Vol. 47 No. 10 情報処理学会論文誌 Oct. 2006

テクニカルノート

GPUを用いた多層ポリゴンモデルの高速ボクセル化手法

原田隆宏† 越塚誠一†

ポリゴンモデルのボクセル化はボリュームグラフィックスの核心技術の1つである．近年，計算機の性能向上にともないボクセルデータの高解像度化が進んでいる．特にGPU（Graphics Processing Unit）の性能の向上は著しく，処理によってはCPUを上回る性能を発揮する．本研究ではGPUを用いたポリゴンモデルの高速なボクセル化手法を提案する．本手法はまずポリゴンモデルから深度剥離によって深度を抽出し，それを用いてボクセル化を行う．本手法は視点方向から見て重なりを持つ多層構造をしたポリゴンモデルのボクセル化も行うことができる．

Full GPU Based Fast Voxelization Technique for Multi Layered Polygonal Models

Takahiro Harada† and Seiichi Koshizuka†

Voxelization of polygonal model is one of the core techniques of the volume graphics. Recently, the resolution of voxel data becomes higher with the progress of computer performance. Especially, the rise of computational power of GPUs is striking. The power of GPUs surpasses CPUs for a specific purpose. In this paper, we present a voxelization technique, which exploit the power of GPU as parallel processing unit. We peel the depths of a polygon model in the first step, then we voxelize the model with the depths. Our technique is able to voxelize multi layered polygonal models.

1. 背　景

ボリュームグラフィックスとは3次元形状を離散的なボクセルデータの集合として扱うものであり，様々な分野で用いられている[10]．たとえば医用分野ではCTやMRIを用いて生体情報をボクセルデータとして取り出す．ほかにも流体力学，衝突判定，3次元空間分析，テクスチャ生成などの分野においてもボクセルデータは利用されている[1),2),5),8),11)]．ボクセルデータはポリゴンモデルから生成することができ，この操作はボクセル化と呼ばれている．近年，計算機の性能の向上によりポリゴンモデルを構成するポリゴン数は増加している．また我々が取り扱うことのできるボクセル数も増加してきている．多くのポリゴンを持つモデルのボクセル化の計算コストは高い．

過去に高速なボクセル化に関する研究が行われている．それらはボクセル化の方法の観点から大まかにスライス法と深度利用法の2種類に分類される．また抽出する対象の観点からは表面と固体形状のボクセル化に分類され，大半の研究は表面のボクセル化であり，固体形状のボクセル化に関する研究は少ない．Dongらはモデルのレンダリング時に各ピクセルの深度をビットで表現し，アルファブレンディングを用いることでモデル表面のボクセル化を行っている[3)]．Fanらは深度剥離を用いてモデル表面のボクセル化を行い，そのボクセルデータを流体計算手法の1つであるLBMでの境界条件として用いている[11)]．これらの手法で抽出した表面ボクセルから内部の詰まった個体形状をボクセル化するには後処理を行う必要がある．この後処理では長い逐次処理が必要であるため計算コストは高い．Karabassiらは固体形状のボクセル化に関する研究を行ったが，前面と背面からの深度のみ取得するため，視線方向からの重なりを持つ形状はボクセル化できない[9)]．高速なボクセル化を行うためにGPUの性能とその並列計算機能に着目した研究もある[3),7),9)]．しかしGPUはCPUによる処理と異なり可能な処理は限られている．Heidelbergerらによる GPUを利用したボクセル化においてもGPUで行うことのできない処理はデータをCPUに読み戻し処理を行っている[7)]．

† 東京大学大学院工学系研究科
School of Engineering, The University of Tokyo

しかし GPU から CPU へのデータ転送はコストが高く，非効率的な処理を行うことになる．

本論文では GPU を用いた高速なボクセル化手法を提案する．本手法はまずポリゴンモデルの深度剥離を行い，ポリゴンモデルを深度情報に変換する．ここで得られた深度は視点からの距離の順に並んでいることに着目し，複雑形状をしたポリゴンモデルを高速にボクセル化する．本手法は処理の間で非効率な GPU から CPU へのデータ転送も行わない．またボクセル化において 3 次元バッファを 2 次元バッファの集合として表現することによって，計算コストの高いボクセル化の処理を並列に処理することが可能となり高速化を行うことができた．なお本手法は個体形状のボクセル化に関する従来の研究[9] においては不可能だった多層構造のポリゴンモデルのボクセル化も可能である．

2. 提案手法

本手法では GPU による高速なラスタライズ機能を利用し，まずポリゴンモデルの深度情報を取得する．そして取得した深度情報をテクスチャとしてフラグメントプログラムに渡すことにより GPU 内部において直接深度情報を参照しボクセル化する．

2.1 深度剥離

ボクセル化の第 1 段階としてシーンの深度の取得を行う．半透明な物体の重なりを正確に描画するための手法である深度剥離手法を応用して各層の深度を取得する[4]．XYZ 方向それぞれ 0 から 1 の範囲をボクセル化する．そのためシーンはそのバウンディングボックス内にスケーリングし描画する．

GPU を用いてシーンを描画する際にピクセルの色のほかに視点からの深度も出力される．デプステストを有効にして描画することにより各ピクセルには視点からポリゴンまで距離が最短のものを描画することができる．このように描画された距離を 1 層目の深度と呼ぶ．次に視点からの距離が次に短いポリゴンの深度を取得する．ここで再度シーンを描画するが，1 層目で取得した深度より深いポリゴンのみを描画することによって 2 層目の深度を取得することができる．各ピクセルにおいてポリゴンの深度と 1 層目で取得したテクスチャに描画された深度を比較し，1 層目より深度が浅いピクセルは破棄することで 2 層目の深度を得ることが可能である．この操作を描画されるピクセルがなくなるまで繰り返すことによりシーンの深度をすべて取得することができる（**図 1**）．

2.2 ボクセル化

第 1 段階で取得したシーンの深度の情報を用いてボクセル化を行う．ボリュームレンダリングのレイキャスティング法では視線に沿いレイを飛ばしサンプリングを行うことでボクセルデータを可視化する．本手法のボクセル化ではこのレイキャスティング法を応用する．バウンディングボックス前面から内部に複数のレイを飛ばし，レイを進める際に深度情報を参照し物体の内外判定を行っていく．

図 1 各層の深度
Fig. 1 Layered depth images. The first (a), second (b) and third (c) depth layers.

取得した深度の各層はそれぞれポリゴンの表面の位置を保持している．さらにそれらは視線方向とポリゴンの法線の方向が一致しているものと反対を向いているものの 2 通りに分類される．視線と法線方向が一致しているものは視点から見て背面のポリゴンであり，反対を向いているものは視点から見て表面のポリゴンである．深度を描画する際，モデルの表面を 1 層ずつ剥離させていくので 1 層目は表面，2 層目は背面となり，1 層目の表面から入射したレイはモデル内に入り 2 層目の背面からモデルを出る．モデルの表面から深度剥離を行うため，得られた深度情報はレイのモデルへの入射位置と離脱位置がそれぞれ対になり，視線からの距離値の順に従い並んでいるため，それらの間を内部と判定しボクセル化する．

次に GPU を用いてこの処理を行いデータを出力することを考える．GPU の仕様上 3 次元バッファは出力することができない．そのため，2 次元バッファを複数枚用いて 3 次元バッファを構築して出力する．この各 2 次元バッファをスライスと呼ぶ．しかし 256^3 のデータをスライスで表現するには 256 枚のスライスが必要となり，多量のピクセル処理が必要となる．ここでバッファには RGBA の 4 チャネルが存在することを用いて，処理しなければならないピクセル数を減らす．スライスのボクセルデータを保持するにはバッファの 4 チャネルを用いる必要はない．そこで 1 チャネルを用いて 1 枚のスライスを表現し，1 枚のバッファに存在する RGBA 4 チャネルのそれぞれに 1 枚のスライスを割り当てる．このようにすることにより 1 枚のバッファを用いて 4 枚のスライスを表現することができる（**図 2**）．この 4 枚のスライスを保持した 1 枚のバッファをパッチと呼ぶ．これによって描画しなければならないバッファの枚数が 1/4 になる．ここで複

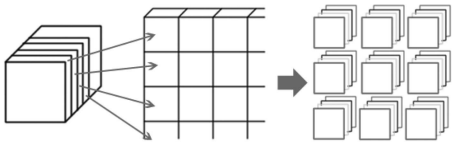


図 2 ボクセルデータ，パッチおよびスライス
Fig. 2 Voxel data, patches and slices. Three dimensional voxel data is represented by sets of patches. The RGBA channels of each patch have four slices.

数回数パッチを描画する代わりにそれらを平面に貼り合わせたバッファを用いて描画を行うことで 1 回の描画ですべてのパッチの処理ができる．GPU は描画される複数のピクセルを並列に処理するので，複数のスライスを 1 枚のバッファで描画することで複数枚のスライスのボクセル化を並列に処理することができる．これはバウンディングボックス内に飛ばすレイを複数のレイに分割し，分割されたレイをいっせいに進めることに相当する．

3. 結　　果

本手法を Pentium4 3.6 GHz の CPU，3.0 GB の RAM，GeForce6800GTO の GPU を搭載した PC に実装した．GPU はコアクロック 400 Hz，ビデオメモリ 256 MB を搭載しインタフェースは PCI Express である．シェーダプログラムは Cg[6] で作成した．

図 3 にボクセル化した結果の一部を示す．これらは上から順に RGBA 各チャネルを示している．**図 4** は 128^3 のボリューム解像度においてモデルをボクセル化した結果を示す．**表 1** に様々なポリゴン数のモデルにおいて本手法を用いてボクセル化を行った計算時間を示す．ボリュームの解像度は 128^3 と 256^3 で行った．深度剥離を行い，その後ボクセル化を行った．ポリゴン数と深度剥離に要する時間は比例している．これは処理を行う頂点数の増加によるものである．なおボリュームの解像度が等しければモデルのポリゴン数が変化してもボクセル化の時間は変わらなかった．これは深度剥離の段階でポリゴンモデルはすべてピクセルデータに変換され，ボクセル化の段階ではポリゴン数に依存しないピクセルデータの処理のみを行うためである．Karabassi らの手法[9] は本研究と同じ固体形状のボクセル化の手法であるが，ボクセル化の処理は CPU を用いて行っている．プロセッサの違いはあるがボクセル化にかかる処理時間は本手法において飛躍的に向上している．また Dong らの手法[3] は表面のボクセル化の手法であるが，論文内で個体形状のボクセル化に関する記述もある．表面のボクセル化にかかる時間は本手法より短いが，このデータの後処理を行い個体形状をボクセル化するとなると深い逐次処理が必要となり非効率的である．

本手法はすべてのポリゴンモデルを正確にボクセル化することができるわけではない．完全に閉じていな

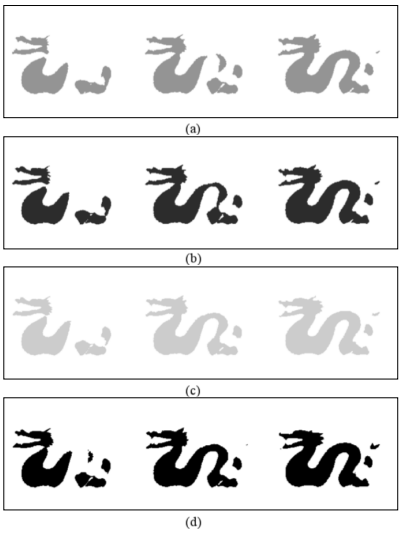


図 3 RGBA チャネルにおける結果
Fig. 3 A part of the result of voxelization in RGBA channel. The result in red channel (a), green channel (b), blue channel (c) and alpha channel (d), respectively.

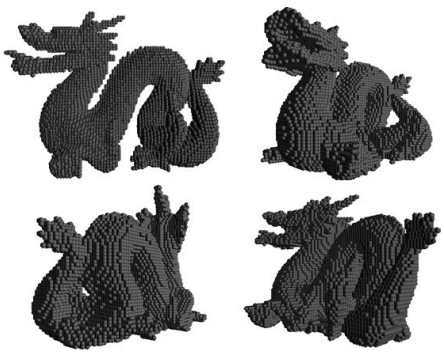


図 4 ボクセル化の結果
Fig. 4 Voxelization result. Voxel resolution was 128^3.

表 1 計算時間（秒）
Table 1 Voxelization time (in seconds).

faces	resolution	depth peeling	voxelization
11,102	128^3	0.0156	0.000388
	256^3	0.0809	0.000449
47,794	128^3	0.0219	0.000388
	256^3	0.0906	0.000449
202,520	128^3	0.0593	0.000388
	256^3	0.127	0.000449
871,414	128^3	0.220	0.000388
	256^3	0.288	0.000449

いポリゴンモデルを入力とすると内外判定を正しく行うことができず正確にボクセル化することはできない．この問題は 1 方向からではなく複数の方向から深度剥離を行うことで改善することができるが，計算速度とのトレードオフとなる[3),9)]．またポリゴンモデルのボクセル解像度よりも細い部分からは内部と判定することのできるボクセルが存在しないため，ボクセルデータを生成することは不可能である．本研究では 256^3 までの解像度でボクセル化を行った．この大きさは 1 チャネルに 1 ボクセルを割り当てる場合の表現できる最大のボクセルであるが，より高解像度のボクセルデータの表現を行うにはそれぞれのボクセルに割り当てるデータをビットで指定することにより可能になる．

4. 結　　論

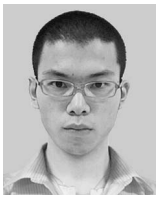
本研究では並列計算機として GPU を用いることによりポリゴンデータからデータ量の多い 3 次元ボクセルデータを高速に生成する手法を開発した．深度剥離を利用し，これにより得られる値は視点からの深度の順に並んでいるという点に着目してデータの GPU から CPU への転送を行わずボクセル化を行った．また 1 ボクセルをピクセルの 1 チャネルで表現し，これらの集合体であるバッファを貼り合わせることで 3 次元バッファを表した．

本手法を計算体系を 1 種のボクセルデータで表現する粒子法の流体計算での初期データの生成や，壁境界の生成に用いることができる．

参 考 文 献

1) Beckhaus, S., Wind, J. and Strothotte, T.: Hardware based voxelization for 3D spatial analysis, *Proc. 5th IASTED International Conference on Computer Graphics and Imaging*, pp.15–20 (2002).
2) Boyles, M. and Fang, S.: Slicing-based volumetric collision detection, *ACM Journal of Graphics Tools*, Vol.4, No.4, pp.23–32 (2000).
3) Dong, Z., Chen, W., Bao, H., Zhang, H. and Peng, Q.: Real time voxelization for complex polygonal models, *Proc. 12th Pacific Conference on Computer Graphics and Applications*, pp.43–50 (2004).
4) Everitt, C.: Interactive order-independent transparency, Technical report, NVIDIA (2001).
5) Fang, S. and Chen, H.: Hardware accelerated voxelization, *Computers and Graphics*, Vol.24, No.3, pp.433–442 (2000).
6) Fernando, R. and Kilgard, M.: *The Cg Tutorial* (2003).
7) Heidelberger, B., Teschner, M. and Gross, M.: Real time volumetric intersections of deforming objects, *Proc. Vision, Modeling, Visualization*, pp.461–468 (2003).
8) Kajiya, J.T. and Kay, T.L.: Rendering fur with three dimensional textures, *Proc.Siggraph*, Vol.23, pp.271–280 (1989).
9) Karabassi, A., Papaioannou, G. and Theoharis, T.: A depth buffer based voxelization algorithm, *Journal of Graphics Tools*, Vol.4, No.4, pp.5–10 (1999).
10) Kaufman, A., Cohen, D. and Yagel, R.: Volume graphics, *IEEE Computer*, Vol.26, No.7, pp.51–64 (1993).
11) Li, W., Fan, Z., Wei, X. and Kaufman, A.: Gpu-based flow simulation with complex boundaries, Technical Report, 031105, SUNY at Stony Brook (2003).

(平成 18 年 3 月 30 日受付)
(平成 18 年 7 月 4 日採録)



原田　隆宏（学生会員）

昭和 56 年生．平成 16 年東京大学大学院工学系研究科システム量子工学専攻修士課程修了．同年東京大学大学院工学系研究科システム量子工学専攻博士課程入学．粒子法の研究に従事．日本機械学会，ACM 各会員．

越塚　誠一

昭和 37 年生．昭和 61 年東京大学大学院工学系研究科原子力工学専攻修士課程修了．同年東京大学工学部助手．平成 3 年博士（工学）．同年講師．平成 5 年助教授．平成 16 年教授．連続体の力学シミュレーションの研究に従事．特に粒子法の開発を行う．平成 17 年に丸善より『粒子法』を出版．平成 18 年に日本学術振興会賞を受賞．日本原子力学会，日本機械学会，日本流体力学会，日本計算工学会，日本応用数理学会各会員．