3D テクスチャ表示技術を用いたボリュームレンダリングシステムの開発とその応用

土井章男[*1]，高橋弘毅[*1]，馬渡太郎[*2]，女鹿幸夫[*3]

要旨

本論文では，医療分野や産業用 CT で得られた高解像度な 3 次元画像に対して，3D テクスチャ圧縮と並列化プログラミング技術を組み合わせたテクスチャベースの高速なボリュームレンダリングシステムを提案する．3D テクスチャ圧縮により，表示速度を落とさずに，使用するビデオメモリ量を 4 分の 1 に減少させ，同時に CPU 内部のコアによる並列処理で，3D テクスチャ圧縮の計算時間を高速化させた．その結果，比較的 GPU 側のメモリが小さな汎用パーソナルコンピュータ上でも，複数の高解像度な 3 次元画像のリアルタイム同時表示が可能となった．また， 3D テクスチャ圧縮なしとありでのボリュームレンダリング画像の画質は，大きな相違のないことが観察された．

キーワード：3 次元画像，ボリュームレンダリング，3D テクスチャ圧縮，並列処理，マルチスレッドプログラミング

1. はじめに

近年，医療分野において，人体内部の計測や診断には，コンピュータ断層撮影（Computed Tomography：CT）がよく使用されている．CT 装置は，大別すると，医療用 CT と産業用 CT に分けられる．これらの計測装置から取得された 3 次元画像は，1）切断・移動・変形処理，2）サブトラクション，強調処理，セグメンテーションなどの画像処理，3）複数回撮像した CT 画像の合成，4）異なったモダリティの 3 次元画像表示やレジストレーション等の加工が行われて，術前計画支援などの各アプリケーションに使用される．その際には，途中過程の複数の高解像度な 3 次元画像群を，リアルタイム表示する必要性がある．

本論文では，複数の高解像度な 3 次元画像に対して，3D テクスチャ圧縮と並列化プログラミング技術を組み合わせた高速なボリュームレンダリングシステムを提案する．一般に 3 次元画像の表示技術としては，ボリュームレンダリングが使用されている．ボリュームレンダリングでは，レイキャスティング[1, 2]，テクスチャベース[3, 4, 5]，スプラッタリング[6, 7]，パーティクルベース[8]など，多くの手法が提案されている．テクスチャ

*1 岩手県立大学ソフトウェア情報学部
[〒020-0173 岩手県滝沢村巣子 152-52]
e-mail: doia@iwate-pu.ac.jp
*2 九州大学医学部
*3 （株）JFP
投稿受付：2011 年 9 月 30 日

ベースのボリュームレンダリングの特徴は，3 次元画像情報を，グラフィックスメモリにテクスチャ画像として，扱うため，表示，移動，回転，拡大などの操作が容易であることと，他のグラフィックスオブジェクト（多角形や自由曲面など）との同時表示に相性の良い点が挙げられる．しかしながら，グラフィックスメモリの容量が小さい PC では，高解像度の 3 次元画像が表示出来ず，この問題点を解決する先行研究は，十分行われていなかった．

本来，テクスチャ圧縮は，ゲームやビデオ業界におけるリアリティ向上とメモリ節約に対する強い要望から生まれた技術であったが，大容量のメモリを消費するテクスチャ-ベースのボリュームレンダリングにも適用可能である．特に 3D テクスチャ圧縮には，複数の手法（DXTC1～5）が選択でき，表示速度を落とさずに，メモリ使用量を 4 分の 1 に減少させる．しかしながら，3 次元画像の場合，ゲーム等に用いるテクスチャに比べて容量が大きいため,3Dテクスチャ圧縮に時間がかかり，その時間は無視出来ない問題点があった．そのため，我々はマルチスレッドプログラミングによる並列処理により，この 3D テクスチャ圧縮処理を高速化した．その結果，複数の高解像度な 3 次元画像に対して,比較的GPU(Graphics Processing Unit)側のメモリが小さな汎用パーソナルコンピュータ上で，解像度を落とさずに高品質なリアルタイム表示が可能となった．

2.　マルチボリュームデータ処理と表示

2. 1. ボリュームレンダリングの表示原理

ボリュームレンダリングの原理は，視点から 3 次元画像（ボリュームデータ）に対して，光線（レイ）を飛ばし，レイが交差する 3 次元画像のボクセルに対応する透明度と色情報を加算して，レイ上の減衰を考慮しながら，最終的な色を決定する．3 次元画像の画素値から透明度と色情報への対応は，3 次元画像の各画素値に対して透明度と色情報を定義した伝達関数で行われ，一般にユーザが対話的に指定する．レイの数は，投影面の画素数の数になり，各ボクセルの幅より短い区間でサンプリングを行う必要があるため，計算時間を要していた[9]．これらの処理を高速化するために，1) レイ上でサンプリング，2) サンプリングごとに色情報と透明度情報に変換，3) レイ上で色情報と透明度情報の積算計算，の各処理を，GPU のテクスチャマッピング機能とαブレンド機能で代替することが可能である．

GPU は，幾何形状の変換やラスターライゼーションなどの作業を，専用のハードウェアや集積回路で行うため，汎用の CPU（Central Processing Unit）に比べて，その処理速度を飛躍的に向上させている．また，GPU とグラフィックスアダプターに装備されたメモリ（汎用の CPU が使用する主記憶メモリと区別するため，以下では「ビデオメモリ」と呼ぶ）間の転送速度も速く，リアルタイムな処理に向いている．

2. 2. テクスチャマッピングを利用したボリュームレンダリング

3 次元画像に対して，伝達関数を用いて，各画素値（スカラ値）を，色情報（R, G, B 値）と透明度情報（α（アルファ）値と呼ばれる）に変換し，GPU 側のビデオメモリ内部に画像情報（テクスチャ画像）として保存する．次に，視線方向に垂直で，3 次元画像の断層面になるポリゴン面を多数，用意しておき，各ポリゴン面には 3 次元画像から生成したテクスチャ画像をテクスチャマッピングする．視線方向が変更された場合は，その視線方向に対して，再度，垂直なポリゴン面を作成する必要がある．また，あらかじめ，X，Y，Z 方向に対するテクスチャマッピングされたポリゴン列をあらかじめ作成しておく方式もある．視点が変更された際は，どの向きのポリゴン列を表示するかを，面の法線ベクトルおよび視線ベクトル情報を用いて，決定する（視線方向に対して，一番正面に向いているポリゴン列を使用する）．ポリゴン面を加算処理する GPU のαブレンディング機能は，各画素値の透明度αを使って，半透明なポリゴンの重ね合わせ演算をハードウェアで行う．これらの処理は，CPU に依存しないため，処理速度が速い．

2.3. テクスチャベースボリュームレンダリング

の欠点

テクスチャベースのボリュームレンダリングの問題点として，元画像データが高解像の場合，テクスチャ画像の解像度が十分でないと，表示にボケが生じる．図 1 は，画像サイズが 512x512x256 画素の CT 画像に対して，3D テクスチャの解像度を 128 と 512 の 3 乗の解像度で表示した例である．元の画像の解像度に対して，テクスチャの解像度が小さい場合，拡大等の処理では画質が劣化する．また，テクスチャ画像を格納するグラフィックスボードに搭載されたグラフィックスメモリが小さい場合，使用可能なテクスチャ画像の大きさは制限される．

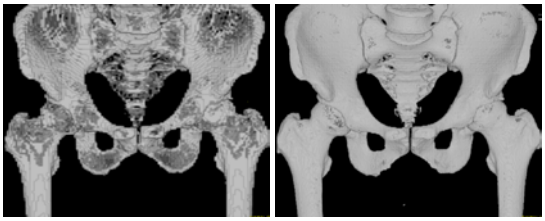

Fig. 1 Volume rendering image for 128^3 and 512^3 texture resolutions

3. 3D テクスチャ圧縮を用いたボリュームレンダリングシステム

3.1. 処理の流れと 3D テクスチャ圧縮

ビデオメモリ上に，色情報と不透明度のテクスチャ（3 次元カラー画像）を保管する際，そのテクスチャの画像圧縮を行って，ビデオメモリを有効に使用することが可能である．この技術は，ゲーム業界やビデオ業界で業界標準となり，現在では，DirectX[10]，OpenGL[11, 12]でサポートされている．この技術を用いると，比較的小さなビデオメモリのグラフィックスアダプターでも高解像度な 3 次元画像のボリュームレンダリング表示が可能となる．テクスチャ画像の圧縮アルゴリズムは，S3TC[12-14]（Microsoft 社の DirectX 6.0 で利用可能となり，DXTC（DirectX Texture Compression）とも呼ばれる）が使用されている．本圧縮アルゴリズムは，単純なメモリアクセスと固定レートによるデータ圧縮（DXT5 の場合，32 ビットの 16 ピクセルを 128 ビットデータに変換するため，元画像は，４分の１に圧縮される）であるため，ハードウェア圧縮に向いており，広く使用されるようになった．圧縮率や画像の種類によって，DXT1～DXT5 の種類があり，DXT5 が一番高品質である．OpenGL でも，OpenGL Ver. 3 からその仕様が決められた．また，当初，2 次元テクスチャを用いて，大理石や木目内部や霧などを表現することは困難であったため，表面だけではなく，内部情報も含んだ 3 次元テクスチャが考案された[11, 12]．OpenGL を使用する場合，3D テクスチャ圧縮は，CPU 側のアプリケーションプログラムで行い，その画像圧縮結果をビデオメモリ側に転送し，テクスチャマッピング表示の際に解凍して利用する．この処理で時間を要するのは，CPU 側のテクスチャ圧縮の処理部分である．そのため，我々は，処理時間を短縮するために，マルチスレッド処理による並列化プログラミングを適用した．

3.2. マルチスレッドプグラミングによる 3D テクスチャ圧縮の高速化

対象とする稼働環境は，CPU 内部に複数の計算機能（コア）を持つ，一般的な市販の PC（パーソナルコンピュータ）である．この場合，スレッドプログラミングを利用した並列化を行うことで，3 次元画像の圧縮処理を CPU の各コアで並列に行える．例えば，Intel の Core i シリーズでは，Core i3，Core i5，Core i7 と細分されており，2 個から 6 個の CPU コアを搭載している．また，Xeon は，Core i シリーズと同じアーキテクチャの，サーバ向け CPU である．さらに，「ハイパースレッディング」技術により，1 個のコア内で，複数のスレッド処理を可能にしている．AMD でも，同様に複数のコアによる計算が可能である．

システムリソースへのアクセス競合は，ユーザが制御しなければならないため，同一のデータに対して，個々のスレッドがアクセスする処理は，避ける必要がある．スレッドプログラミングは，共有メモリ・アーキテクチャのマシンでプログラムを効率よく，実行出来るため，PC クラスター構成の分散メモリ・アーキテクチャ・システムと比較すると，データ転送のロスが少なく，費用，設置面積，取扱いの面でも有利である．以下に，3D テクスチャ圧縮と並列化アルゴリズムの手法の流

れを示す.

1. 作業用のバッファ 1（4x4x4 画素）を確保
2. 圧縮結果保管用バッファ 2（3 次元画像の 4 分の 1 の大きさ）を確保

（並列処理）

3. バッファ 1 に対して，3 次元画像のブロック（4x4x4 画素）の画素列の並びを入れ替えて，格納
4. バッファ 1（4ｘ4ｘ4 画素）のブロックごとに画像圧縮の実行
5. バッファ 2 にその圧縮結果を保管

すべてのブロックを圧縮した場合,6 へ移動する. まだ,圧縮するブロックがある場合は,2 へ移動して，次のブロックを処理.

（並列処理）

6. 圧縮処理の終了後，バッファ 2 を 3D テクスチャ登録 （ビデオメモリへ転送）

ここでは，表示する 3 次元画像を同解像度の 3D テクスチャを作成･表示するものと仮定している. ステップ 3 から 5 までは,並列に計算が行われる. DXTC 圧縮した画像を OpenGL に登録する場合，専用のデータ順があるため，3 次元画像から，4x4x4 画素を 1 ブロックとする抜き取りを行い，ブロックごとに圧縮したデータを順にバッファ 2 に保管する．具体的には，3 次元画像の 1 次元配列から，（0, 0, 0）～（3, 3, 3）の 4x4x4 画素，（4, 0, 0）～（6, 3, 3）の 4x4x4 画素，（7, 0, 0）～（9, 3, 3）の 4x4x4 画素，というように切り出す．ステップ 6 の圧縮した 3D テクスチャをビデオメモリに送付する OpenGL の関数は，“glCompressedTexImage3D()”を用いている.

3.3. データ領域の分割と処理手順

一般に，並列数が多いシステムでは，通信コストの削減とロードバランスの均等のためには，領域分割や仕事量の配分方法が重要であるが，3D テクスチャ圧縮のプログラムインターフェースがブロック領域を対象にしているため，単純な固定領域のブロック分割方式を採用した．ブロック分割方式は分割アルゴリズムが簡易なため，複雑な領域分割計算が不必要であり，実装も容易である. また，1024 画像の 3 乗以上の大規模な 3 次元画像の場合，３D テクスチャ圧縮の時間が大きくなるため，あらかじめ，テクスチャ圧縮をバッチ処理で行っておき，圧縮した画像を外部ファイルに保存して，使用する方式が有効である.

4. 評価

4.1. 評価方法

使用するテクスチャのメモリ使用量は，DXTC 圧縮（S3TC 圧縮）（DXTC5）を適用した場合，画像圧縮なしに比べて，4 分の 1 に減少する．ビデオメモリの容量が小さい PC で高解像度な 3 次元画像と，実践的な使用方法を想定して，単体の 3 次元画像に対して，図 2 の分離パターンを用意して，サンプル画像を切断し，切断後の 3 次元画像を囲む立方体形式（各辺は元の画像に対して，垂直）でテクスチャを保持・作成した[15]．図 2 の分離パターンの 2 分割-1 の場合，必要となるテクスチャのメモリ量は，元のテクスチャに比較して，2 倍となる.2 分割-2 の場合は,元のメモリ量に対して，1.5 倍となる.

計算時間の計測と視覚評価に使用したパーソナルコンピュータ（PC）は，HP 製の PC（モデル名：HPZ400 Workstation），Windows 7 Professional (64bit)，Intel® Xeon®CPU W3680@3.33GHz，RAM 12.00GB，コア数 6 である．グラフィックスプロセッサは，NVIDIA®Quadra FX1800，メモリ（GDDR3 192bit I/FSDRAM） 768MB，RAMDAC/ピクセルクロック 400MHz である．CPU 計算時間の計測方法は，単独モジュールとして，実行して，clock 関数による 5 回の計測を平均している（単位は秒である）. ハイパースレッディングは使用されている.

4.2. 評価結果

表 1 は，評価に用いたサンプル画像のモダリティ（CT，MRI）と画像サイズである．各画像の X,Y 方向の画素のピッチ幅は，0.625ｍｍ，Z 方向の画素のピッチ幅は，1.0mm である．計算で使用したテクスチャの解像度は 512^3 である．本評価で使用したブロックの大きさは，4x4x4 画素領域を使用した．図 3 と図 4，図 5 と図 6，図 7 と図 8 は，それぞれ，サンプル画像 A，B，C に対する，3D テクス

チャ圧縮なしと，3D テクスチャ圧縮ありの場合の表示結果を示している．

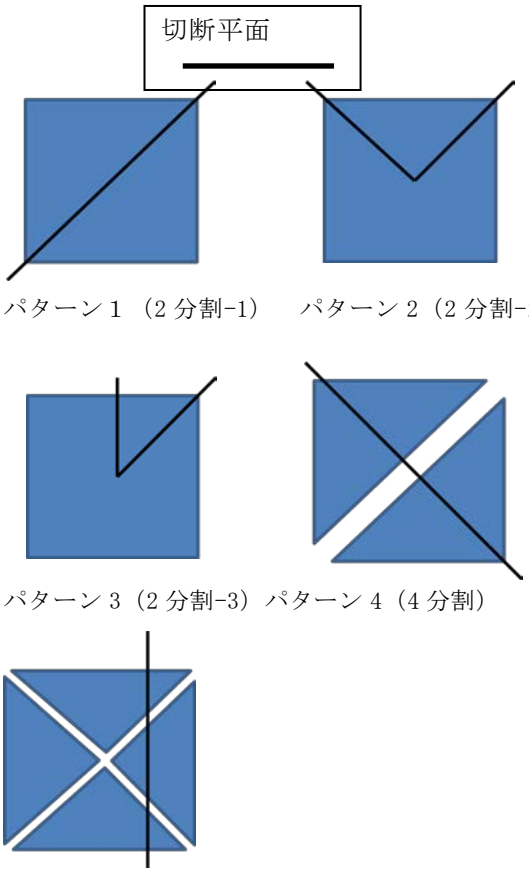

Fig. 2 Division patterns for 3D images

Table 1　Sample images for evaluation

サンプル	モダリティ	画素サイズ
A	CT	512x512x256
B	MRI	512 x 512 x 248
C	CT	1024x1024x1024

表 2 は，3D テクスチャ圧縮なし/ありで表示されたボリュームレンダリング画像（425x481 画素）の赤，緑，青成分に対して，ピーク信号対雑音比（PSNR：Peak Signal-to-Noise Ratio）[16]と二乗平均平方根（RMS：Root Mean Square）を計算した結果である．PSNR のピーク値は 255 を採用し，背景部分（黒色）は，評価の対象から除去している．

3D テクスチャ圧縮を行った場合，使用するビデオメモリ量は，4 分の 1 になっているが，視覚的な評価では,3D テクスチャ圧縮なしとありの画質は，大きく変化していないのが観察された．また，表示速度にも変化は見られなかった．表 2 は，使用した伝達関数に大きく影響を受けるが、表示される色数が少ないほど，誤差は小さくなる傾向にあった．また、各画像の評価値は，標準的な画像圧縮やビデオ圧縮と同程度な数値となっている．

Table 2　PSNR and RMS Error for Red, Green, and Blue components

サンプル	PSNR (R,G,B)	RMS(R, G, B)
A	(∞, 51.46, 51.46)	(0.0, 0.43, 0.43)
B	(29.43, 33.74, 33.97)	(8.61, 5.24, 5.01)
C	(25.22, 35.99, 24.06)	(13.98, 4.04, 15.97)

図 9，図 10，図 11 は，それぞれ，サンプル画像 A，サンプル画像 B，サンプル画像 C に対して，各分割パターンにおける，並列数（スレッド数）に対応した 3D テクスチャ圧縮の計算時間を示している．縦軸は計算時間（秒），横軸は”OFF”が圧縮処理を行なわない場合である．つまり，単純に CT 値から伝達関数によりテクスチャ画像（カラー画像）に変換される時間である．それ以外の数字列は，スレッド数を表し，スレッド数 1 は，単一のコアのみ（並列化は行わない）で圧縮処理を行った場合である．2 以上のスレッド数では，並列計算で画像圧縮を行っており，スレッド数を増加させることによる圧縮処理の高速化が確認出来る．スレッド数は，1，2，4，6，8，10 で実験したが，それ以上のスレッド数では，大きな改善は見られなかった．圧縮時間は，テクスチャの画素数や元画像の色数や分布に影響を受けるが，非圧縮でテクスチャ画像を作成する時間に比較して，1.2 倍から 3 倍程度の増加で，圧縮したテクスチャ画像を作成出来る．

興味深い点は，図 9 に比較して，同じ解像度にもかかわらず,図 10 の計算時間が大きな点である．一般に，サンプル A のような空白部分（表示は黒の部分）が多い場合や単色に近い場合，関数内部で，自動的に圧縮処理の一部を省略または簡略化していると思われる．

5. 適用事例

5.1. 術前計画支援システムへの応用

我々は，本ボリュームレンダリングシステムを 3 次元ベースの術前計画支援システムに利用している．本術前計画支援システムでは，骨切り面の設定や人工関節の配置シミュレーションが可能である．図 12 は，骨盤，大腿骨部分の CT 画像から，骨部分のみを中央の画面にボリュームレンダリング表示し，右側の 3 枚のパネルでは，x，y，z 軸方向の断面図を表している．中央画面にある緑色の線は骨切り面（2 平面）の配置を表しており，各骨切り面の配置・移動・回転は，中央画面で対話的に行える．

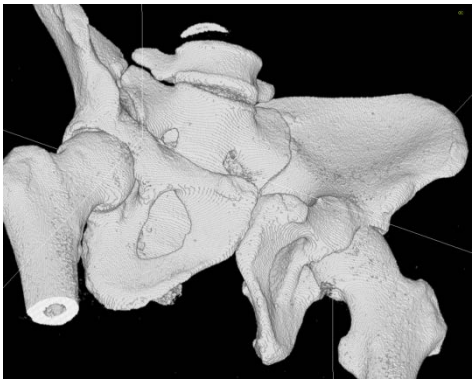

Fig. 3 Volume rendering for sample image A without 3D texture compression

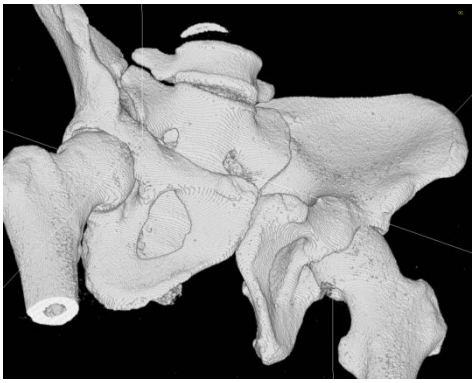

Fig. 4 Volume rendering for sample image A with 3D texture compression

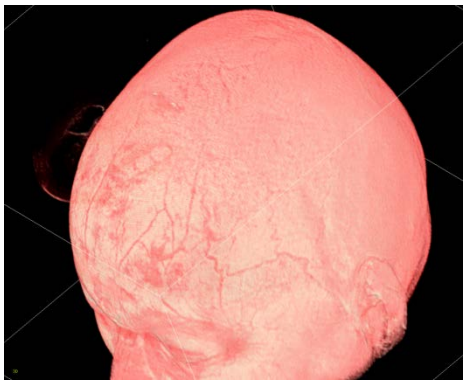

Fig. 5 Volume rendering for sample image B without 3D texture compression

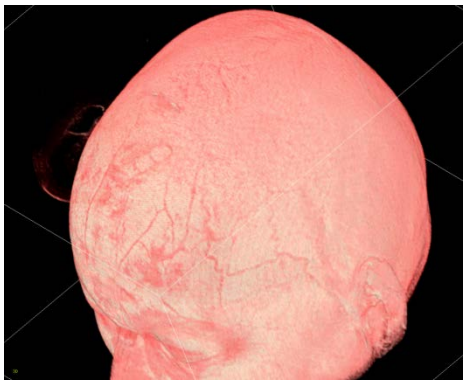

Fig. 6 Volume rendering for sample image B with 3D texture compression

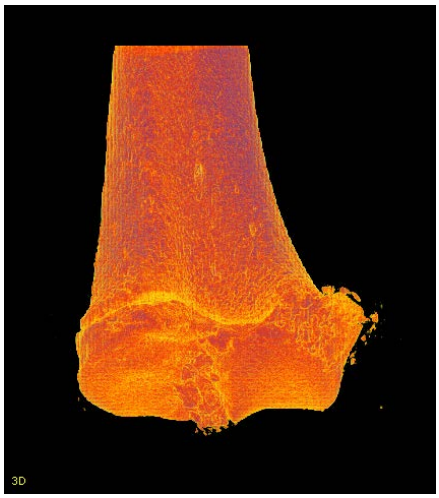

Fig. 7 Volume rendering for sample image C without 3D texture compression

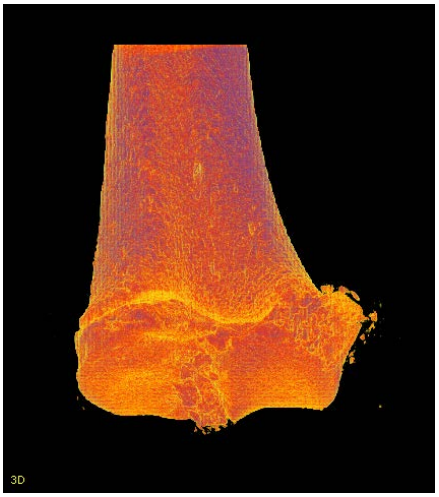

Fig. 8 Volume rendering for sample image C with 3D texture compression

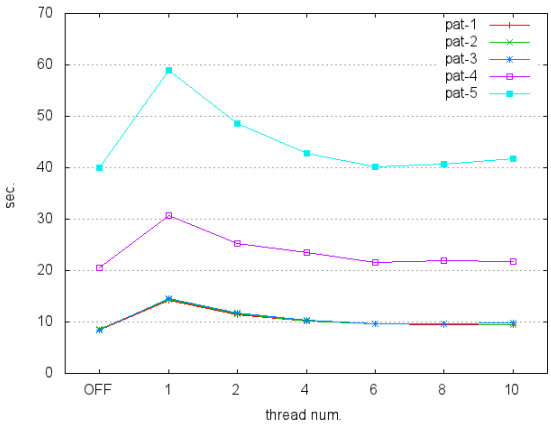

Fig. 9 Computational time for sample image A

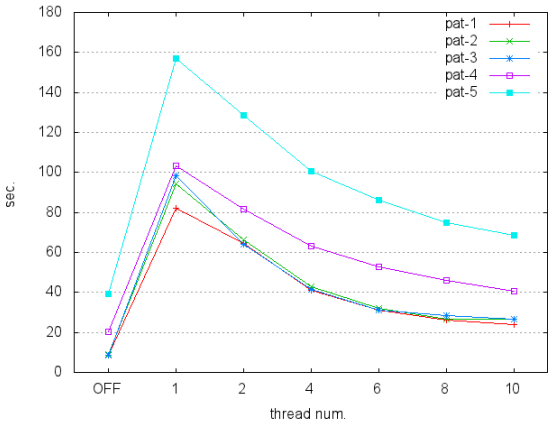

Fig. 10 Computational time for sample image B

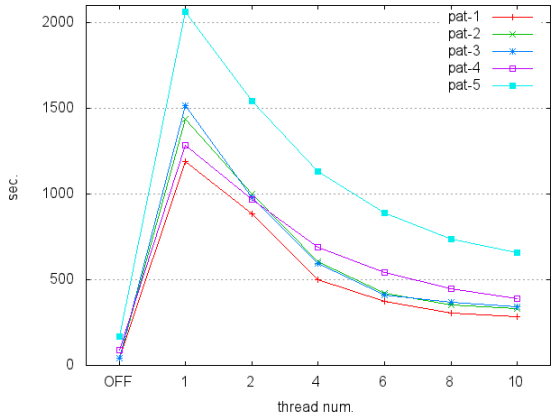

Fig. 11 Computational time for sample image D

図 13 は，その平面情報を用いて，骨盤から大腿骨部分を切断・分離し，独立した 3 次元画像として，個別にボリュームレンダリング表示している．分割時に，独立した 3 次元テクスチャとポリゴン列を生成する必要があるが，各 3 次元画像は色情報と透明度のテクスチャを貼りつけたポリゴン形状のため，移動・回転・拡大のような対話的操作が容易である．本アプリケーションは，突発性大腿骨頭壊死症を発症した若年齢の患者を対象に，人工股関節置換術の適用が困難な場合に使用される大腿骨頭前方（後方）回転骨切り術（股 ARO），股内反（大腿骨転子間彎曲内反骨切り術）（彎曲内反骨骨切り術）[17]等の骨切り術の術前計画支援にも応用している．

図 14 は，CT 画像の大腿骨骨頭の壊死部分（赤色）を 3D テクスチャ画像に追記した事例である．壊死部分の抽出は，壊死部分の表示された MRI 画像を参照して，スライス画像単位で対話的に作成している．カラー情報は各 8 ビットの RGBA 情報を保持しており，3 次元画像の形式で保存している．この様な付加情報を CT 画像や MRI 画像に追加する際は，作成した 3D テクスチャを部分的に変更するだけで，高速なボリューム表示や変更が可能である．

図 15，16 は，ポリゴン形状で近似された人工股関節を股関節部分に手前から大腿骨側に対話的に移動させた際の表示例である．人工股関節は半透明なポリゴン形状で定義されており，内部に入るにつれて，骨の不透明部分の影響で混合（融合）

される．各画素における奥行判定は，3 次元画素と人工股関節のポリゴンとの判定ではなく，3 次元画像のテクスチャを貼り付けたポリゴンとの判定であるが，3 次元テクスチャを貼り付けたポリゴン列の間隔が短いため，不自然な隠面消去の失敗例は生じていない．

6. おわりに

本稿では，3D テクスチャ圧縮技術とマルチスレッドによる並列化プログラミング技術を組み合わせたボリュームレンダリングシステムを提案した．一般に 3D テクスチャが大きくなるほど，テクスチャ圧縮の計算時間は大きくなるが，アルゴリズムを並列化することで，数倍の高速化を実現した．特に，3D テクスチャ圧縮のアルゴリズムが処理する画像領域は独立しているため，スレッド化（並列化）の効果が十分に発揮された．さらに本システムにおける 3 次元画像と人工股関節の表示能力は，CPU やグラフィックスボードの性能が向上すれば，その表示速度は自動的に向上する．

3D テクスチャ圧縮を行った場合，表示品質が重要となるが，ボリュームレンダリングの画質では，大きな差が見られなかった．また，圧縮された 3D テクスチャを解凍する際の処理時間や表示速度が重要となるが，3D テクスチャ圧縮なしと比較して，相違はなかった．これらの点で，3D テクスチャ圧縮アルゴリズム（DXT5）は，画質や表示速度に影響せずに，メモリ使用量を 4 分の 1 に減少させる点で，効果的かつ実用的である．

Fig. 12 Placement of two cutting planes in 3D image coordinates

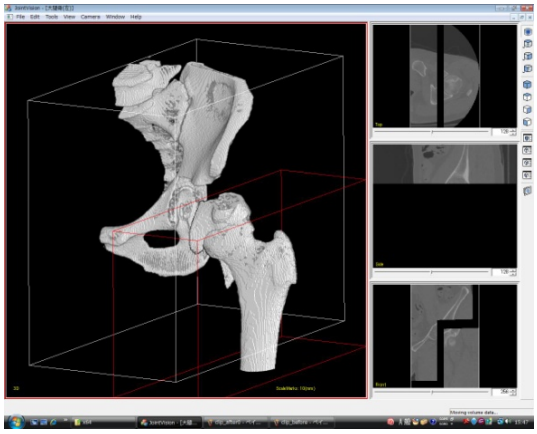

Fig. 13 The results of division calculation and volume rendering

Fig. 14 Colored CT image (red color regions show necrosis areas)

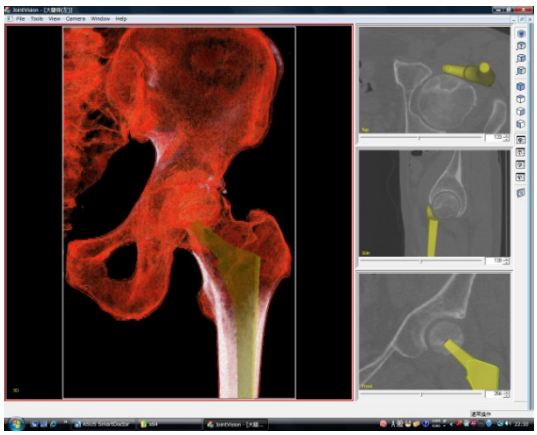

Fig. 15 Placement of a leg implant and overlay display (1)

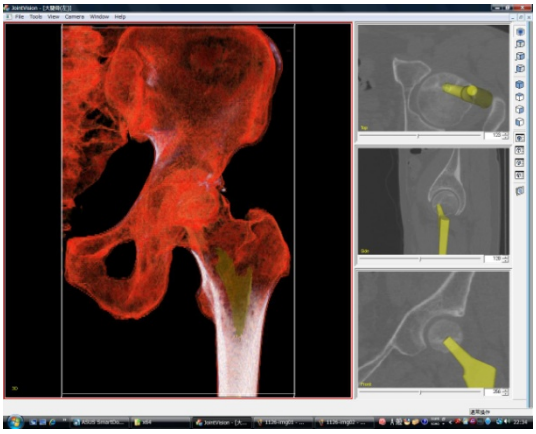

Fig. 16 Placement of a leg implant and overlay display (2)

謝辞

本研究に使用した CT 画像，MRI 画像のデータ取得では，千葉大学医学部，岩手医科大学医学部，九州大学医学部の協力を得ました．本研究の一部は，文科省科学研究費補助金(課題番号 23500562)，文科省イノベーションクラスター事業受託研究「研究開発課題名：「いわて発」高付加価値コバルト合金によるイノベーションクラスターの形成」，科学技術振興機構 A-STEP（研究開発課題名：仮想立位 CT 画像の生成とその応用）の助成により行われました．

文　献

[1]Levoy M：Efficient ray tracing of volume data. ACM Transaction s on Graphics 9(3): 245-261, 1990.

[2]吉岡政洋，森健策，末永康仁，他：ソフトウェアによる高速ボリュームレンダリング手法の開発と仮想化内視鏡システムへの適用．Medical imaging technology 19(6)：477-486，2001.

[3]Engel K, Hadwiger M, et al：Real-Time Volume Graphics. AK Peters, Ltd.; 2006.

[4]LaMar E, Hamann B, Joy K: Multiresolution Techniques for Interactive Texture-Based Volume Visualization. Proc. of IEEE Visualization 1999: 355-361, 1999.

[5] Kruger J, Westermann R: Acceleration Techniques for GPU-based Volume Rendering. Proc. of IEEE Visualiztion Conference 2003: 288-292, 2003.

[6] Zwicker M, Pfister H, Baar J, et al：EWA Volume Splatting. Proc. of IEEE Visualization 2001: 29-36, 2001.

[7]Chen W, Ren L, Zwicker M, et al：Hardware-Accelaerated Adaptive EWA Volume Splatting. Proc. of IEEE Visualization 2004: 129-137, 2004.

[8]小山田耕二：アンサンブル平均による非構造格子向けボリュームレンダリング．日本信頼性学会誌：信頼性，REAJ 誌 31(4)（通巻 176 号）：262-269，2009.

[9]藤代一成，奥富正敏，他 17 名：ビジュアル情報処理　―CG・画像処理入門―．（財）画像情報教育振興協会（CG-ARTS 協会）：115-116，2007.

[10]Microsoft Corp.：DirectX 開発者向け技術情報．Microsoft DirectX デベロッパーセンター：“http://msdn.microsoft.com/ja-jp/directx/default.aspx“,2010.

[11] Shreiner D, Woo M, Neider J, et al：Open GL Programming Guide Fifth Edition – The Official Guide to Learning OpenGL, Version 2. Addison-Wesley Pearson Education: 2006.

[12] Lengyel E：The OpenGL EXTENSIONS GUIDE. Charles River Media, INC.: 21-86, 2003.

[13] Brown P：EXT_texture_compression_s3tc. “http://www.opengl.org/registry/specs/EXT/texture_compression_s3tc.txt”, 2009.

[14] Wikipedia：S3 Texture Compression. “http://en.wikipedia.org/wiki/S3_Texture_Compression”, 2010.

[15] Doi A, Takahashi H, Mawatari T, et al：Development of a volume rendering system using 3D Texture Compression techniques on general-purpose personal computers. Proc. of iCAST2011: 461-464, 2011.

[16] Huynh-Thu Q, Ghanbari M: Scope of validity of PSNR in image/video quality assessment. *Electronics Letters* **44** (13): 800–801, 2008.

[17] 江口正雄，杉岡洋一，香月一郎，西尾篤人：日本で考案または工夫された骨切り術－転子間彎曲内反骨切り術．整形外科 MOOK 66：167-175，1993.

Development of a volume rendering system using 3D texture displaying techniques and its applications

Akio DOI[*1], Hiroki TAKAHASHIJ[*1] , Taro MAWATARI[*2], Sachio MEGA[*3]

*1 Iwate Prefectural University

*2 University of Kyushu

*3 JFP, Inc.

In this paper, we present the development of a high-speed volume rendering system that combines 3D texture compression and parallel programming techniques for rendering multiple high-resolution 3D images obtained with medical or industrial CT. The 3D texture compression reduces the memory consumption to 1/4 of the original without having a negative impact on the display speed, and parallel processing using multiple core units inside CPU make it possible to decreases the computational time of 3D texture compression, until the computational time without 3D texture compression. By using this approach, it is possible to utilize personal computers with ordinary graphics capabilities in order to display multiple high-resolution 3D images. In the evaluation, there are no large differences between the rendering images with/without 3D texture compression.

Key words: Volume Rendering, Three Dimensional Image, 3D Texture Compression, Parallel Processing, Multi-thread Programming