情報処理学会論文誌　コンピューティングシステム　Vol. 1　No. 3　41–53 (Dec. 2008)

GPUによる高速なコーンビーム再構成：円軌道装置のためのRGBAデータへの詰め込み

吉田征司[†1,*1]　伊野文彦[†1]
西野和義[†2]　萩原兼一[†1]

本稿では，コーンビームCT（Computed Tomography）のボリューム再構成を対象として，GPU（Graphics Processing Unit）を用いた高速化手法を提案する．提案手法の主な貢献は2点である．まず，入出力データをRGBA形式へ詰め込むことにより，描画処理の繰返し（レンダリングパス）数を削減し，高速な再構成を実現する．提案手法における詰め込みの特長は，詰め込みに関する制約を緩和したことにあり，円軌道のコーンビームCT装置から得られる実際の投影像を扱える．次に，ボリュームを構成するボクセルのうち，計算の一部を共有できるものがあることに着目し，それらの計算結果をボクセル間で再利用する．再利用により計算量を削減し，さらなる高速化を図る．512^2画素からなる308枚の投影像を用い，512^3ボクセルのボリュームを再構成した結果，7.4秒を要した．この実行時間は既存手法と同等であり，CPU実装と比較して15.6倍高速である．

Fast Cone Beam Reconstruction Using the GPU: RGBA Packing for Circular Orbit

Seiji Yoshida,[†1,*1] Fumihiko Ino,[†1]
Kazuyoshi Nishino[†2] and Kenichi Hagihara[†1]

This paper presents a GPU-accelerated volume reconstruction method for cone beam computed tomography (CT). Our method has two main contributions. The first one is an RGBA packing scheme that realizes fast reconstruction by reducing the number of rendering passes, namely the number of loops needed for image drawing. Our packing scheme relaxes restrictions on data packing, allowing us to deal with real projections obtained from a circular cone-beam CT. The other one is a data reuse scheme that omits computation by sharing computational results between voxels in the volume. This data reuse scheme reduces the computational amount to achieve further acceleration. As a result, the proposed method takes 7.4 seconds to reconstruct a 512^3 voxel volume from 308 512^2-pixel projection images. This execution time is competitive against previous results, achieving a 15.6-fold speedup over a CPU implementation.

1. はじめに

GPU[1]（Graphics Processing Unit）とは，グラフィクス処理の高速化を目的としたチップである．近年，性能が著しく向上していて，単精度であるものの，CPUを超える浮動小数点演算性能を安価に提供している．また，プログラマビリティが向上しており，従来のグラフィクス処理だけでなく汎用計算にGPUを応用する試みが注目されている[2]．

一方，コーンビームCT（Computed Tomography）はCT撮影法の一種である．撮影対象を中心として，X線源と平面状の検出器を回転させながら撮影する．この際，円錐状に広がるX線を撮影対象に照射し，検出器より得られる複数枚の2次元画像（投影像）から撮影対象の3次元データ（ボリューム）を構築する．この構築のことを再構成と呼ぶ．

コーンビームCTを実現する装置として可動式のCアームがある．Cアームを固定式のCT装置と比較すると，前者は装置を手術台などに近づけて術中や治療中に撮影することが容易である．このような用途では，時間制約が強く，実時間の再構成が不可欠である．さらに，技術革新とともに，高精細な投影像を高速に得ることが可能になり，再構成に要する時間の短縮が求められている．たとえば，現在のコーンビームCTが秒間30〜50枚の投影像を撮影できるのに対し，CPUが逆投影できる投影像は秒間2.6枚にとどまる[3]．

そこで，高速な再構成を目指して，GPU[3)–6)]を含め，PCクラスタ[7),8)]，FPGA[9]あるいはCell[10]などの様々なハードウェアを用いた研究が遂行されている．これらのうち，最も高速な手法AG-GPU[3]は，市販のGPUを駆使し，撮影とほぼ同じスループット（秒間52.5枚）の再構成を実現している．しかし，この手法はGPUでの高速化に寄与する入出力データの詰め込みについて言及していない．一方，同一著者らによるRapidCT[4]はAG-GPUと同じ性能を達成していて，データの詰め込みに言及している．しかし，この詰め込み手法は，平行ビームのX線で撮影したかのように投影像をあらかじめリビニング[11]

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†2 株式会社島津製作所
Shimadzu Corporation
*1 現在，株式会社日立メディコ
Presently with Hitachi Medical Corporation

© 2008 Information Processing Society of Japan

(rebinning) する必要がある．さらに，彼らの必要とするリビニングは，円軌道のコーンビーム CT に対し原理的に適用できない（付録 A.1 参照）．リビニングを省く場合，入出力データを詰め込めず（後述），性能が 1/3〜1/2 に低下してしまう[4]．

本研究では，この課題を解決することを目的として，GPU を用いてリビニング前の投影像を直接再構成できる高速化手法を提案する．提案手法は，文献 3)，4) と同様にコーンビーム CT を対象とし，その再構成アルゴリズムである Feldkamp 法[12] を高速化する．

高速化の主なアイデアは，入出力データの詰め込みおよび計算結果の再利用である．前者は，入出力データに対し，4 つのスカラ値を 1 つの RGBA 値に格納することにより，描画処理の繰返し（レンダリングパス）数を削減する．その際，投影像を構成する各画素の座標を変えないよう，4 枚のスカラ投影像を 1 枚の RGBA 投影像に格納する．平行ビームを前提として投影像における画素の座標を変更する既存手法と比較して，提案手法は円軌道のコーンビーム CT に対する適用を可能にする．一方，後者は再構成の計算量を削減し，さらなる高速化を図る．具体的には，ボリュームを構成するボクセルのうち，計算の一部を共有できるものがあることに着目し，それらの計算結果をボクセル間で再利用することにより計算を省く．

以降では，まず 2 章で関連研究を紹介し，3 章で GPU のパイプラインを示す．次に，4 章で提案手法の基となる Feldkamp 法について述べる．5 章で提案手法について述べ，6 章で評価実験の結果を示す．最後に，7 章で本稿をまとめる．

2. 関 連 研 究

GPU を用いた実装としては，Cabral ら[13] が Feldkamp 法による再構成を初めて実装した．また，Mueller ら[5),6)] は，繰返し型アルゴリズムである SART を GPU 上に実装している．しかし，GPU の計算精度が CPU よりも低いため，多くの処理を CPU が担う必要があり，その速度向上は限定的である[14]．

そこで，彼らは Feldkamp 法を基にした手法[3),4)] を提案している．文献 3) は，GPU の持つ 2 種類のプロセッサ FP（Fragment Processor）および VP（Vertex Processor）のうち，FP のみを用いる手法 MP-GPU および両者を用いてプロセッサ間の負荷分散を図る手法 AG-GPU を示している．後者は，FP における線形の計算が VP に移せること[15] に着目していて（3 章で後述），秒間 14.5 枚の再構成を秒間 40.4 枚に高速化する．さらに，ボリューム内の関心領域（ROI: Region Of Interest）をステンシルバッファであらかじめ指定できれば，EFK（Early Fragment Kill）を用いて計算量とともに実行時間を削減できることを示している．

詰め込みに関しては，投影角度が直交する 4 枚のスカラ投影像を RGBA 投影像に詰め込む手法[6] や，軌道面と垂直な方向に連続する 4 ボクセルを詰め込む手法[4] がある．しかし，前者は投影像の撮影間隔が等しいことを前提にする．実際の装置では，正確さに関して機械的な限界があり，投影像の撮影間隔は均等ではなく，回転軌道も揺れて正円ではない．また，後者も投影像を平行ビームで撮影したかのようにあらかじめリビニングする必要がある．そのような投影像は計算機上の投影シミュレーションで生成できても，実際のコーンビーム CT から得ることはできない（付録 A.1 参照）．

Schiwietz ら[16] は，再構成の主要な処理をすべて GPU 上に実装し，パイプライン状に実行する手法を提案している．また，Riabkov ら[17] は，Feldkamp 法を対象に GPU 上のビデオメモリに保持するデータに関して 2 通りの実装があることを示している．ただし，これらは Mueller らの手法ほど高速ではない．李ら[18] は，nVIDIA 社の GPU で動作する統合開発環境 CUDA[19]（Compute Unified Device Architecture）を用い，Feldkamp 法の逆投影処理を 3.6 秒で実行している．しかし，再構成全体の実行時間は明らかでない．また，EFK などのグラフィクス固有の機能は CUDA と併用できない．

上記の GPU による高速化に加え，Cell[10]，FPGA[9]，グリッド[20] や PC クラスタ[7),8)] による試みもある．しかし，文献 3) が示すように，これらのハードウェアは GPU が引き出す性能を上回っていない．また，一般に CT 装置が再構成結果の管理や閲覧のための PC を備えていることから，GPU による高速化手法はその PC に内蔵カードを追加するだけでよく，同一のシステム構成を維持できる点も都合がよい．GPU が娯楽分野の要望に応え，今後もこれまでの性能向上を維持できれば，将来においても有望な高性能ハードウェアである．

3. GPU のグラフィクスパイプライン

図 1 に，GPU のグラフィクスパイプラインを示す．GPU は，ビデオメモリに加え，プログラム可能な 2 つの演算器 VP および FP を持ち，これらはラスタライザを挟んでパイプラインを構成している．このパイプラインは，図形の形状や位置などを表す頂点データを入力とし，画像を構成する画素を出力する．本来のグラフィクス処理においては，画素の出力先は画面を表すフレームバッファである．GPU を汎用処理に用いる場合は，ビデオメモリ上の他のバッファ[21] へ画素を出力する（オフスクリーンレンダリング）．

上記のパイプラインにおいて，VP は頂点データに対する処理を並列実行する．ラスタラ

© 2008 Information Processing Society of Japan

イザは，頂点ごとの計算結果を基に，頂点が囲む領域からフラグメントを生成する．ここで，フラグメントは画素に対応し，頂点ごとの計算結果（色情報や座標など）を線形補間した値を持つ．フラグメントごとの処理は FP が並列実行する．たとえば，テクスチャと呼ばれる模様を表す画像を張り付けるなどの処理を行う．VP および FP における処理は，それぞれ頂点プログラムおよびフラグメントプログラムとして与える．各々のプログラムでは，1 つの頂点および 1 つのフラグメントに対する処理を記述する．

なお，フラグメントは頂点ごとの計算結果を線形補間した値を持つことから，フラグメントに対する線形の計算は頂点ごとの計算に置き換えることができる．すなわち，フラグメントプログラムの命令を頂点プログラムに移動できる．フラグメント数よりも頂点数の方が少ない場合，この命令移動はプロセッサ間の負荷分散だけでなく，計算量を削減する効果もある[15]．

また，EFK は描画領域を限定することにより，ラスタライザから FP に流入するフラグメントの数を削減する手法である．FP は，フラグメントごとにフラグメントプログラムを起動するため，EFK の改善効果は流入を回避できたフラグメントの数に比例する．この回避により，ビデオメモリおよび FP 間のデータ転送量および FP の計算量を削減できる．

GPU はビデオメモリ上のデータを参照できるが，主記憶を直接参照できない．したがって，描画の前に，データを主記憶からビデオメモリに転送（ダウンロード）する必要がある．逆に，CPU が GPU の計算結果を参照するときは，データをビデオメモリから主記憶へ転送（リードバック）する必要がある．

GPU を汎用処理に用いる場合，入出力データを 2 次元テクスチャとして保持し，これを 2 次元配列と見なして FP で並列処理することが典型的である．なお，GPU は 2 次元テクスチャに最適化されているため，3 次元データも 2 次元テクスチャとして格納する方が高速である．また，テクスチャには RGBA 版があり，1 テクセルあたり 4 要素の RGBA 値を格納できる．各要素は 32 ビット浮動小数点数として表せる．

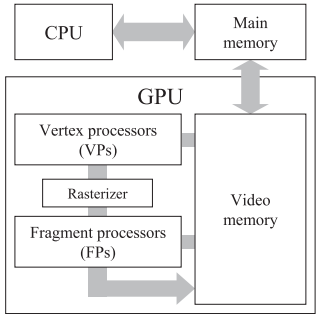


図 1　GPU のグラフィクスパイプライン
Fig. 1　Graphics pipeline in the GPU.

4. Feldkamp 法によるコーンビーム再構成

Feldkamp 法[12] は，投影像のフィルタリングおよび逆投影処理で構成される．I 枚の投影像 $P_1, P_2, \ldots, P_I$ を入力としてボリュームを再構成することを考える．**図 2** に，i（$1 \leq i \leq I$）番目の投影像 P_i およびボリュームにおける座標系を示す．xyz 空間上の立方体はボリュームを表し，その中心から X 線源までの距離は d_i である．一方，uv 平面上の長方形は検出器を表し，その中心から X 線源までの距離は d' である．この図では，角度 θ_i の方角へ d_i だけ離れた X 線源から物体を照射し，距離 d' にある検出器で投影像 P_i を得ている．ここで，θ_i は X 線源およびボリュームの中心を結んだ直線と X 軸のなす角度である．

まず，フィルタリングについて述べる．i 番目（$1 \leq i \leq I$）の投影像 P_i が与えられたとき，そのフィルタ済投影像 Q_i は，P_i およびフィルタ C を u 方向に畳み込み積分したものである．このとき，Q_i 上の座標 (u, v) における画素値 $Q_i(u, v)$ を次式で与える．

$$Q_i(u,v) = C(u) * (W_1(u,v,i) P_i(u,v)) \tag{1}$$

ここで，$P_i(u, v)$ は P_i 上の座標 (u, v) における画素値であり，重み W_1 は次式で与える．

$$W_1(u,v,i) = \frac{d'}{\sqrt{d'^2 + u^2 + v^2}} \tag{2}$$

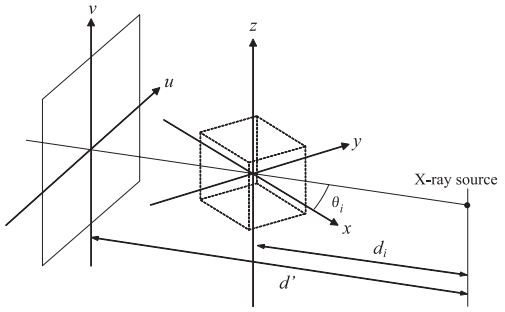


図 2　Feldkamp 法における座標系
Fig. 2　Coordinate system in the Feldkamp method.

© 2008 Information Processing Society of Japan

また，フィルタ関数 $C(u)$ には Shepp-Logan フィルタ[22] がよく用いられ，次式で与える．

$$C(u) = \frac{2}{\pi^2(1-4u^2)} \tag{3}$$

以上をまとめると，P_i と C の畳み込み積分は，フィルタサイズ K を用いて次式で表せる．

$$Q_i(u,v) = \sum_{k=-K}^{K} C(k) W_1(u-k,v,i) P_i(u-k,v) \tag{4}$$

次に，逆投影処理について述べる．I 枚のフィルタ済投影像 $Q_1, Q_2, \ldots, Q_I$ を逆投影し，ボリューム F を再構成する．ボリューム内の座標 (x,y,z) のボクセル値 $F(x,y,z)$ を次式で与える．

$$F(x,y,z) = \frac{1}{2\pi I} \sum_{i=1}^{I} W_2(x,y,i) Q_i(u(x,y,i), v(x,y,z,i)) \tag{5}$$

ここで，座標 (u,v) および重み W_2 は，x，y，z および i に依存し，次式で与える．

$$u(x,y,i) = \frac{d'(-x\sin\theta_i + y\cos\theta_i)}{d_i - x\cos\theta_i - y\sin\theta_i} \tag{6}$$

$$v(x,y,z,i) = \frac{d'z}{d_i - x\cos\theta_i - y\sin\theta_i} \tag{7}$$

$$W_2(x,y,i) = \left(\frac{d_i}{d_i - x\cos\theta_i - y\sin\theta_i} \right)^2 \tag{8}$$

以降では，

$$f_i(x,y,z) = W_2(x,y,i) Q_i(u(x,y,i), v(x,y,z,i)) \tag{9}$$

とする．

5. 提 案 手 法

本章では，提案手法の基礎となる手法を示したのち，核となる詰め込みおよび再利用について説明する．

5.1 基本的な手法

図 3 に，基本的な手法における再構成処理の概要を示す．GPU への入力となる投影像 P_1〜P_I の各々はテクスチャとして格納する．一方，出力となるボリュームは xy 平面に平行なスライス（断面）の集合で表し，スライスの各々をテクスチャとして格納する．なお，ビデオメモリ上に保持する入出力データは，すべての投影像 P_1〜P_I および 1 枚のスライスである．ボリュームの全体は主記憶側に保持し，計算結果の出力先となるスライスのみをビデオメモリ上に保持することにより，ビデオメモリの消費量を抑える．以降では，ボクセル

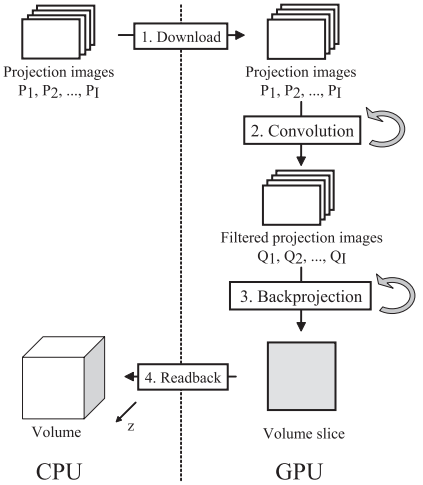


図 3　再構成処理の概要
Fig. 3　Overview of reconstruction process.

(x,y,z) を含むスライスを F_z と表す（$0 \le z \le N-1$）．

基本的な手法は，下記の手順で再構成を実現する（**図 4**）．

(1)　入力データのダウンロード．すべての投影像 P_1〜P_I を主記憶からビデオメモリへ転送する．

(2)　畳み込み積分．P_1〜P_I の各々に畳み込み積分を施し，フィルタ済投影像 Q_1〜Q_I を得る．

(3)　すべての z（$0 \le z \le N-1$）に対し，手順 (4) および (5) を順に処理する．

(4)　逆投影．式 (5) に基づいて，Q_1〜Q_I を順にスライス F_z に逆投影する．

(5)　出力データのリードバック．F_z をビデオメモリから主記憶へ転送する．

図 4 に示すように，CPU 側の処理は z および i（スライスおよび投影像）に関する 2 重ループで構成されていて，ループ本体は 1 枚の投影像 Q_i を 1 枚のスライス F_z へ逆投影するための描画処理である．したがって，再構成全体の描画回数（レンダリングパス）は IN 回である．

一方，GPU 側は上記の描画処理を担当し，スライス上のボクセルを並列処理する．このときの各フラグメントは F_z 上のボクセル (x,y,z) に対応する．フラグメントごとに，以下

© 2008 Information Processing Society of Japan

```
入力：投影像 P_1, P_2, ..., P_I，フィルタサイズ K および
      撮影パラメータ d', d_1, d_2, ..., d_I, θ_1, θ_2, ..., θ_I
出力：ボリュームを構成するスライス F_0, F_1, ..., F_{N-1}
BasicReconstruction()
begin
  投影像 P_1, ..., P_I をダウンロード;
  畳み込み積分のフラグメントプログラムをセット;
  for i = 1 to I do begin
    Q_i ← Convolution(P_i, K); // 描画
  end;
  逆投影のフラグメントプログラムをセット;
  for z = 0 to N − 1 do begin
    for i = 1 to I do begin
      F_z ← Backprojection(Q_i, d_i, θ_i, d', z); // 描画
    end;
    スライス F_z をリードバック;
  end;
end
function Convolution(P_i, K)
begin
  // u および v は GPU がフラグメントごとに適切に設定
  out ← 0;
  for k = −K to K do begin
    W_1 および C を計算; // 式 (2) および (3)
    val ← P_i の座標 (u − k, v) における画素値;
    out ← out + C · W_1 · val;
  end;
  return out;
end
function Backprojection(Q_i, d_i, θ_i, d', z)
begin
  // x および y は GPU がフラグメントごとに適切に設定
  分母 d_i − x cos θ_i − y sin θ_i を計算;
  u，v および W_2 を計算; // 式 (6)〜(8)
  u および v をテクスチャ座標 coord に変換;
  val ← Q_i の座標 coord における画素値; // 自動補間
  val ← W_2 · val;
  current ← F_z の座標 (x, y) におけるボクセル値;
  return val + current;
end
```

図 4　基本的な手法の疑似コード
Fig. 4　Pseoudocode of the basic method.

にあげる 4 つの処理を行い，$f_i(x,y,z)$ を出力する．

- 座標変換．(x,y,z) に対応する Q_i の座標 (u,v) を，式 (6) および (7) に基づいて計算する．(u,v) は，Q_i の中心を原点とし，実空間における 1 画素の大きさがボリュームおよび投影像間で等しい場合の座標である．一方，投影像を保持するテクスチャは左上端に原点を持ち，大きさもボリュームと投影像間で異なる．そこで，(u,v) を実空間における 1 画素の大きさに正規化し，左上端を原点に持つテクスチャ座標 coord に変換する．
- テクスチャ参照．coord におけるテクセル値を参照し，画素値 $Q_i(u,v)$ を得る．ここで，u および v は実数であるため，整数座標上にのみ値を持つ Q_i の画素値を補間する必要がある．この補間には，GPU がハードウェアの機能として提供するバイリニア補間を用いる．バイリニア補間は，周囲 4 点の重み付き和で補間値を表すが，この計算は専用のハードウェア（テクスチャユニット）が自動的に処理するため，高速である．
- 重み計算．式 (8) に基づいて W_2 を計算し，式 (9) により $f_i(x,y,z)$ を算出する．
- 積算．現在のボクセル値 $\sum_{j=1}^{i-1} f_j(x,y,z)$ に $f_i(x,y,z)$ を加算し，新しい値として出力する．

5.2　投影像の詰め込み

逆投影処理は，異なる投影像間にデータ依存を持たない．したがって，複数の投影像を同時に逆投影できる．そこで，4 枚の投影像 Q_i，Q_{i+1}，Q_{i+2} および Q_{i+3} を 1 枚の RGBA テクスチャに格納し，それらをベクトル演算で 1 度に処理できれば，逆投影を高速化できる．この際，RGBA テクスチャにおける座標 (u,v) に，各投影像における同一座標の画素 $Q_i(u,v)$，$Q_{i+1}(u,v)$，$Q_{i+2}(u,v)$ および $Q_{i+3}(u,v)$ を格納することにより，元の投影像における画素の位置関係を維持する．さらに，基本的な実装における GPU の処理を以下のように変更する．

- ベクトル演算の適用．4 枚の投影像 Q_i〜Q_{i+3} に対する u，v および W_2 をベクトル演算で計算する．これにより，1 回の描画でボクセル (x,y,z) に対して $u(x,y,i)$〜$u(x,y,i+3)$，$v(x,y,z,i)$〜$v(x,y,z,i+3)$ および $W_2(x,y,i)$〜$W_2(x,y,i+3)$ を得ることになる．同様に，式 (9) による $f_i(x,y,z)$〜$f_{i+3}(x,y,z)$ の算出や現在値へのこれらの積算もベクトル演算できる．
- テクスチャ参照．u，v および W_2 を用いてテクスチャの 4 点を参照し，4 枚の投影像ごとの画素値 $Q_i(u(x,y,i),v(x,y,z,i))$〜$Q_{i+3}(u(x,y,i+3),v(x,y,z,i+3))$ を得る．ここで，テクスチャ参照はスカラ単位であることに注意されたい．この理由は，複数の

© 2008 Information Processing Society of Japan

座標を RGBA 値に格納して複数の画素を 1 度に参照する手段がないためである．したがって，上記の 4 点を逐次参照する．ただし，i に関して連続する投影像を詰め込んでいるため，各々の (u, v) は近接することが多い．この場合，テクスチャキャッシュにより，2 点目以降の参照を高速化できる．

データの詰め込みにより GPU は 1 回の描画で 4 枚のスカラ投影像を逆投影できる．つまり，基本的な実装と比較して，i に関するループを 1/4 にできるため，レンダリングパスは $IN/4$ 回に削減できる．さらに，描画のための前処理（入出力テクスチャの切替えなど）に要するオーバヘッドを描画そのものの実行時間に対して低減させることができるため，GPU の持つ性能を引き出せて有効である．

提案手法では，参照すべき座標 (u, v) が投影像 Q_i〜Q_{i+3} ごとに異なり規則性がない．したがって，詰め込みの前後においてテクスチャ参照の回数は変わらない．一方，RapidCT [4] の詰め込みでは，参照すべき画素の v 座標がボクセルの z 座標と同一であるため，1 回のテクスチャ参照で 4 つのボクセルに対する逆投影を処理できる．ゆえに，参照の回数を 1/4 に削減できる．GPU がベクトル演算器で構成されていた頃のように，1 回のテクスチャ参照で RGBA のすべてを読み出せば，提案手法は RapidCT と比較して 4 倍のデータをビデオメモリから転送する必要がある．しかし，このような無駄を省くために，現在のアーキテクチャはスカラ演算器で構成されている [23]．さらに，コンパイラがスカラ単位の参照命令に置き換えて不要な参照を除去するため，詰め込みにより転送量は増大しない．

5.3　計算結果の再利用

z 軸と平行な直線上に並ぶボクセルを考える．これらのボクセルは，同一の x 座標および y 座標を持ち，z 座標のみが異なる．つまり，これらは異なるスライス上に存在するが，スライス内の位置は同じである．この場合，式 (6)〜(8) において，z に依存しない部分の計算結果は同一である．具体的には，下記の部分が z に依存しない．

- $u(x, y, i)$
- $v(x, y, z, i)$ の分母
- $W_2(x, y, i)$

そこで，これらの計算結果をボクセル間で再利用するために，**図 5** のようにフラグメントプログラムを修正する．

まず，複数のスライスを一度に処理できるように，入力データに加えて出力データも RGBA 形式に詰め込む．つまり，計算結果の出力先をスカラから RGBA バッファに変更する．RGBA 値は 4 要素で構成されているため，提案手法は 4 枚のスライス F_z〜F_{z+3} 内で計算結果を

```
入力：投影像 P_1, P_2, ..., P_I，フィルタサイズ K および
      撮影パラメータ d', d_1, d_2, ..., d_I, θ_1, θ_2, ..., θ_I
出力：ボリュームを構成するスライス F_0, F_1, ..., F_{N-1}
ProposedReconstruction()
begin
  投影像 P_1, ..., P_I をダウンロード;
  畳み込み積分のフラグメントプログラムをセット;
  for m = 0 to I/4 − 1 do begin
    i = 4m + 1;
    (Q_i, Q_{i+1}, Q_{i+2}, Q_{i+3})
      ← VecConvolution(P_i, P_{i+1}, P_{i+2}, P_{i+3}, K);
  end;
  逆投影のフラグメントプログラムをセット;
  for n = 0 to N/4 − 1 do begin
    z = 4n;
    for m = 0 to I/4 − 1 do begin
      i = 4m + 1;
      (F_z, F_{z+1}, F_{z+2}, F_{z+3})
        ← VecBackproj((Q_i, Q_{i+1}, Q_{i+2}, Q_{i+3}),
          (d_i, d_{i+1}, d_{i+2}, d_{i+3}),
          (θ_i, θ_{i+1}, θ_{i+2}, θ_{i+3}), d', z); // 描画
    end;
    スライス F_z〜F_{z+3} をリードバック;
  end;
end

function VecBackproj((Q_i, Q_{i+1}, Q_{i+2}, Q_{i+3}),
  d, θ, d', z)
begin
  // x および y は GPU がフラグメントごとに適切に設定
  // RGBA 値を太字で a のように表し，
  // その要素を a_1〜a_4 で表す
  分母 d − x cos θ − y sin θ を計算;
  u および W_2 を計算; // 式 (6)，(8)
  for n = 0 to 3 do begin
    v を計算; // 式 (7)．z 座標は z + n
    u および v をテクスチャ座標 coord_1〜coord_4 に変換;
    for j = 1 to 4 do begin
      val_j ← Q_{i+j} の座標 coord_j における画素値;
    end;
    val ← W_2 · val;
    out_{n+1} ← val_1 + val_2 + val_3 + val_4;
  end;
  current
    ← F_z〜F_{z+3} の座標 (x, y) におけるボクセル値;
  return out + current;
end
```

図 5　提案手法の疑似コード
Fig. 5　Pseoudocode of the proposed method.

© 2008 Information Processing Society of Japan

再利用することになる．

出力データの詰め込みを施したうえで，図 5 のフラグメントプログラムでは，最初に u，v および W_2 に共通する分母 $d_i - x\cos\theta_i - y\sin\theta_i$ を計算する．その後，z に依存しない u および W_2 を計算する．スライスごとの計算は，F_z から順に F_{z+3} までを逐次処理する．最後に，計算結果 $f_i(x,y,z)$，$f_i(x,y,z+1)$，$f_i(x,y,z+2)$ および $f_i(x,y,z+3)$ を変数 out に格納し，現在のボクセル値に加算する．

出力データの詰め込みにより，i に関するループに加え，z に関するループも 1/4 に削減できる．結果として，レンダリングパスは $IN/16$ 回になる．なお，図 5 を図 4 と比較すると，フラグメントプログラム内のループは逆に増加している．つまり，フラグメント 1 つあたりの計算量は多くなる．このことは，より大きな粒度で並列処理できることを意味する．特に，描画のための前処理のオーバヘッドが描画そのものに対して大きい場合，高速化に役立つ．

6. 評 価 実 験

提案手法の性能および再構成結果の画質を評価するために，GPU を用いて実験した結果を示す．

6.1 実験の手順

実験に用いた PC は，CPU として Core 2 Extreme 2.93 GHz を持ち，GPU として nVIDIA GeForce 8800 GTX を備える．また，主記憶容量は 2 GB であり，ビデオメモリ容量が 768 MB である．OS は Windows XP であり，ビデオドライバのバージョンは 162.01 である．提案手法の実装には，C++言語，OpenGL ライブラリ[24] および Cg（C for Graphics）言語[25] を用いた．また，ボリュームの各スライスを保持する出力用バッファには Frame Buffer Object[21] を用いた．コンパイラには Visual Studio 2005 を用い，コンパイルオプションは/O2 /Oi /Ot である．また，グラフィクス関連の開発環境として，nVIDIA SDK 9.5 および Cg 1.5 を用いた．

表 1 に，実験に用いた実装の一覧を示す．これらのうち，実装 A〜D は提案手法を組み合わせたものである．実装 E は，実装 D に文献 3) の負荷分散を実装したものである．具体的には，u の分子，v の分子および分母 $d_i - x\cos\theta - y\sin\theta$ の計算をフラグメントプログラムから頂点プログラムに移した．実装 F は，実装 D に文献 3) の EFK を組み込んだものである．

表 2 および**図 6** に，実験に用いた投影像の仕様およびそれらの再構成結果を示す．これらは，実際の装置で撮影したファントムデータおよび動脈瘤を含む人体頭部を撮影した臨床データである．なお，フィルタサイズは $K = 300$ とした．

表 1　実験に用いた実装
Table 1　Implementations used for experiments.

手法	再利用	詰め込み	負荷分散	EFK
実装 A	×	×	×	×
実装 B	○	×	×	×
実装 C	×	○	×	×
実装 D	○	○	×	×
実装 E	○	○	○	×
実装 F	○	○	×	○
MP-GPU [3]	×	?	×	×
AG-GPU [3]	×	?	○	×
RapidCT [4]	×	○	○	×
AG-GPU w/ EFK [3]	×	?	○	○

表 2　実験に用いた投影像の仕様
Table 2　Specification of projection images used for experiments.

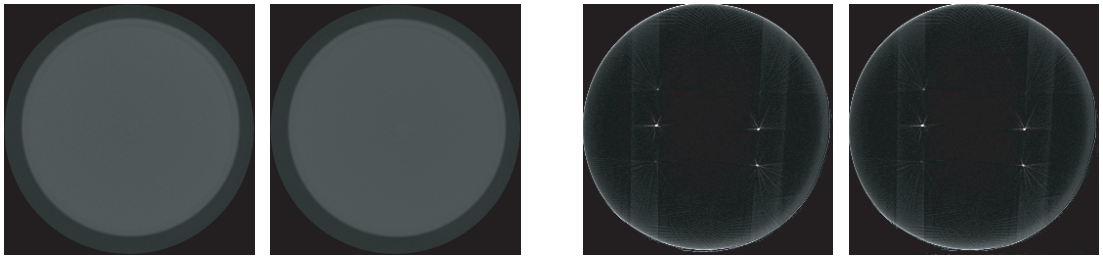
パラメータ	データ 1 アクリル円柱	データ 2 金属球	データ 3 人体頭部
U：投影像サイズ横	512	512	512
V：投影像サイズ縦	512	640	640
I：投影像数	308	92	96
N：ボリュームサイズ	512	256	512

6.2 実行時間の評価

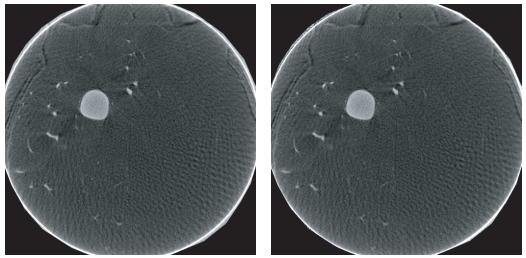
図 7 に，各手法の実行時間 T を示す．ここで，提案手法の T は，データ 1 に対し，投影像の転送から再構成で得たボリュームを主記憶へ転送し終えるまでの時間を指す．一方，既存手法の T は，文献 4) における $I = 360$ のときの実行時間 T' を基に正規化した値 $T = 308/360T'$ である．なお，双方ともに同一の GPU を用いたが，性能に大きな影響を持つドライバのバージョンが不明であることに加え，T' の計測区間が不明であるため，厳密な比較でないことに注意されたい．

実装 A〜D では，詰め込みと再利用を併用する実装 D が最速である．その実行時間は 7.4 秒であり，AG-GPU および RapidCT の 7.6 秒よりも高速である．一方，MP-GPU の 21.2 秒と比較すると実装 D は約 2.9 倍ほど高速であり，CPU に対しては 15.6 倍ほど高速である．したがって，提案手法は AG-GPU が言及していない詰め込み手法を示し，RapidCT

© 2008 Information Processing Society of Japan



(a) データ 1：アクリル円柱　　(b) データ 2：金属球



(c) データ 3：人体頭部

図 6　投影像の再構成結果（左：提案手法；右：CPU）
Fig. 6　Reconstruction results of projection images. The left-hand side is the proposed result while the right-hand side is the CPU result.

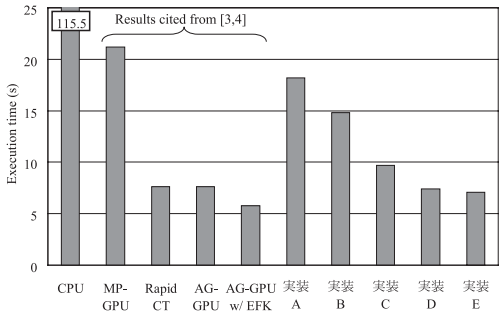


図 7　データ 1 に対する実行時間
Fig. 7　Execution time for data 1.

における詰め込みの適用条件を緩和しつつ，その性能と遜色のない実行時間を実現できている．特に，RapidCT は現実的でない平行ビームを前提にしているため，実際の装置から得られる投影像を扱える提案手法は有用である．

次に，EFK の効果を示す．**図 8** に，描画領域の割合 α を変えたときの実装 F の実行時間を示す．ここで，α は EFK を用いないときに 1 であり，すべて描画しないときに 0 である．逆投影に要する実行時間は α に比例しているが，残りの内訳はほぼ同一である．$\alpha = 0$ のとき，FP は何も処理しないが，逆投影に 1.2 秒ほど要している．この実行時間は，CPU が描画命令を発行する時間および GPU がフラグメントの処理を回避するための時間である．$\alpha = 0.25$ のとき，再構成に要する実行時間は 5.6 秒である．一方，図 7 において，EFK を併用する AG-GPU は 5.8 秒である．いずれも EFK により実行時間を 1/4 ほど削減できている．このように，AG-GPU と同様に，提案手法も EFK により実行時間を短縮できる．ただし，ドライバのバージョンに加えて，AG-GPU における α が不明であるため，同一条

© 2008 Information Processing Society of Japan

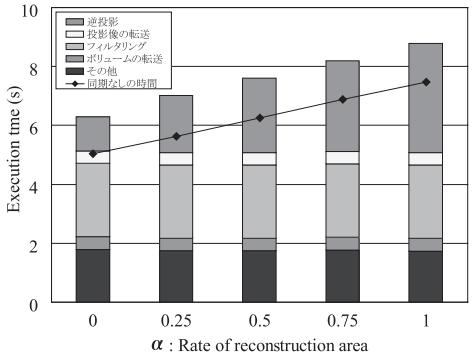


図 8　EFK により ROI の大きさを変えたときの実行時間（データ 1）
Fig. 8　Execution time for different ROI sizes using EFK.

表 3　データ 1 に対する実行時間の内訳（秒）
Table 3　Breakdown of execution time for data 1 (s).

内訳	実装				
	A	B	C	D	E
T_2：投影像の変換	0.2	0.2	0.5	0.5	0.5
T_3：投影像の転送	0.4	0.4	0.4	0.4	0.4
T_4：畳み込み積分	4.8	4.8	2.1	2.1	2.1
（うち描画）	4.1	4.1	1.8	1.7	1.7
T_5：逆投影	13.7	6.4	6.3	3.2	3.4
（うち描画）	12.5	6.1	6.0	3.2	3.3
T_6：ボリュームの転送	0.5	0.4	0.5	0.4	0.4
T_7：ボリュームの変換	1.0	0.9	0.9	0.6	0.6
T_1：その他	3.0	3.4	1.2	1.3	1.7
T：内訳の合計	23.5	16.4	11.8	8.5	9.1
同期なしの時間	18.2	14.8	9.7	7.4	7.1

件下での公平な比較は不可能である．なお，EFK のための初期化時間は 40 ミリ秒未満であり，そのオーバヘッドは十分に小さい．

次に，高速化への貢献の度合いを手法ごとに調べるために，実行時間の内訳を調べた．**表 3** に，提案手法の内訳を示す．畳み込み積分および逆投影の内訳のうち，描画時間は GPU における計算時間を表し，glBegin 関数から glEnd 関数までに対応する．なお，内訳を計測するためには，glFinish 関数を用いて GPU を CPU に同期させる必要があり，同期なしの結果（図 7）よりも実行時間が長くなることに注意されたい．

表 4　各手法の実効性能と実効バンド幅
Table 4　Effective performance and bandwidth of each method.

指標		実装ごとの実測値					理論値
		A	B	C	D	E	
実効性能 (GFLOPS)	演算器	117.0	112.0	62.6	99.3	61.3	345.6
	テクスチャユニット	33.9	69.4	70.6	132.3	128.3	172.8
	合計	150.9	181.5	133.1	231.6	189.6	518.4
実効バンド幅（GB/s）		28.7	37.1	59.8	70.8	68.6	86.4

表 3 より，詰め込みを施した実装 C は実装 A の実行時間を 47%ほど削減でき，再構成を 9.7 秒で終えている．一方，計算結果を再利用する実装 B は再構成を 14.8 秒で完了し，19%の削減率を達成している．前者の削減率が後者よりも大きい理由は，詰め込みが逆投影だけでなく畳み込み積分の実行時間も短縮できるためである（表 3）．このように，双方を短縮できる詰め込みの方が再利用よりも高速化に貢献できる．

一方，実装 C および D を比較すると，詰め込みを施したのちの再利用による削減率は 24%にとどまる．削減率が 47%から 24%に低下した理由は，FP およびビデオメモリ間の転送バンド幅が性能ボトルネックになるためである．**表 4** が示すように，実装 D における逆投影処理の実効バンド幅は 70.8 GB/s であり，この値は理論値 86.4 GB/s の 82%にあたる．したがって，再利用により計算量を削減したとしても，ビデオメモリからのデータの供給が計算性能に追い付かず，性能が頭打ちになっている．実際に，実効の計算性能は 99.3 GFLOPS にとどまり，これは理論値 345.6 GFLOPS の 28%に相当する．このような状況下で性能を向上させるためには，計算量よりも GPU 内部のデータ転送量を削減することが重要である．たとえば，EFK は計算とともに転送も省けるため，図 7 が示すように，さらなる性能向上に貢献できる．

同一の理由により，実装 D に負荷分散を適用しても（実装 E），その短縮効果は小さかった．実際に，同期なしの実行時間が 0.3 秒ほど短縮されたものの，逆投影における描画時間はほぼ同じであった．提案手法における負荷分散は転送量を削減しないので，性能を向上できなかった．

図 9 に，投影像の数 I，投影像サイズ UV およびボリュームサイズ N を変えたときの実行時間を示す．この図より，I に比例して実行時間が延びている．また，N が再構成処理の応答時間（グラフの y 切片）を支配していて，投影像のサイズ UV がグラフの傾きを決めている．

さらに，$I = 308$ における実行時間の内訳を解析すると（**表 5**），ボリュームサイズ N を

© 2008 Information Processing Society of Japan

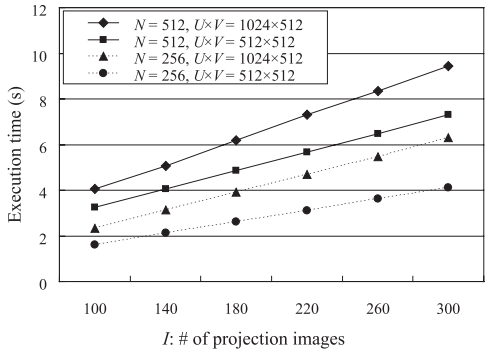


図 9　データサイズごとの実行時間
Fig. 9　Execution time with different data sizes.

表 5　データサイズごとの実行時間と内訳（秒）
Table 5　Execution time and its breakdown with different data sizes (s).

内訳	$U = V = 512$		$U = 1024, V = 512$	
	$N = 256$	$N = 512$	$N = 256$	$N = 512$
T_2：投影像の変換	0.5	0.5	1.8	1.7
T_3：投影像の転送	0.4	0.4	0.8	0.8
T_4：畳み込み積分	2.0	2.0	3.9	3.9
（うち描画）	1.7	1.7	3.4	3.4
T_5：逆投影	1.1	3.7	1.1	3.7
（うち描画）	1.0	3.3	1.0	3.3
T_6：ボリュームの転送	0.06	0.42	0.06	0.42
T_7：ボリュームの変換	0.07	0.58	0.07	0.63
T_1：その他	1.1	1.1	1.2	1.1
T：内訳の合計	5.3	8.8	8.9	12.4
同期なしの時間	4.3	7.5	6.5	9.7

固定して投影像のサイズ UV を 2 倍に増大させても，逆投影の時間はさほど変わらない．この場合，投影像の変換，転送およびそれらの畳み込み積分の実行時間が 2 倍程度に増大している．一方，UV を固定して N を倍増すると，逆投影およびボリュームの転送および変換に要する時間が長くなる．このとき，ボクセルの数が 8 倍になるため，転送時間もほぼ比例して 7 倍ほど増大している．しかし，逆投影は 3 倍程度の増大にとどまっている．したがって，逆投影の時間はボクセルの数というよりはスライスの数に決められていて，より少ないレンダリングパスを実現することが大切である．

表 6　CPU 版に対する画質の比較
Table 6　Reconstruction quality compared to CPU.

指標	データ 1	データ 2	データ 3
誤差 Δ の平均	1.7	0.49	0.11
誤差 Δ の分散	0.16	70.4	0.12
誤差ありボクセルの比率 r（%）	17.5	0.86	10.7
PSNR（dB）	113.1	105.2	161.8

6.3　画質の評価

参考資料として再構成結果に関する画質評価を示す．この結果は，提案手法が医療に使用できることを結論づけるものではないことに注意されたい[⋆1]．図 6 に，CPU 実装との比較を示す．**表 6** に，ボクセル値の誤差 $\Delta = |\,\mathrm{round}(F_1) - \mathrm{round}(F_2)\,|$ を示す．ここで，F_1 および F_2 はそれぞれ CPU 実装および GPU 実装のボクセル値であり，$\mathrm{round}(F_1)$ は F_1 の四捨五入を表す．また，誤差ありボクセルの比率 r は，$\Delta \neq 0$ を満たす誤差ありボクセルの数を N_e として，$r = N_e/N^3$ である．データ 1 において，誤差の平均は 1.7 である．ボクセル値の範囲は 1000〜1800 であるため，約 0.1%ほど値がずれている．残りのデータでは，さらに誤差が小さく良好である．なお，誤差が生じる理由は，CPU および GPU のアーキテクチャが異なるためである．たとえば，GPU は積和計算を 1 つのアセンブリ命令で処理できるため，計算結果が CPU と必ずしも一致しない．

データ 1 は，誤差の平均が他のデータに比べて大きいことに加えて，誤差を含むボクセルの比率が 17.5%に達している．このデータは，再構成領域の大半をアクリル樹脂が占めていて，値が小さい背景（空気）部分はほとんどない．したがって，誤差は背景部分では発生しにくく，値が大きな物質部分において強調される可能性が高い．

また，データ 2 において分散が大きい．誤差を含むボクセルの比率が 1%未満であることから，ごく一部のボクセルが大きな誤差を生じている．このデータは，小さな金属球を撮影したものであるため，金属球の中心を正確に再構成することが求められる．このような特性が他のデータと比較して分散を増大させている．

次に，PSNR（Peak Signal-to-Noise Ratio）値を調べた（表 6）．PSNR は，画質の劣化を表す指標であり，40 dB を超えると，視覚的に劣化を認識することが難しい．すべてのデータにおいて PSNR が 100 dB を超えている．実際に，図 6 に示す可視化結果において

⋆1 ただし，提案手法を採用するコーンビーム CT は医療機器として製品化されている．

© 2008 Information Processing Society of Japan

も明確な差は確認できない．まとめると，GPU 上で動作する提案手法は CPU 実装と比較して誤差を生じるが，視認できるほどではない．

7. おわりに

本稿では，コーンビーム CT を対象として，GPU によりボリューム再構成を高速化する手法を示した．提案手法は，入出力データを RGBA 形式に詰め込むことにより，少ないレンダリングパスを実現し高速処理を実現する．また，計算結果を再利用することにより，さらなる高速化を図る．

実験の結果，512^3 ボクセルのボリュームに対し，CPU 実装と比較して 7.4 秒で再構成できた．この実行時間は既存手法と同程度であるが，高速化に寄与する詰め込み手法に関して，実際のコーンビーム CT に適用できる手法を明らかにしたため，提案手法は有用であると考える．また，逆投影処理の性能ボトルネックがビデオメモリの転送バンド幅にあることを示した．さらなる性能向上には，GPU 内部におけるデータ転送量の削減が有効であり，EFK で実現できる．

今後の課題は，ビデオメモリ容量を超える大規模なボリュームの再構成である．

謝辞　本研究の一部は，科学研究費補助金基盤研究（A）(2)（20240002），若手研究（B）（19700061）および大阪大学グローバル COE プログラム「予測医学基盤」の補助による．本研究の遂行にあたり様々なご支援をいただいた株式会社島津製作所の向田嘉宏氏と三品幸男氏に感謝いたします．また，貴重なご意見をいただいた査読者の方々に感謝いたします．

参考文献

1) Luebke, D. and Humphreys, G.: How GPUs Work, *Computer*, Vol.40, No.2, pp.96–100 (2007).
2) Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E. and Purcell, T.J.: A Survey of General-Purpose Computation on Graphics Hardware, *Computer Graphics Forum*, Vol.26, No.1, pp.80–113 (2007).
3) Xu, F. and Mueller, K.: Real-time 3D computed tomographic reconstruction using commodity graphics hardware, *Physics in Medicine and Biology*, Vol.52, No.12, pp.3405–3419 (2007).
4) Mueller, K., Xu, F. and Nephytou, N.: Why do Commodity Graphics Hardware Boards (GPUs) work so well for acceleration of Computed Tomography?, *Proc. SPIE Medical Imaging 2007* (2007).
5) Mueller, K. and Yagel, R.: Rapid 3-D Cone-Beam Reconstruction with the Simultaneous Algebraic Reconstruction Technique (SART) Using 2-D Texture Mapping Hardware, *IEEE Trans. Medical Imaging*, Vol.19, No.12, pp.1227–1237 (2000).
6) Xu, F. and Mueller, K.: Accelerating Popular Tomographic Reconstruction Algorithms on Commodity PC Graphics Hardware, *IEEE Trans. Nuclear Science*, Vol.52, No.3, pp.654–663 (2005).
7) Vollmar, S., Michel, C., Treffert, J.T., Newport, D.F., Casey, M., Knöss, C., Wienhard, K., Liu, X., Defrise, M. and Heiss, W.D.: HeinzelCluster: accelerated reconstruction for FORE and OSEM3D, *Physics in Medicine and Biology*, Vol.47, No.15, pp.2651–2658 (2002).
8) Shattuck, D.W., Rapela, J., Asma, E., Chatzioannou, A., Qi, J. and Leahy, R.M.: Internet2-based 3D PET image reconstruction using a PC cluster, *Physics in Medicine and Biology*, Vol.47, No.15, pp.2785–2795 (2002).
9) Gac, N., Mancini, S. and Desvignes, M.: Hardware/Software 2D-3D backprojection on a SoPC Platform, *Proc. 21st ACM Symp. Applied Computing* (*SAC'06*), pp.222–228 (2006).
10) Kachelrieß, M., Knaup, M. and Bockenbach, O.: Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware, *Medical Physics*, Vol.34, No.4, pp.1474–1486 (2007).
11) Turbell, H.: Cone-Beam Reconstruction Using Filtered Backprojection, Ph.D. Thesis, Linköpings Universitet, Linköping, Sweden (2001).
12) Feldkamp, L.A., Davis, L.C. and Kress, J.W.: Practical cone-beam algorithm, *J. Optical Society of America*, Vol.1, No.6, pp.612–619 (1984).
13) Cabral, B., Cam, N. and Foran, J.: Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware, *Proc. 1st Symp. Volume Visualization* (*VVS'94*), pp.91–98 (1994).
14) Mueller, K. and Xu, F.: Practical Considerations for GPU-Accelerated CT, *Proc. 3rd IEEE Int'l Symp. Biomedical Imaging* (*ISBI'06*), pp.1184–1187 (2006).
15) Ikeda, T., Ino, F. and Hagihara, K.: A Code Motion Technique for Accelerating General-Purpose Computation on the GPU, *Proc. 20th IEEE Int'l Parallel and Distributed Processing Symp.* (*IPDPS'06*), p.10 pages (CD-ROM) (2006).
16) Schiwietz, T., Bose, S., Maltz, J. and Westermann, R.: A Fast And High-Quality Cone Beam Reconstruction Pipeline Using The GPU, *Proc. SPIE Medical Imaging 2007*, pp.1279–1290 (2007).
17) Riabkov, D., Xue, X., Tubbs, D. and Cheryauka, A.: Accelerated cone-beam backprojection using GPU-CPU hardware, *Proc. 9th Int'l Meeting Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine* (*Fully 3D '07*), pp.68–71 (2007).
18) 李　美花，楊　海圏，小泉和人，工藤博幸：CUDA アーキテクチャを用いた高速コー

© 2008 Information Processing Society of Japan

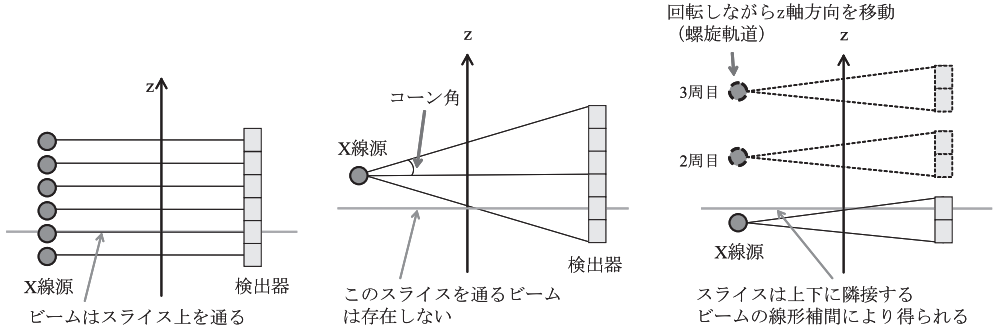
ンビーム CT 画像再構成，*Medical Imaging Technology*, Vol.25, No.4, pp.243–250 (2007).
19) nVIDIA Corporation: CUDA Programming Guide Version 1.1 (2007). http://developer.nvidia.com/cuda/
20) Germain-Renaud, C., Osorio, A. and Texier, R.: Interactive Volume Reconstruction and Measurement on the Grid, *Methods of Information in Medicine*, Vol.44, No.2, pp.227–232 (2005).
21) OpenGL Extension Registry: GL_EXT_framebuffer_object (2006). http://oss.sgi.com/projec ts/ogl-sample/registry/EXT/framebuffer_object.txt
22) Shepp, L.A. and Logan, B.F.: The Fourier Reconstruction of a Head Section, *IEEE Trans. Nuclear Science*, Vol.21, No.3, pp.21–43 (1974).
23) nVIDIA Corporation: NVIDIA GeForce 8800 GPU Architecture Overview, Technical Report TB-02787-001_v01, nVIDIA Corporation (2006). http://www.nvidia.com/object/IO_37100.html
24) Shreiner, D., Woo, M., Neider, J. and Davis, T.: *OpenGL Programming Guide*, 5th edition, Addison-Wesley, Reading, MA (2005).
25) Mark, W.R., Glanville, R.S., Akeley, K. and Kilgard, M.J.: Cg: A system for programming graphics hardware in a C-like language, *ACM Trans. Graphics*, Vol.22, No.3, pp.896–897 (2003).
26) Taguchi, K. and Aradate, H.: Algorithm for image reconstruction in multi-slice helical CT, *Medical Physics*, Vol.25, No.4, pp.550–561 (1998).
27) 田口　彰：マルチスライス CT のシステム構成と画像再構成法，日本放射線技術学会東北部会雑誌，Vol.12, pp.30–34 (2003).

付　　録

A.1　リビニングできないことの補足

Mueller ら[4] は，コーン角（軌道面に対するビームの角度）を 0 度にしたものを並行ビームと呼んでいる．したがって，**図 10** (a) に示すように，彼らの手法は xy 平面に対して並行なファン（扇型）ビームを必要とする．しかし，コーンビーム CT のように円軌道でコーンビームを照射する場合，X 線源に近い領域でリビニングできる情報がほとんどない（図 10 (b)）．したがって，ビームが z 軸方向の厚みを増していくこと（コーン角）を無視してコーンビームで得た投影像をそのまま使わざるをえない．この場合，z 軸方向に軌道面から遠ざかるにつれ，コーンビームが複数のスライスを貫いてしまう．貫くスライスが数枚程度になるようコーン角を狭めて X 線を照射したとしても，ボリューム内の物体に視認できる歪みや線状のアーチファクトが発生する[26]．一般に，コーンビームの特長は 1 回の回転で物体をスキャンできる大きなコーン角にあるため，この近似は再構成原理の点で無理がある．

ただし，Shepp-Logan データのようなファントムを用いる場合，実際の CT 装置で X 線を照射する代わりに，計算機上で順投影をシミュレートして投影像を生成する．このとき，円軌道の直径 d' を物体に対して十分に大きくしておけば，コーンビームを平行なファンビームで近似することは可能である．



(a) 既存手法[4] が必要とする平行ビーム　(b) コーンビーム CT　(c) マルチスライスヘリカル CT

図 10　ビームの体軸（z 軸）方向における断面図
Fig. 10　Cross sectional view of beam along z-axis.

© 2008 Information Processing Society of Japan

一方，マルチスライスヘリカル CT のように z 軸を中心とした螺旋軌道で狭いコーンビームを照射する場合（図 10 (c)），並行ビームへのリビニング手法がいくつか提案されている．たとえば，Taguchi ら[26] らは線形補間に基づく ex-360LI に言及し，その画質を向上させる手法を提案している．4 列程度の狭い検出器を用いても，ex-360LI は大きなアーチファクトを生じているため，512 列を超えるような検出器を貫くコーンビームへの近似は無理である．なお，Feldkamp 法は円軌道のための再構成手法であり，螺旋軌道のための手法ではない[11),27)]．したがって，螺旋軌道の CT に提案手法をそのまま適用することはできない．

(平成 20 年 4 月 25 日受付)
(平成 20 年 8 月 11 日採録)

吉田　征司

平成 18 年関西学院大学理工学部情報科学科卒業．平成 20 年大阪大学大学院情報科学研究科修士課程修了．現在，株式会社日立メディコ勤務．在学中は，GPU を用いた医用画像処理の高速化に関する研究に従事．

伊野　文彦（正会員）

平成 10 年大阪大学基礎工学部情報工学科卒業．平成 12 年同大学院基礎工学研究科修士課程修了．平成 14 年同大学院同研究科博士課程中退．同年同大学助手．現在，同大学准教授．博士（情報科学）．平成 15 年国際会議 HiPC'03 最優秀論文賞，平成 16 年 SACSIS'04 最優秀論文賞受賞．高性能計算に関する研究に従事．

西野　和義

昭和 62 年京都大学理学部卒業．平成 5 年同大学院同研究科博士課程修了．理学博士．同年株式会社島津製作所入社．X 線撮像装置・画像処理システムの開発に従事．

萩原　兼一（正会員）

昭和 49 年大阪大学基礎工学部情報工学科卒業．昭和 54 年同大学院基礎工学研究科博士課程修了．工学博士．同大学助手，講師，助教授を経て，平成 5 年奈良先端科学技術大学院大学教授．平成 6 年より大阪大学教授．平成 4〜5 年文部省在外研究員（米国メリーランド大学）．平成 15 年国際会議 HiPC'03 最優秀論文賞，平成 16 年 SACSIS'04 最優秀論文賞受賞．現在，並列処理の基礎および応用に興味を持っている．

© 2008 Information Processing Society of Japan