較し，大小関係を判断する．

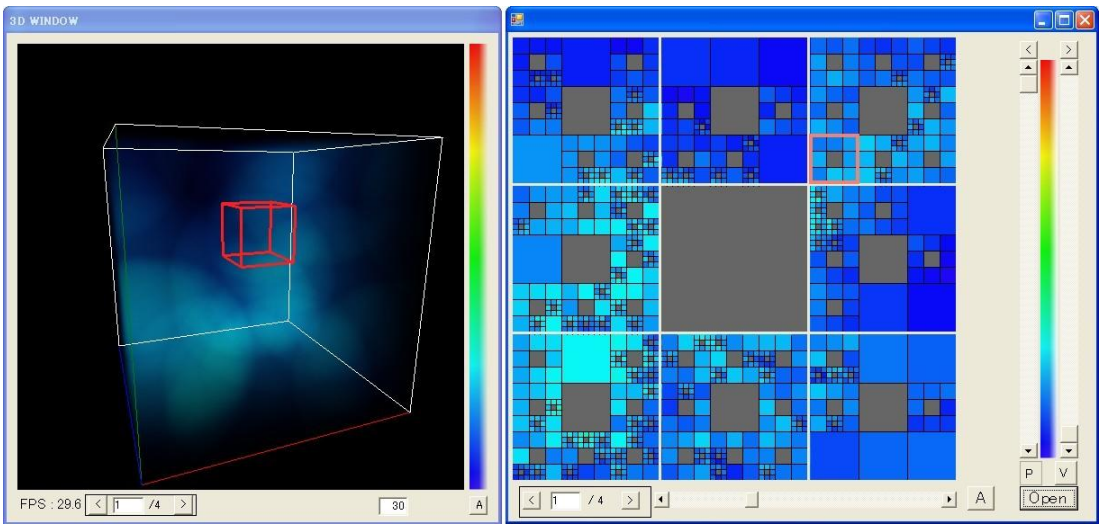

Fig. 56　実験 2 における比較対象領域　左：VR 右：SC I

実験 2 の実験結果は Table 5 のようになった．

Table 5　実験 2 の結果

実験 2	VR	SC Ⅰ	SC Ⅱ
平均点	0.722	0.944	0.972
平均回答時間（s）	97.6	46.0	58.6

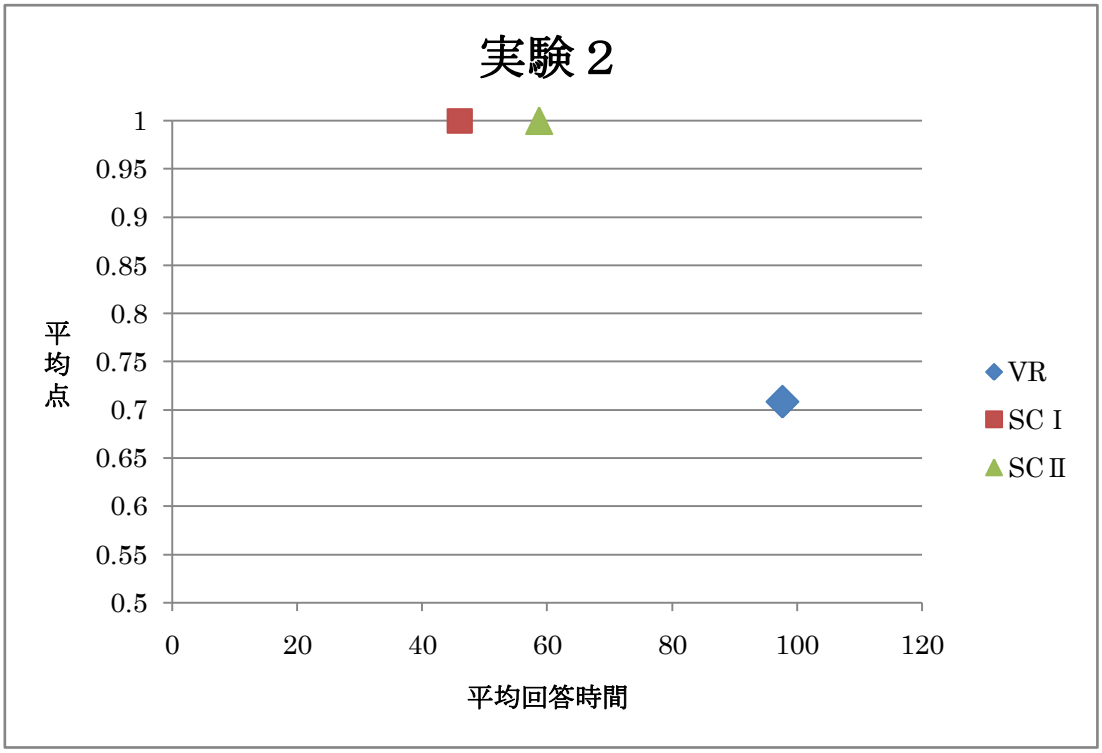

Fig. 57　実験 2 の結果を平均回答時間と平均点を軸としてプロットしたグラフ

この実験では VR の成績が他を大きく下回った．さらに実験 1 における VR を用いた場合の平均点をも下回っている．これには 2 つの原因があると考えられる．ひとつは遮蔽性の問題であり，ボリューム内部の指定領域のボクセル色分布は視点から見て手前側にあるボクセル色に遮蔽されるため容易には把握困難であること．たとえ手前側に遮蔽がない場合でも指定領域よりも奥側のボクセル色がピクセル色として表れている可能性がある．このため必然的に視点操作の手順が必要となる．もうひとつの問題は視点依存性の問題であり，あるデータについて指定領域の値を把握しやすいような視点位置・角度はボリューム内の値分布に依存するためボリュームデータごとに異なる．したがって各データを同一の視点から比較することができず，これが直感的な比較評価を妨げる．この 2 つの問題点が平均点・平均回答時間に影響したと考えられる．SC Ⅰ および SC Ⅱ では遮蔽性・視点依存性が存在しないためデータ間の比較が容易となり高い平均点を短時間で記録することができている．さらにこの局所特性把握では実験 1 の全体平均値把握の点数を上回った．これは平均値を推測する対象領域を一部に指定したことで考慮すべき矩形の数が減少したため認知負

荷の低減につながったためであるといえる．実験１で全域の平均を推測する際にバイアスがかかる可能性を示したが，実験２では考慮すべき領域面積・要素数が限定されたためバイアスの影響をそれだけ受けにくくなりその結果として平均点の向上につながったという推論が可能である．

５-４　実験３　特徴値の把握

実験３では，ボリュームが内包するボクセルに格納されたデータの最大値を見積もる．被験者は可視化パラメータを調節することでデータに含まれるボクセル値の最大値を目視で見積もり大小比較を行う．各データを見積もった最大値の大きいものから順に並べ替えてもらい，その正答率および回答に要した時間を調べる．この実験ではブランクに平均値を可視化する手法を用いるメリットは無いと考え，VR, SC I の２手法のみを試した．

実験３の実験結果は表６のようになった．

Table 6　実験３の結果

実験３	VR	SC I
平均点	0.750	0.944
平均回答時間（s）	80.7	61.8

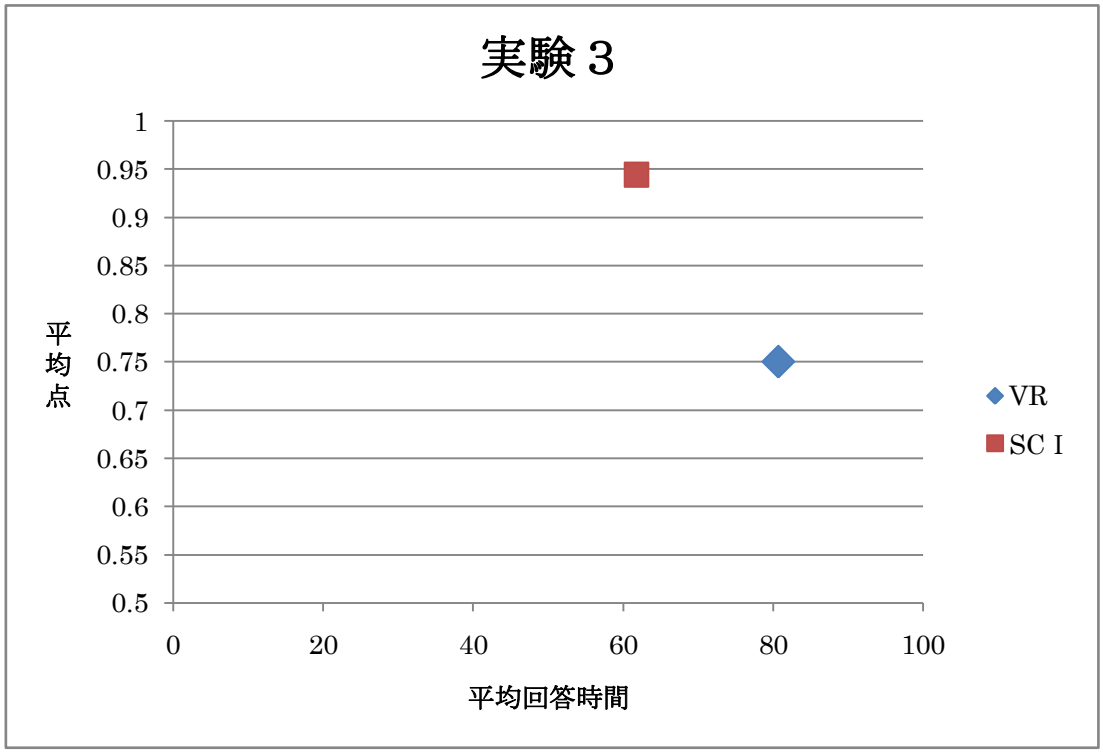

Fig. 58　実験３の結果を平均回答時間と平均点を軸としてプロットしたグラフ

実験 3 では VR と SC I で平均点・平均回答時間に大きな差が出た．この理由を考察するためにタスクにおいて被験者がどのような操作を行ったかを例示する．最大値を抽出したい場合には Min Max Bar を操作し，v_{max}・v_{min} の差を小さくすることによってデータ値の高い部位のみがレンダリング結果に現れるようにできる．この操作を VR で行ったものが次の Fig. 59、SC I で行ったものが Fig. 60 である。

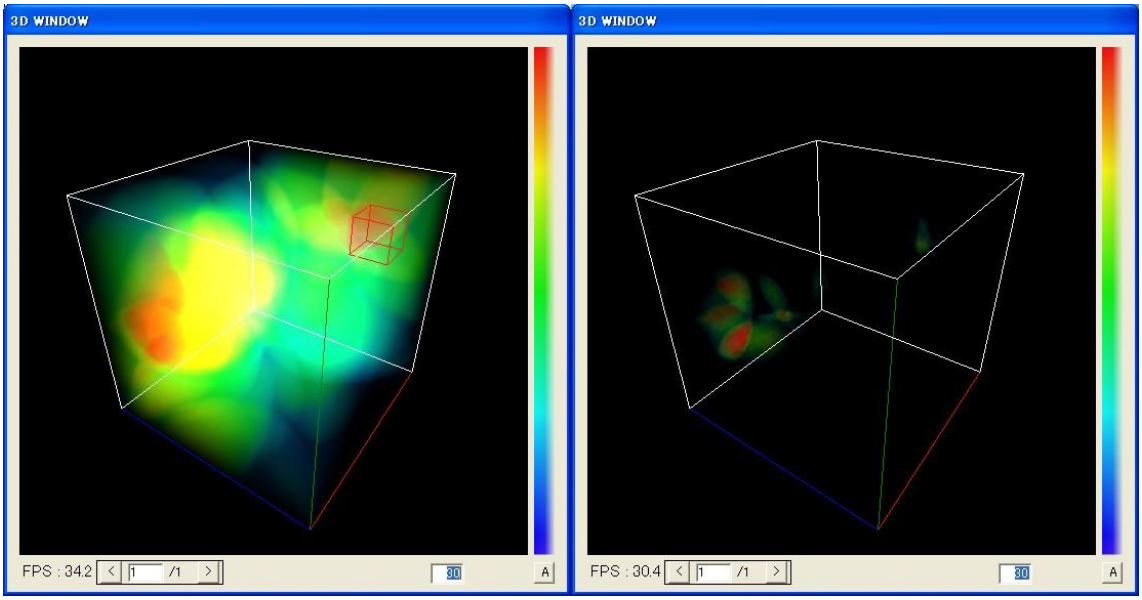

Fig. 59　VR における可視化像の調節

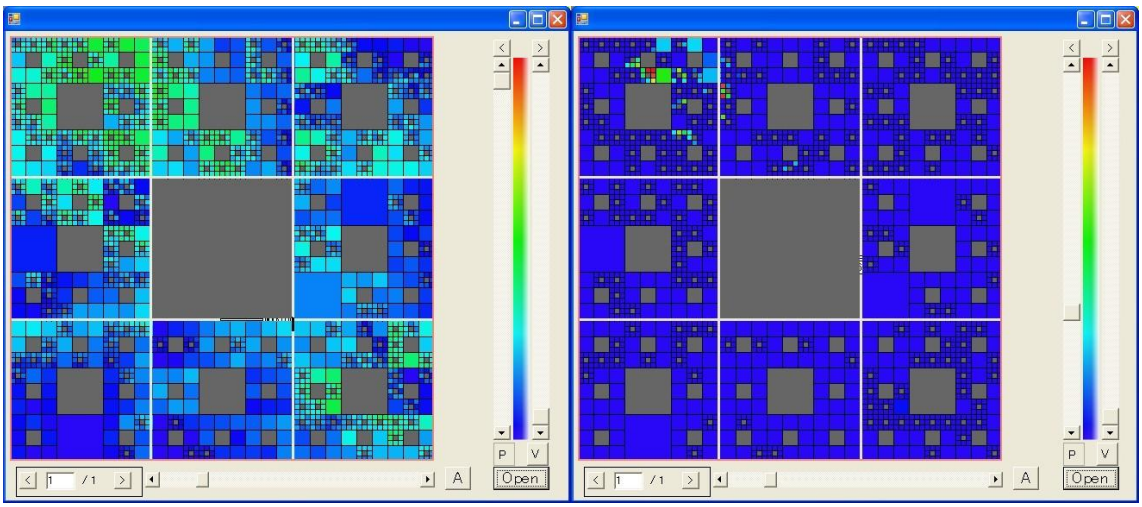

Fig. 60　SC I における可視化像の調節

このようにパラメータを調節した後でボリュームデータ同士を比較する．v_{min} の値がデータ

に含まれるボクセルの最大値を超えれば可視化結果には何も現れなくなる．3D Window であれば Window 全域が黒に，2D Window であれば全域が青になる．このことはその状態では Texture Function に代入する u の値が u<0.0 となることより明らかである．従ってボリュームデータの最大値同士を比較するためには v_{min} をどこまで上昇させれば可視化結果に何も現れなくなるかを比較すればよい．

それでは SC I が VR に比べ平均点および平均回答時間において優れている理由は何か．上の Fig. 59 と Fig. 60 を見比べると SC I の方が値の高い領域（u>0.0 の領域）を明確に示していることがわかる．ボリュームレンダリングでは複数のサンプリング点の寄与を足し合わせることで内部構造の可視化像を描画することを前提にしているため，伝達関数はボクセル値を不透明度（アルファ値）をもつ色に変換する．本研究ではサンプリング点を均等間隔で配置しているため高い値を持つ領域が小さい場合には有効なサンプル点の数が不足し，出力ピクセル色はアルファ値が小さいためくっきりとした可視化像が得られない．正確には有効なサンプリング点の数は領域の大きさではなくその領域の視線（ray）方向の長さ（これを L とする）に比例する．ここで問題になるのがボリュームレンダリングの結果には解釈の一意性が十分確保されない場合があるという問題である．たとえば L が非常に短い u=1.0 の領域と L が比較的長い u=0.90 の領域は結果的に似たようなピクセル色として出力されうる．この問題は u=1.0 のテクスチャ色と u=0.90 のテクスチャ色が類似しており，ただし伝達関数によって付与されるアルファ値が異なる場合に生じやすい．この問題を解決するためには伝達関数におけるボクセル値とアルファ値の対応関係を変更・調節するためのパラメータを追加する必要がある．一方で 2 次元展開を用いた SC I では各ボクセルは各矩形に独立して可視化される．そのためこの手法ではアルファ値を用いる必要性が存在せず，矩形の不透明度は常に 1.0 である．したがってグラデーションテクスチャに十分な色解像度があれば解釈の一意性を期待することができ，結果として本実験のような特徴値の抽出において大いに効果を発揮できたと考えられる．平均回答時間についても同様の理由と視点変更を伴わないというインタラクション・コストの低さが結果に表れている．

5-5　実験に関する考察

本章で行ったユーザーテストは提案手法と 3 次元可視化手法の定性的な相違を明確化することを目的としたものである．実験ごとの結果についての考察は上ですでに述べたとおりであるが以下に簡潔にまとめる．

実験 1 ではデータ全域の平均値の大小を目視で判断するというテストを行った．これは可視化像全体を眺めてデータの特性を判断する際にボリュームレンダリング・2 次元展開両手法がもたらす効果を確認するためのものである．手法 A）と手法 B）で可視化像は明確に異なり，したがってその解釈プロセスも大きく異なるにも関わらず平均点・平均回答時間において大きな差は確認できなかった．手法 C）では求める平均値が中央の矩形に明示的に可視化されているため他の手法に比べ優位な結果につながった．

実験２ではデータのある領域における平均値の大小を目視で判断するというテストを行った．これは遮蔽の有無がボリューム内部のある部位の特性把握にもたらす効果を確認するためのものである．ボリュームレンダリングでは指定した領域がボリューム表面でなく内部にあるとき遮蔽の問題が顕著になる．一方で 2 次元展開ではボリューム内部のボクセルも遮蔽なく表示される．

実験３では各ボリュームデータに含まれる最大値同士を比較した．ここでは 2 次元展開を利用した可視化が優位な傾向を示した．透明度を用いない 2 次元可視化では最大値を格納した矩形の存在が明確に示されるため把握も容易である．

以上の結果から導き出せる 2 次元展開の特徴は以下の通りである．2 次元展開による手法は個々のボクセルが興味の対象となる場合に効果を発揮する傾向がある．単一のボクセルだけで意味を持ちうるケースはそれほど多くないものの，その単一のボクセルが確かに存在するという確証を可視化主体に与えることができるためこれは有意義である．逆に多数の矩形を考慮する場合にはその効果が十分に発揮できない場合がある．このようなケースではブランク領域に平均値を可視化することで認知負荷やバイアスの影響を軽減することが可能である．一方でボリュームレンダリングや流線では複数のボクセルにわたって寄与が合成され，一種の Composition によって可視化主体にとって意味のある結果を描画している．可視化主体はこれらの可視化像を様々な角度から眺め，意識上に構築した 3 次元像に意味を見出すが，それは他の像と容易に比較可能なほど明確なものではなくやや曖昧な解釈を得るに過ぎないようである．とはいえ本研究では単純なボリュームレンダリングによる比較実験のみしか行っていないためこの推察を結論付けるには不十分である．

第６章　改良

5 章の考察を踏まえて可視化システムに改良を施す.

６-１　改良方針

以下の方針に基づいて改良を施す.

1). 同一画面で複数データを比較するための可視化エフェクトの実装
2). 3 次元可視化による可視化像を SC 上に表示するスナップショット機能
3). キャッシュ機能による Octree 構築の高速化

1) は複数データの比較をより効率化するための施策である．5 章にて行った複数データ比較実験では複数のデータを読み込むことはできても一度に表示できるデータはひとつだけでありボタン操作によって可視化するデータを切り替えながら比較する必要があった. 3 章にて開発したアプリケーションは複数プロセスの同時起動が可能であるからアプリケーションを２つ立ち上げて比較することも可能ではあるが同一画面内で比較できればより効率が良いと考えた.
2) は 2 次元展開による可視化のみではデータの 3 次元的な連続性や形状を把握することが困難であるという欠点の克服を試みるものである．3D Window の可視化像をキャプチャーし,「ボリュームデータの地図」である SC 上に貼り付けることでボリュームデータの特徴把握を補助する.
3) は Octree の構築を高速化することで大容量データに対しても快適に可視化を実現することを目的とする.
まず改良後のシステムのスクリーンショットを以下の Fig. 61 に示す.

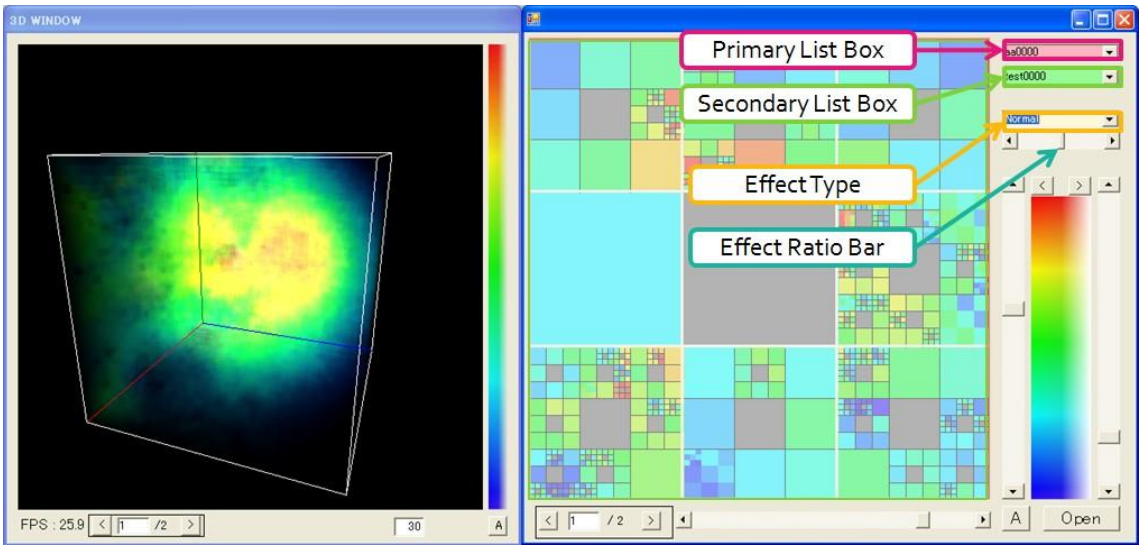

Fig. 61　改良された可視化システムのスクリーンショット

改良前との大きな違いは 2D Window の右上部に 3 つの List Box と 1 つの水平スクロールバーが追加されている点である.

6-2　複数データの可視化

複数データの比較性能をより向上させるために複数のボリュームデータを同時に読み込み，そのうち 2 つを選択して同一ウィンドウ上で比較可視化することを可能にする．また選択した 2 つのボリュームデータに対してある種の処理（エフェクト）を付加することで同座標のボクセルに格納された値の相関関係を可視化する手法を提案する.

まず複数のボリュームデータを読み込んだ際，これらのデータ一覧が右上の Primary List Box および Secondary List Box にて選択可能になる(Fig. 62).

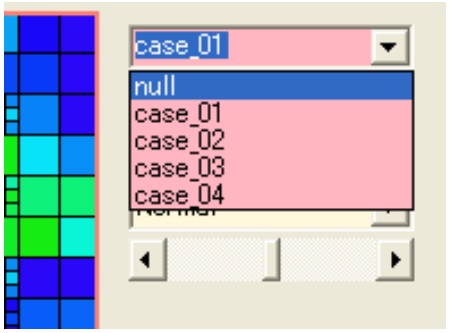

Fig. 62　読み込んだ複数データからの選択（Primary List Box）

このように 2 つの List Box を用いて読み込んだデータのうち 2 つを選択することができ

るがこの２つのデータには主従関係（優先関係）がある．すなわち Primary List Box で選択したデータが主データ（Primary Data）であり Secondary List Box で選択したデータが従データ（Secondary Volume Data）である．

その一段下にある List Box は Effect Type List Box と呼ばれ，用いる Effect の種類を選択する．ここでいう Effect とは Primary Data と Secondary Data の関係性を定義するものであり，プリセットとして Normal, Multiply, Add, Subtract の 4 種の Effect が用意されている(Fig. 63)．

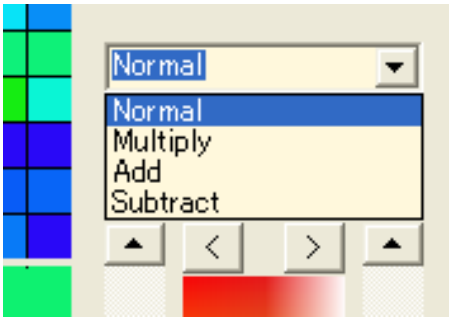

Fig. 63　Effect Type の選択（Effect Type List Box）

4 種のエフェクトの効果はそれぞれ以下の通りである．

1). Normal　―　２つのデータの値を混合（Mix）して可視化する

2). Multiply　―　２つのデータの値を乗算して可視化する

3). Add　―　２つのデータの値を加算して可視化する

4). Subtract　―　Primary Data の値から Secondary Data の値を減算した結果を可視化

２つのデータを対等に扱わず一方を主，他方を従とする理由がこのエフェクトである．例えば Subtract ではどちらからどちらの値を引くかが重要であるし，混合においてもその比率はどちらを基準とするかを設定しなければならない．

Effect Type List Box の下にある水平スクロールバーは Effect Ratio Bar と呼ばれ，Effect の強弱を調節するバーである．

改良後のシステムにおける可視化の手順を説明する．ボクセル分割は改良前と同様であるのでここでは触れない．ただし Primary Data と Secondary Data に対し計２つの Octree を保持する必要がある．

改良前はボクセル値を即座に色に変換して最終結果としていたが改良後はエフェクトをかける手順が加わったためまず Primary Data，Secondary Data のデータ値を浮動小数点バッファにレンダリングする．浮動小数点バッファは色ではなく浮動小数（float）値をピクセルに格納できるバッファである．レンダリングされるのは可視化結果に表れる領域のみであり，これは改良前のシステムにおいて RGB 色で行っていたレンダリング処理をデータ値（浮動小数値）を直に書き出す処理に変更したにすぎない．すなわちここで必要とな

るのは 2D Window の描画エリアと同じサイズの浮動小数点バッファ 2 枚（Primary Data 用と Secondary Data 用）である．それぞれ Primary Buffer，Secondary Buffer と呼ぶことにする．これらのバッファをテクスチャとみなしてテクスチャサンプリングを行うことで各ピクセルに格納されたデータ値を自由に取得することができる．

最終工程は次のようになる．2 枚の浮動小数点テクスチャをサンプリングしてあるピクセルにおけるデータ値を取得する．Primary Buffer からサンプリングして取得したデータ値をv_p，Secondary Buffer から取得したデータ値をv_sとする．２つの値にエフェクトを施して算出した値v_eを Colorization Function によって色に変換し，最終出力とする．ここまでの工程を図示したものが以下の Fig. 64 である．

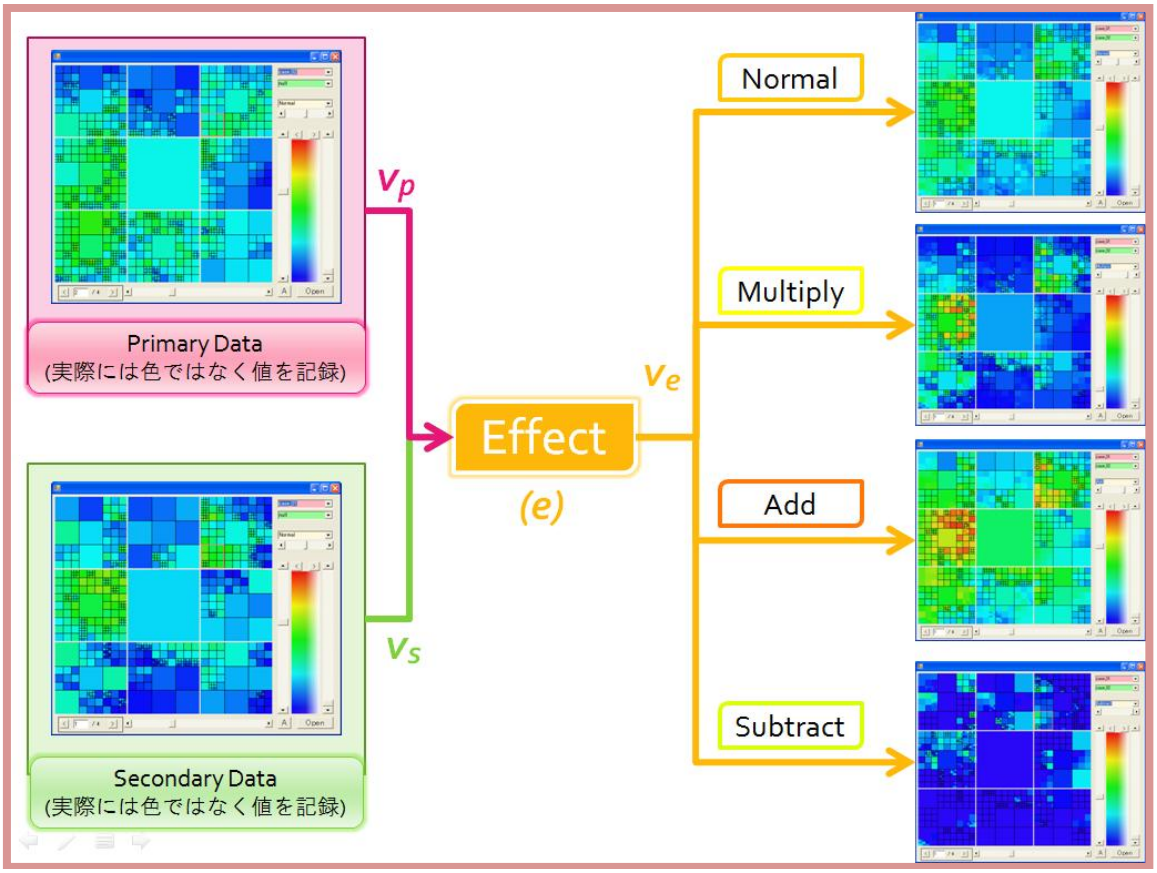

Fig. 64　改良後のシステムにおける可視化工程

Effect によってv_p・v_sの値はv_eに変換される．この変換式は Effect Type により異なり，また Effect Ratio Bar により設定される値eによってこの Effect の影響度は調節される．Eの値は 0.0 ≦ e ≦ 1.0 の範囲である．具体的には以下の変換式に従う．

1)．Normal

$$v_e = (1.0 - e)v_p + ev_s \qquad (10)$$

2). Multiply ― ２つのデータの値を乗算して可視化する

$$v_e = (1.0 - e)v_p + ev_p v_s \quad (11)$$

3). Add ― ２つのデータの値を加算して可視化する

$$v_e = v_p + ev_s \quad (12)$$

4). Subtract ― Primary Data の値から Secondary Data の値を減算した結果を可視化

$$v_e = v_p - ev_s \quad (13)$$

いずれの Effect Type においても$e = 0.0$を設定した場合には$v_e = v_p$となるように設計されている．こうして算出されたv_eを Colorization Function に代入し，１次元テクスチャによって色へ変換することで可視化が完了する．

例として Multiply Effect の有用性を考えてみる(Fig. 65)．２つのデータにおいて共通して高い値をもつ部位のみを抽出したい場合にこの Effect は有効である．ボクセル値を乗算することで Primary Data・Secondary Data の一方もしくは両方の値が低いボクセルは可視化結果に現れにくくなる．そして 2 次元展開による可視化はこれらの部位を遮蔽なしに可視化できるという利点を備える．

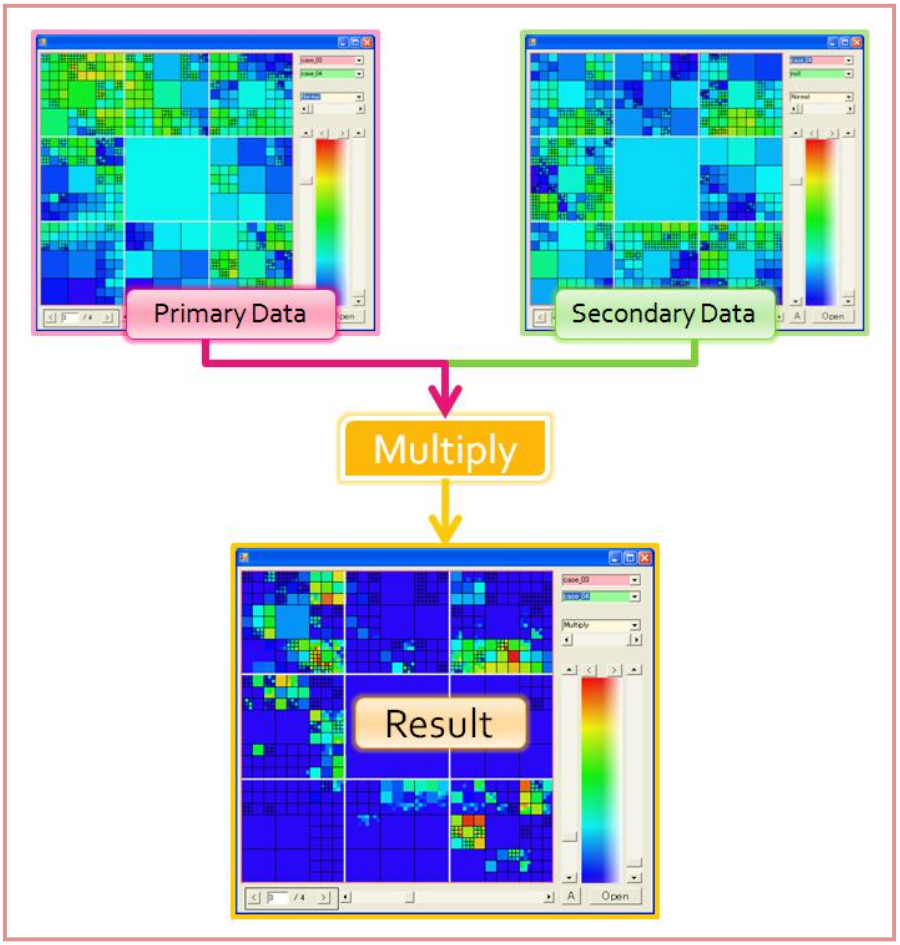

Fig. 65　Multiply Effect を利用した重複領域抽出

6-3　スナップショット機能

前章までの議論より 2 次元展開では 3 次元空間の連続性が保たれないため形状把握などの空間的なデータ値の連続性が関わる特徴を把握することが極めて困難であることが明らかである．これは 2 次元展開の最大のデメリットであるといえよう．本研究では 3 次元可視化手法と相互運用することでこのデメリットを軽減してきたが，そのために 2 つの可視化 Window を交互に眺めることが必要とされた．

ここで 2 次元展開図がボリュームデータの地図であるという観点に立ち返り，地図である SC 上の矩形―すなわちある「地点」を指定したときにその周辺の様子―3 次元可視化像を地図に重ねて表示することで SC 上のみでボリュームデータの 3 次元特性をある程度把握可能となるような改良を行う．これは 3D Window の可視化像をスナップショットとして SC 上に貼り付ける機能であるのでスナップショット機能と呼ぶことにする．

先行例としては Google 社の提供する地図サービス Google Map および Google Street View があり(Fig. 66)，本機能はこのアイディアをボリュームデータ可視化に取り入れるも

のである[31][37].

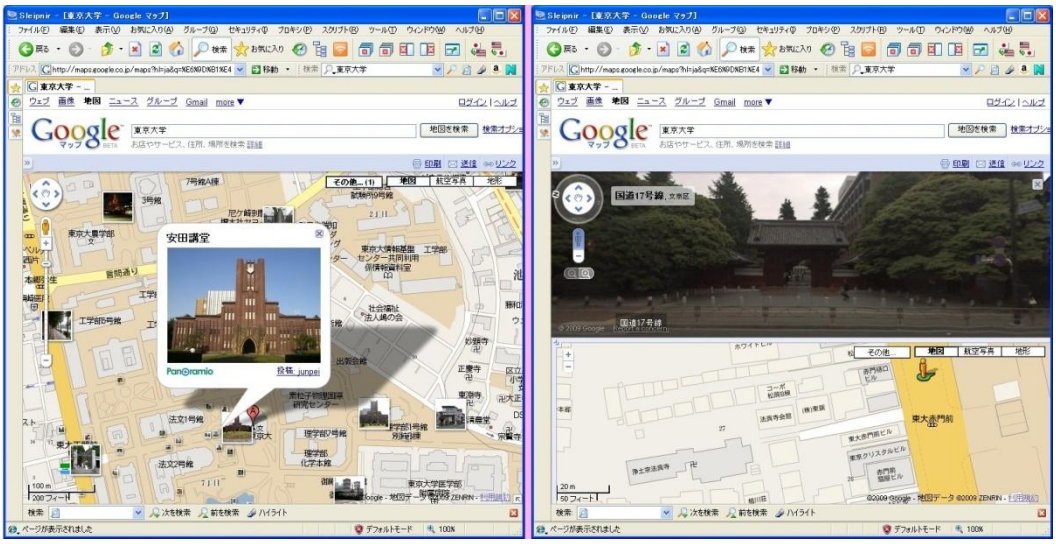

Fig. 66　Google Map（左）と Google Street View（右）　[31][37]

Google Map では地図上に配置されたアイコン（バルーンフラッグ）をクリックすることでその地点や周囲に関する情報およびユーザーの投稿した写真が地図上にポップアップ表示されるようになっている．Google Map に付加された機能である Google Street View では地図上の道路へズームすることで自動的にズーム地点の周囲を撮影した画像が表示される．この画像は 360°全周囲画像でありマウス操作により視点方向を変化させ全周囲を眺めることが可能である．このようにスナップショットを地図に重ねて表示することで地図情報のみからは得ることのできない現実 3 次元空間の様子をユーザーが理解することが可能である．Google Map では画像の存在を地図上にアイコンを配置することでユーザーに通知している．これによってユーザーはその画像が地図上のある一地点と厳密に関連付けられていることを直感的に理解できる．

この Google Map 等に見られるアイディアを我々の提案手法に取り入れ改良したものが次の Fig. 67 である．

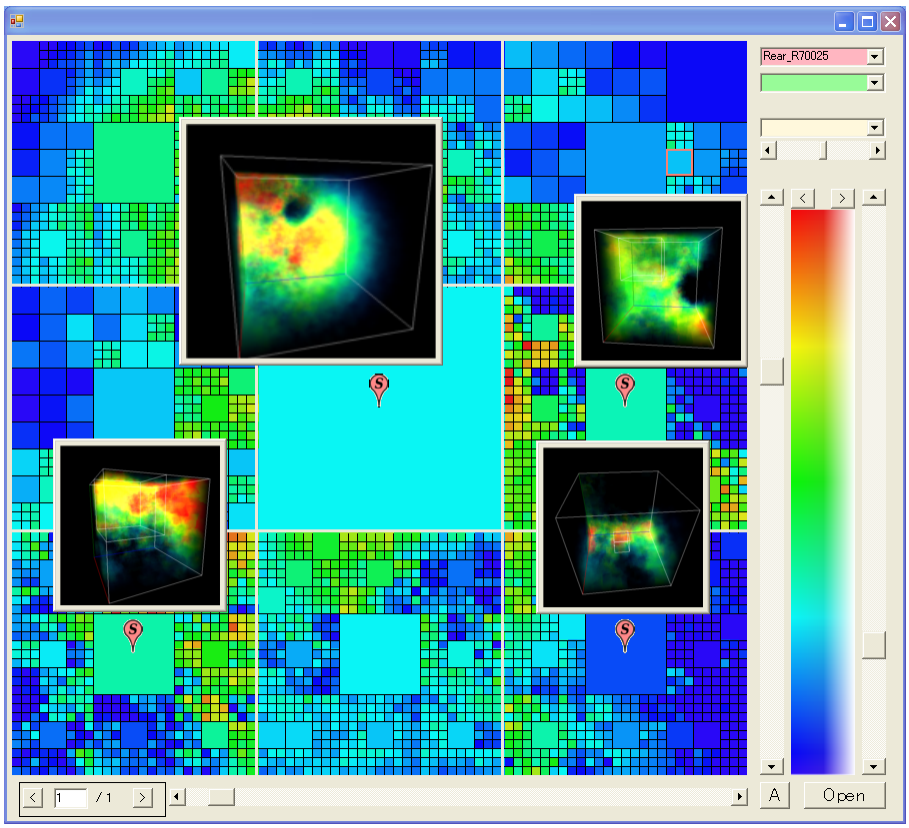

Fig. 67　スナップショット機能を実装したシステム

スナップショット機能について説明する．まず通常の可視化ワークフローの中でボリュームデータの特徴を捉えた 3 次元可視化像を得る．この 3D Window での可視化像をスナップショットとして 2D Window に転送するためには，3D Window 上で右クリックした際に現れるメニューから"Snapshot the View"を選択する．こうすることで 2D Window 上にフラッグが追加される．このフラッグはスナップショットがその位置において撮影されたことを示すマーカーである．2D Window においてフラッグが立てられる場所はそのときに注視していたボクセルに相当する矩形の位置である．

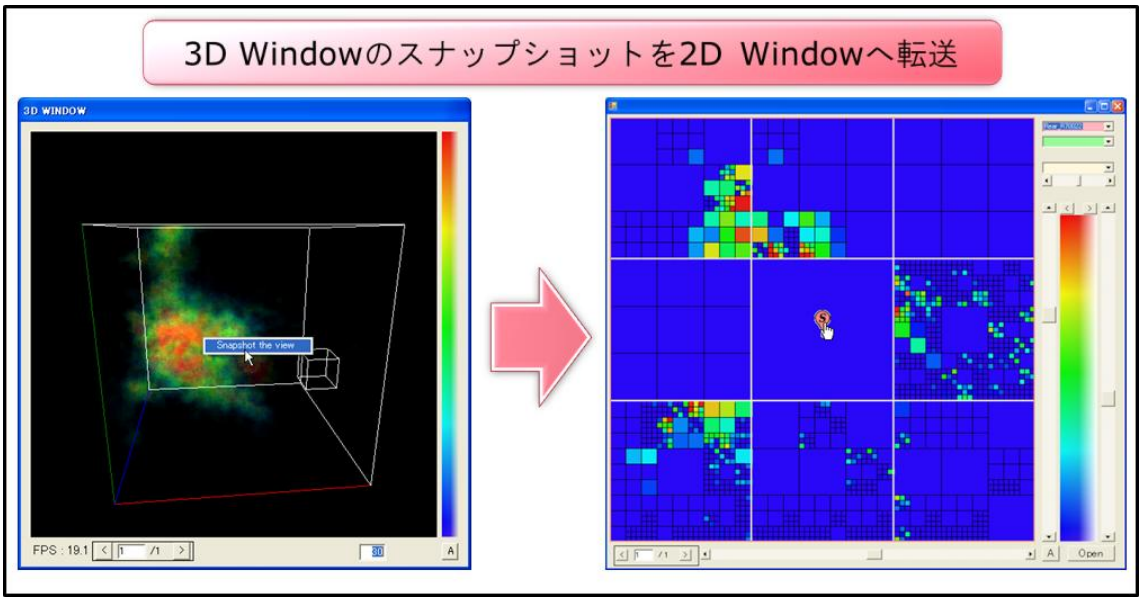

Fig. 68　スナップショットを撮影した地点にフラッグが追加される

このフラッグをマウスでクリックすることで保存したスナップショットが 2D Window 上にてポップアップ表示される.

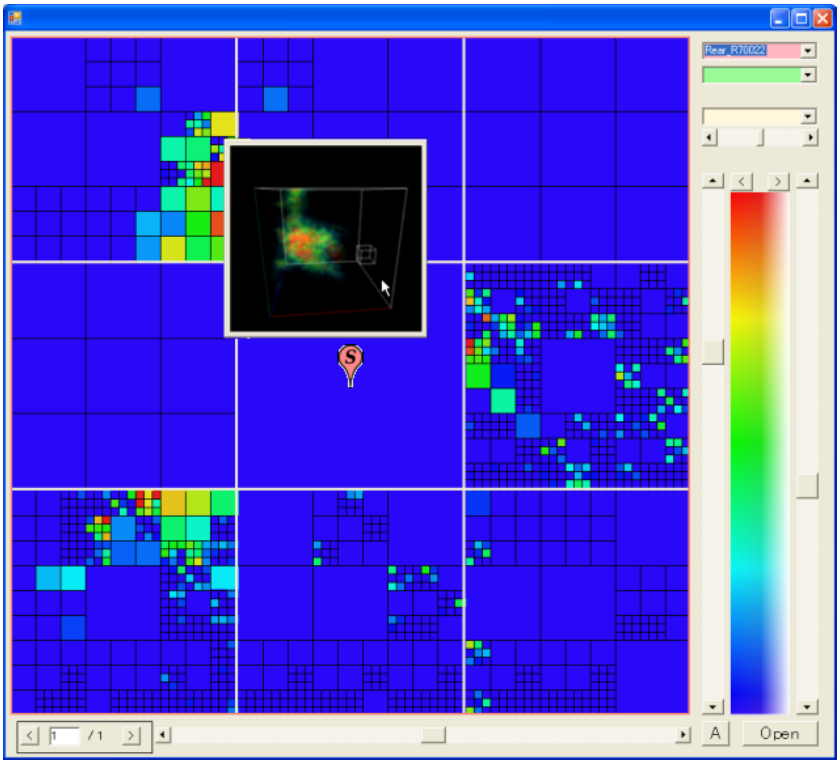

Fig. 69　フラッグをクリックすることでスナップショットが表示される

データの特徴を捉えたレンダリング像や流れ場の特徴を捉えた流線可視化像を数枚保存しておくことで 3D Window を参照することなくボリュームデータの特徴がある程度認識できるようになる．以下の Fig. 70 はベクターデータの可視化にスナップショット機能を活用した例である．

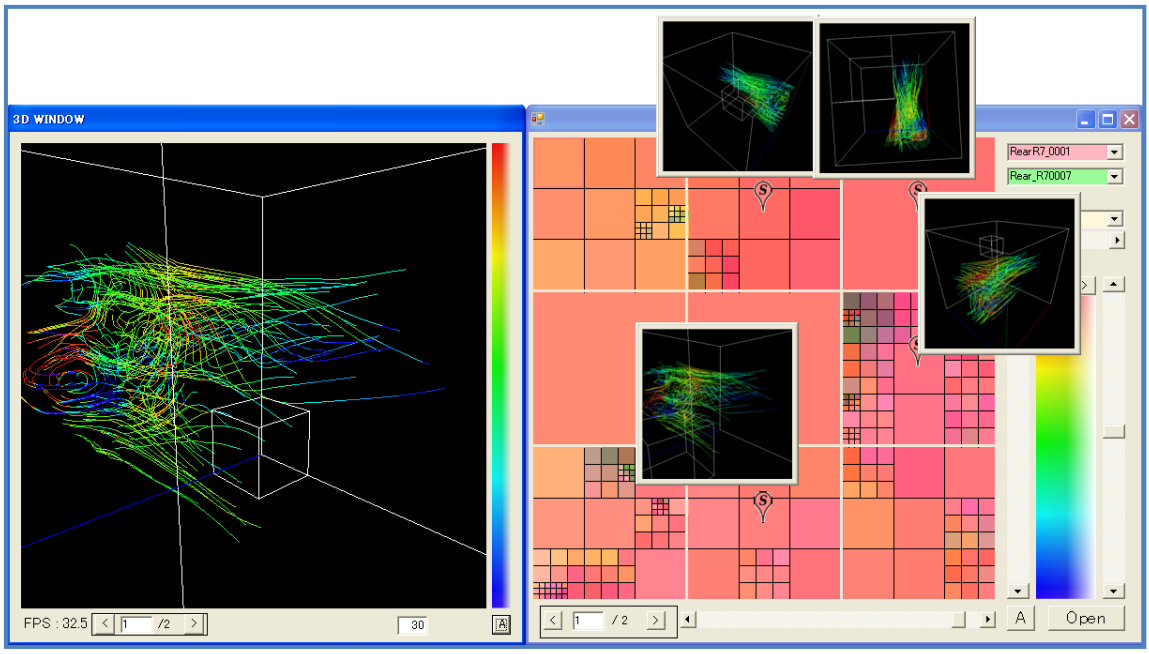

Fig. 70　スナップショット機能を活用したベクターデータ可視化

6-4　キャッシュ機能によるOctree構築の高速化

2 次元展開による可視化では事前処理における Octree 構築が可視化パフォーマンスを低下させる要因であった．この Octree 構築処理は頻繁に行う計算ではないものの多数のボクセルの平均値および分散値を多数回算出する必要があるためボリュームデータの解像度が上昇するにつれて可視化パフォーマンスが低下する．そこで上位階層（低解像度）のボクセル平均値および分散値をメモリ上にキャッシュしておくことでこの Octree 構築を高速化するという改良を行った．

低解像度のボクセルになるほど内部に多数の R_{max} ボクセルを内包するため平均値および分散値の算出に時間がかかる．例えば R=5 のボリュームデータについて Octree を構築するとき R=0 ボクセルは 8^5=32768 個の R=5 ボクセルを内包しており，その平均値の算出は 32768 個のボクセルの格納値を足し合わせ 32768 で割るという処理になる．分散値の算出はさらに計算量の多い処理である．しかしながら R=0 のボクセルは 1 個しか存在しないためその平均値および分散値をメモリ上に保持したところでメモリ占有量の増加は微々たるものである．同様に R=2 のボクセルは高々8 個しか存在しないためこれらの平均値・分散値もメ

モリ上に保持しておくことができ，Octree 構築の際の再計算を避けることで計算時間の短縮につながる．これを図示したのが以下の Fig. 71 である．

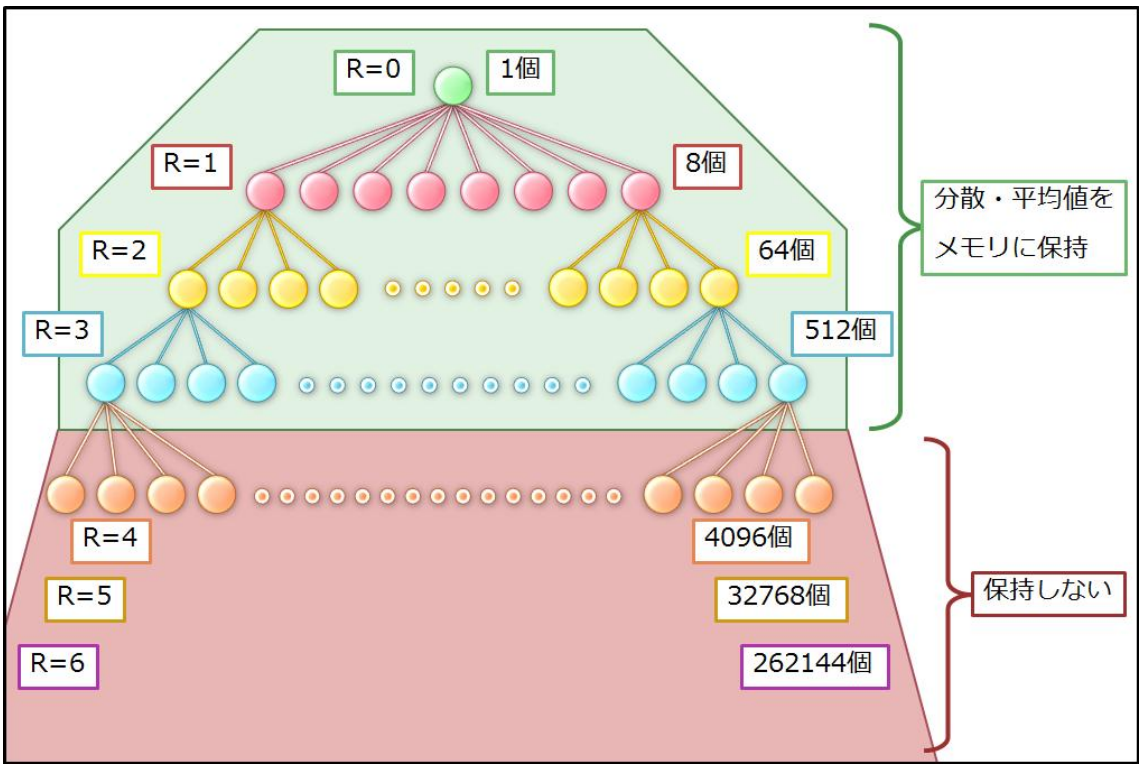

Fig. 71　分散値・平均値のキャッシュ（R=3 まで）

メモリ占有量と計算時間のバランスを考え，改良後のシステムでは R=3 までのボクセル平均値・分散値を保持するようにした．また Window サイズの制約から矩形として可視化されるボクセルは高々R_{draw} 階層分にすぎないことに着目する．必ずしもボリュームデータ全体の Octree を構築する必要はなく，2D Window に表示される枝領域の分だけ構築すれば可視化上問題ないため，計算対象を可視化結果に現れるボクセルに限定した．ただしこの場合ズームイン・ズームアウト機能により可視化対象領域が変更された際にも Octree 再構築が必要となる．これらの改良により R=7～8 程度のボリュームデータをストレスなく閲覧・可視化することが可能となった．

ここまでに行った改良は前章までの考察および評価実験に基づくものである．これらは本手法には依然として様々な発展・改良の余地が残されていることを示している．今回は残念ながらこの改良版の評価までは行うことは叶わなかったが，こうした考察・評価に基づいた改良を重ねることによってより洗練された可視化手法を構築可能であろう．

第７章　結論

本研究はフラクタル図形を用いたOctreeの２次元展開という新しいコンセプトを提示し，それをボリュームデータに適用することで今までにない種々の特長をもった新しい可視化手法が構築可能であることを示したものである．提案手法は 1 枚の 2 次元平面上にてボリュームデータ全域を遮蔽・視点依存性なく閲覧することを可能にし，尚且つマルチレゾリューションな可視化像を提供する．そして本手法を既存の 3 次元可視化手法と連携させることでボリュームデータの多角的な可視化ワークフローが構築可能であることが示された．

本研究ではまず従来の可視化における手法およびその発展のオーバービューを通じて多角的可視化の必要性，科学的可視化・情報可視化両手法の融合，２次元可視化・３次元可視化の融合といった新しい可視化の試みが求められていることを確認した．

こうした試みに向けた第一歩としてフラクタル図形を利用した２次元展開という情報可視化の考え方に基づく手法開発し，従来科学的可視化の可視化対象であったボリュームデータに適用するというアプローチを採用した．この手法を開発するにあたり既存の可視化手法を研究しそれらの共通点や相違を考察することで有益な着想を多数得た．その結果提案手法はこれまでの可視化手法の特徴を取り入れつつもフラクタル図形の自己相似性を可視化そのものに利用した点，３次元ボリューム空間にダイレクトなアクセスを提供した点などにおいて新規性のあるものになったと自負する．尚，今回提案した手法の核となる「SC によるボリュームデータの 2 次元展開」および「SC の矩形選択による 3 次元座標・領域指定」の 2 点に関しては特許を申請中である（2009 年 1 月現在）[3][4].

印象的であったのはシェルピンスキーのカーペットというキーワードの下に，階層データ, マルチレゾリューション, Level of Detail, Octree, ボリュームデータ, Focus+Context, 2D+3D 可視化，科学的可視化と情報可視化の融合といった可視化における種々の要素が一同に会し，互いに結びつけられることでひとつの可視化手法・システムを形成していったことである．このキーワードの発見・設定が本研究の肝であったと言える．

本研究の次なるステップはより実用に即した機能を提供することであろう．実際の可視化の現場において本手法の有用性が最大に発揮される場面はどのような面かを探ることがまずは重要である．そのためにより詳細な研究・試行を重ねる必要がある．本手法はまだ提案されたばかりの未熟なものであり実用面からいえば心許ない部分も多い．他の可視化手法がそうであったように本手法も今後時間をかけて修正・洗練を重ねることでより実用に即した形態へと進化することが求められる．

別の目標としてボリュームデータ以外のデータへの適用もまた考慮されるべきである．本手法は Octree 構造をもつデータ構造であればどのような対象にも適用可能である．空間データや科学的データに限定せずとも情報可視化分野において本手法を適用することも可能である．

近年の可視化トレンドのうち本研究には盛り込むことのできなかったものもある．特にインテリジェントな可視化の重要性は今後ますます高まっていくことが期待される．可視化結果に人間が意味を見出す端緒，もしくは誤った理解に陥る兆候をコンピュータが察知しサポート・補佐することで，より知的高次元での可視化が実現されるようになるであろう．既に可視化分野においてはビジュアル・データマイニングと呼ばれる可視化とデータマイニングを効率よく組み合わせた融合アプローチが盛んに研究されている．こうした知的可視化アプローチを本手法のみならず様々な場面に取り入れることが可視化研究全般における次なるステップであるという実感を抱いた．

参考文献

[1] 岩丸雅紀, 岡本孝司, “フラクタル図形を用いたボリュームデータの２次元展開”, 第 36 回可視化情報シンポジウム講演論文集, pp. 75-78, July, 2008

[2] 岩丸雅紀, 岡本孝司, “フラクタル図形を利用した２次元展開によるボリュームデータの可視化手法とその応用”, 可視化情報学会全国講演会講演論文集, pp. 167-168, October, 2008

[3] 岩丸雅紀, 岡本孝司, "ボリュームデータ可視化装置および方法並びにプログラム," 特願 2008--142786.

[4] 岩丸雅紀, 岡本孝司, "シェルピンスキーのカーペット上の単一の矩形もしくはそれの内包する点の座標を指定することにより 3 次元ボリューム内の対応付けられた座標または領域を指定する手法およびその装置," 特願 2008--231398.

[5] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *ACM SIGGRAPH Computer Graphics*, Vol.22 Issue 4, pp. 65-74, 1988

[6] J. Kr¨uger and R. Westermann, "Acceleration Techniques for GPU-based Volume Rendering," *Proceedings of the 14th IEEE Visualization 2003*, pp. 38, 2003

[7] A. Kaufman and K. Mueller, "Overview of Volume Rendering," *The Visualization Handbook*, Charles D. Hansen and Chris R. Johnson, Academic Press, pp. 127-174, 2004 (Book style with paper title and editor)

[8] L. Li and Han-Wei Shen, "Image-Based Streamline Generation and Rendering," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, No. 3, pp. 630-640, May/June 2007

[9] “Comparing 2D Vector Field Visualization Methods A User Study,” IEEE Trans. Visualization and Computer Graphics, Vol. 11, Issue 1, pp. 59-70 , Jan./Feb., 2005

[10] Daniel Acevedo, Cullen D. Jackson, Fritz Drury, and David H. Laidlaw , “Using Visual Design Experts in Critique-Based Evaluation of 2D Vector Visualization Methods,” IEEE Trans. Visualization and Computer Graphics, Vol. 14, Issue 4, pp. 877-884 , July, 2008

[11] Nelson L. Max, “Efficient light propagation for multiple anisotropic volume scattering,” *Proceedings of the 5th Eurographics Workshop on Rendering*, pp. 87-104, 1994

[12] Jos Stam, “Multiple scattering as a diffusion process,” *Proceedings of the 6th Eurographics Workshop on Rendering*, pp. 41-50, 1995

[13] William E. Lorensen and Harvey E. Cline, “Marching Cubes: A high resolution

3D surface construction algorithm," *Computer Graphics*, Vol. 21, Nr. 4, July 1987

[14] H. Koike, "Fractal Views: A Fractal-Based Method for Controlling Information Display," *ACM Transaction on Information Systems*, Vol. 13, No. 3, pp.305-323, July, 1995

[15] T. Itoh, Member, IEEE Computer Society, Y. Yamaguchi, Y. Ikehata, and Y. Kajinaga, "Hierarchical Data Visualization Using a Fast Rectangle-Packing Algorithm," *IEEE Trans. Visualization and Computer Graphics*, vol. 10, No. 3, pp. 302-313, May/June 2004

[16] B. Johnson, B. Shneiderman, "Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Space," *IEEE Visualization '91*, pp. 275-282, 1991.

[17] G.G. Robertson, Jock D. Mackinlay, and Stuart K. Card, "CONE TREES: ANIMATED 3D VISUALIZATIONS OF HIERARCHICAL INFORMATION," *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 189-194, 1991

[18] Takayuki Itoh, Member, IEEE Computer Society, Yumi Yamaguchi, Yuko Ikehata, and Yasumasa Kajinaga, "Hierarchical Data Visualization Using a Fast Rectangle-Packing Algorithm," *IEEE Trans. Visualization and Computer Graphics*, vol. 10, NO. 3, pp. 302-313, May/June 2004

[19] G. W. Furnas "Generalized fisheye views," *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'86)*, pp. 16-23, 1986

[20] P. Pirolli, S.K. Card, and M.M. Van Der Wege, "The effects of information scent on visual search in the hyperbolic tree browser, "*ACM Transactions on Computer-Human Interaction*, Vol. 10, issue 1, pp. 20-53, March, 2003

[21] P. Pirolli, S.K. Card, and M.M. Van Der Wege, "Visual information foraging in a focus + context visualization, "*Proc. the SIGCHI*, pp. 506-513, 2003

[22] J. D. Mackinlay, G. G. Robertson, and S. K. Card, "The perspective wall: detail and context smoothly integrated," *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp. 173--179, 1991.

[23] M. Tory, T. Möller, M. S. Atkins, A.E. Kirkpatrick, "Combining 2D and 3D Views for Orientation and Relative Position Tasks," *Proceedings of the SIGCHI*, pp. 73-80, 2004

[24] H. Piringer, R. Kosara, and H. Hauser, "Interactive Focus + Context Visualization with Linked 2D 3D Scatterplots," *Proc. the 2nd International Conference on Coordinated & Multiple Views in Exploratory Visualization*, pp. 49-60

[25] A. Teyseyreand and M. Camp, "An Overview of 3D Software Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, No. 1, pp. 87-105, Jan/Feb 2009

[26] Harvey S. Smallman, Mark St. John, Heather M. Oonk, and Michael B. Cowen, "Information Availability in 2D and 3D Displays," IEEE Computer Graphics and Applications, vol. 21, no. 5, pp. 51-57, Sep/Oct, 2001

[27] John Lamping , Ramana Rao, "Laying out and visualizing large trees using a hyperbolic space", *Proceedings of the 7th annual ACM symposium on User interface software and technology*, p.13-14, November 02-04, 1994, Marina del Rey, California, United States

[28] Jun Rekimoto and Mark Green, "The Information Cube: Using Transparency in 3D Information Visualization," *Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93)*, pp. 125-132, 1993.

[29] George Robertson, Jock D. Mackinlay, Stuart Card, "The Perspective Wall: Detail And Context Smoothly Integrated," *Proceedings of CHI '91 Conference*, pp. 173-179, April 28 - June 5, 1991

[30] Heidi Lam, "A Framework of Interaction Costs in Information Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, No. 6, pp. 1149-1156, Jan/Feb 2009

[31] Google Map, http://maps.google.co.jp/

[32] aravind.ca, http://www.aravind.ca

[33] UCSD Computer Graphics Lab, http://graphics.ucsd.edu

[34] Medical Imagekkk, http://www.biotronics3dj.com

[35] ヒューリンクス株式会社, http://www.hulinks.co.jp

[36] Visualization World, http://www.viz-journal.kgt.co.jp

謝辞

本研究は，指導教員であられる岡本孝司教授，染矢聡准教授を始め，多くの方々の助言と支えによって取り組むことができました．この場を借りて，御礼申し上げます．

岡本先生には，情報系の学生が僕一人であったこともあり入学当初から大変目をかけていただきました．修士 1 年の 5 月から早速学会に参加させていただき，その後も国際会議を始め様々な議論の場に参加する機会を与えてくださいました．学生を encourage することにかけては岡本先生とその研究室は他のどこにも負けないものがあると大いに感じている次第です．もちろん研究では，理論から実践哲学にいたるまで実に多くのアドバイスを頂きました．今回修士論文としてまとめた研究テーマを修士 1 年の 11 月に提案した際にもより実践的・効率的な可視化のために何が必要であるかを研究室会の度に指摘してくださいました．今回の修士論文はこの 2 年間で先生から頂いたアドバイスや交わした議論が研究として結実したものです．本当にありがとうございました．

染矢先生には，研究のことはもちろん，日常の学生生活から各所の学会参加に至るまで，この 2 年間非常にお世話になりました．多くの学生を抱えて休む暇もないほどお忙しい中でも僕たち一人ひとりに声をかけてコミュニケーションをとってくださったこと，分からないことがあったときにはいつでも助けてくださったことを覚えています．本当に嬉しかったです．染谷先生を見習うことで研究が辛かった時期も乗り越えることができました．ありがとうございました．

岡本研究室の先輩の篠原さん，ネジェットさん，飯田さん，桑原さん，大原さん，浅野さん，王子さん，藤井さんには，2 年前，新しくメンバーになった私を温かく迎え入れて頂き，研究室にいるときや研究室旅行など，大変楽しい時間を過ごすことが出来ました．特に王子さん，藤井さん，浅野さんはケアンズの学会で右も左も分からない僕を連れて現地をナビゲートしてくださいました．楽しい思い出をありがとうございます．

この 2 年間，共に修士論文を闘った同期の吉田くんとは，お互いに刺激を与えながら，励ましあい，切磋琢磨することが出来ました．また，研究に関してだけでなく，私生活でも，とても楽しい時間を過ごすことが出来ました． 2 年の間で築くことの出来たこの友人関係は私にとって大きな財産です．2 年間，本当にありがとう．

そして，研究室の後輩である M1 の飯塚くん，緒方さん，内田くん，福田くん，平川くん，熊谷くん，Yanrong さん，Uddin さん，Saha さん，倉くん，秘書の下川さん，門脇さん，池田さん，この 2 年間，色々な面で支えていただき，本当にありがとうございました．見る見るうちに力を付けてくる後輩たちは頼もしくもあり，先輩としては負けていられないと研究する上での励みにもなりました．

学外の方々では可視化用の CFD データを提供してくださった広島大学の中島卓司先

生，M1 の頃から学会等様々な場で助言をくださったお茶の水女子大学の伊藤貴之准教授にも感謝申し上げます. 幸運にも在学中に申請することのできた 2 件の特許案件につきましては東京大学 TLO（casti）の成田真一様，アイテック国際特許事務所の國分敦先生に多大なご尽力をいただきました．本当にありがとうございました.

最後に，ここまで私を支えてくれた両親と友人たちに感謝の意を示し，簡単ではありますが，謝辞とさせていただきます．皆様，この 2 年間，本当にありがとうございました．これからも宜しくお願い致します.

平成 21 年 1 月 28 日

岩丸 雅紀