

GPU を用いた位相情報圧縮のための一手法

金谷 孝之* 手島 裕詞† 西尾 孝治‡ 小堀 研一§
 広島国際大学* 静岡理科大学† 大阪工業大学‡ 大阪工業大学§

1. はじめに

近年3次元グラフィックスハードウェア（以下GPUと略す。）の飛躍的な進歩により、計算速度や計算精度が向上し、さらにCPUと同じように、ユーザによってGPUの機能をプログラム可能となっている。また、GPUには多数の画素に対して同様の処理を行うという性質があるため、GPUは並列処理に適した内部構造を持っている。そこで、GPUをグラフィックス描画以外の演算にも応用しようとするGPGPU（General Purpose GPU）^[1]の研究が活発に行われている。

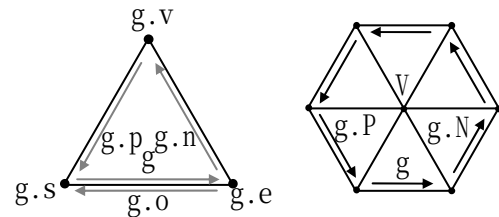
一方、インターネットが日常のインフラになった今日では、3次元モデルをブラウザなどで容易に閲覧できる環境が望まれている。閲覧用のデータは、ポリゴン近似モデルである場合が多い。ただし、ポリゴンモデルである程度の精度を保持するためには、データ量が非常に大きくなるという問題がある。このような場合には、ポリゴンデータも圧縮して転送することが望ましい。ポリゴンモデルのデータには、頂点座標とそれらの連結性が記述されている。したがって、これらの幾何と位相情報をどれだけ少ない情報量で表現できるかが圧縮の鍵となる。位相情報圧縮に着目した代表的な手法の一つにJ. Rossignacによって提案されたEdgebreaker^[2]がある。後述するように、位相情報を五つのシンボルで表現して圧縮する手法である。しかし、Edgebreakerアルゴリズムでは、そのスタート地点によってデータの情報量が異なるため、最適なスタート地点を見つける必要があり、計算コストが大きくなる。

そこで本稿では、GPUの並列性に着目し、Edgebreakerアルゴリズムを、異なるスタート地

点で並列に計算することによって、処理時間を短縮する手法を提案する。

2. Edgebreaker^[2]

Edgebreakerは、三角形ポリゴンを辿る際のスタート地点となる三角形の一边をgate（図中のg）と呼び、gateを底辺とする三角形の頂点Vとの関係を分類して符号化する方法である。それは図1に示すようなHalfEdgeデータ構造と任意の頂点Vに接続した三角形群の外周であるBounding Loopの二つのデータ構造を利用する。



(a) HalfEdge データ (b) Bounding Loop

図1 データ構造

符号は、図2に示すような、C, L, E, R, Sの五つである。Cは、Bounding Loopの頂点Vにすべての面が接続されている場合であり、L, Rは、gateを底辺とする三角形の頂点Vを中心に反時計周りに回ったとき、三角形の欠落が直後にあるか、直前にあるかによって区分される。さらに、分岐においてはS, 終端ではEが用いられる。Edgebreakerアルゴリズムは、HalfEdgeデータ構造とBounding Loopの状態を基にBounding Loopにおける稜線の接続関係を随時更新し、符号化するという手法である。

図3にEdgebreakerの例を示す。図3(a)の点線をスタート地点とした場合、Edgebreakerでは、図中の矢印のように形状を探索することになり、結果として、その符号列 H_a は、 $H_a = \text{CCRRRSLCRSER RELCRRRCRRRE}$ となる。また、スタート地点を変更し、同図(b)の点線にすることによって、符号列 $H_b = \text{CSLLCRLLLRCRRRLLCRSERREE}$ を得る。例えば、Huffman符号化^[3]による H_a と H_b の1文字あたりの平均情報量はそれぞれ1.94と2.15となる。

"A Compression Method for Topology Information using a GPU"

* Takayuki Kanaya, Hiroshima International University

† Yuji Teshima, Shizuoka Institute of Science and Technology

‡ Koji Nishio, Osaka Institute of Technology

§ Kenichi Kobori, Osaka Institute of Technology

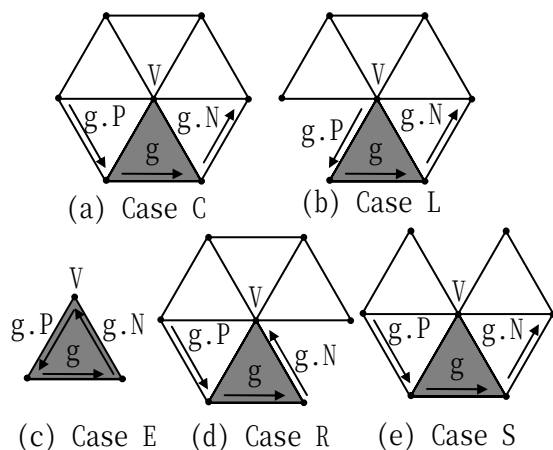


図2 Edgebreaker の符号

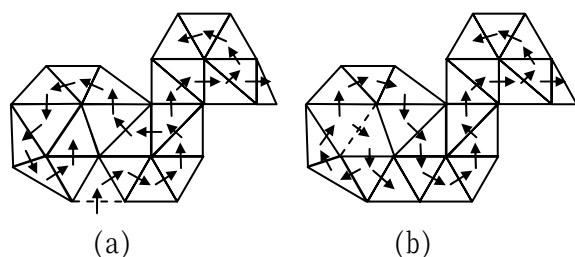


図3 Edgebreaker の例

3. 提案手法

上述したように、Edgebreaker のスタート地点によって、その情報量は異なる。そのため、最適な Edgebreaker のスタート地点を見つける必要がある。最適なスタート地点を見つけるための最も単純な方法としては、すべての候補に対して処理を行い、最適な解を見つけることである。しかし、その処理時間は大きい。そこで、GPU の並列性を用いて、処理時間を短縮する。

提案手法の全体の流れとしては、まず、CPU 側で Edgebreaker 処理に必要なデータをテクスチャデータに適した 2 次元配列として生成する。次に、そのデータをテクスチャとして、GPU のテクスチャメモリに読み込ませ、GPU を用いて、Edgebreaker アルゴリズムを実行する。最後に、各スタート地点の情報量を比較し、その中で最適な情報量である符号列を得る。

1) テクスチャの生成

3 次元の形状データを読み込み、図 1 に示した HalfEdge データ構造を生成する。このデータを 2 次元テクスチャとして GPU のテクスチャメモリに読み込む。さらに、Edgebreaker の結果を出力するためのテクスチャを用意する。この結果用テクスチャを図 4 に示す。各行は異なるスタート地点となる稜線に対応し、各列は、

Edgebreaker の符合とそのとき処理した稜線 gate 番号が書き込まれる。

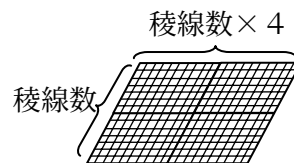


図4 結果用テクスチャデータ

2) GPU での処理

1) で生成したテクスチャデータ群を基に、1 回の描画で一つの gate に対する Edgebreaker 処理を行い、更新された結果を FBO (Framebuffer Object) を用いて結果用テクスチャメモリに出力する。この処理を形状を構成する稜線数回行う。処理 i 回目において、 $i-1$ 回目までの結果用テクスチャで、それまでの状態をトレースする。その後、 i 回目の Edgebreaker 処理を行う。また各ピクセルにおいて、異なるスタート地点を設定して、並列に処理する。

4. おわりに

本稿では、位相情報圧縮アルゴリズムの代表的な手法である Edgebreaker において高速に最適な情報量を得るために、GPU の並列性を利用する手法について述べた。今後の課題としては、効率の良いテクスチャデータの構築が挙げられる。

謝辞

本研究の遂行にあたり多大なるご協力を頂いた大阪工業大学の中村徳裕助手、内之宮仁志氏に深く感謝致します。

参考文献

- [1] M. Pharr 編：GPU Gems 2 日本語版，ボーンデジタル，pp.415-543，2005.
- [2] J. Rossignac: Edgebreaker: Connectivity compression for triangle meshes, GUVU Technical Report, GIT-GUVU-98-35, 1998.
- [3] 藤原洋編：インターネット時代の数学シリーズ 5 マルチメディア情報圧縮，共立出版，pp.29-32，2000.