ポリゴンモデルの圧縮技術

増田 宏

Compression Methods for Polygonal Models

Hiroshi Masuda

Key words: Geometry Compression, Polygon Models, Progressive Mesh, Computer Graphics, CAD, Geometric Model

1. はじめに

インターネットが日常のインフラになった今日では、3 次元モデルを製品形状の定義だけに用いるのではなく、複数の部門や会社間で共有し，ブラウザ等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、必ずしも加工精度に準じた精度は必要ない。多くの場合、ポリゴン近似モデルでも十分である。たたし、ポリゴンモデルである程度の精度を保とうとすると、データ量が非常に大きくなる。自由曲面モデルをポリゴン近似した場合や、3D スキャナを用いてモデル生成を行った場合にその傾向が強い。

そのような場合には、画像や音声を圧縮転送するように、ポリゴンデータも圧縮して転送することが望ましい。ポリゴン圧縮の研究では、1995 年に Deering が最初の論文を発表して以来、2000 年頃をピークとして非常に多くの研究がされてきた。

効率的なデータ圧縮には、メディアの特性を適切に利用することが必要である。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。従って解決すべき問題は、これらの幾何と位相をどれだけ少ないビット数で表現できるかにある。位相の表現では、頂点の連結性を表現するときの冗長性をどれだけ排除できるかが鍵である。一方、頂点座標の圧縮では、できるだけ品質が劣化しないように情報量を効果的に削減することが重要である。近傍の頂点座標から座標を予測したり、周波数領域に変換して高周波領域を減衰させる手法が取られている。多くの場合、位相データの圧縮には損失のない圧縮、座標圧縮には損失のある圧縮が用いられている。

本稿においては、これまで提案されている代表的な圧縮手法について紹介する。例外処理などの詳細なアルゴリズムには立ち入らず、各手法の基本的な考え方を示していく。

2. 位相圧縮

位相圧縮は、頂点の連結性、すなわち位相情報を圧縮するものである。圧縮効率は、１頂点あたり平均何ビットで表現できるかで評価され、研究が進むにつれて次第に効率が高まってきた。ただし、どの手法が有効かは、3D モデルの性質や利用目的にも依存するので、目的に応じた手法を選択することが必要である。

2.1 Generalized Triangle Mesh

三角形メッシュでは頂点は複数の三角形に共有されているので、個別の三角形ごとに座標を転送すると、同じ座標を何回も送らなければならない。OpenGL では、重複を減らすために、三角形ストリップなどの単位でデータ転送する API が用意されている。従来の一般化三角形ストリップでは、座標を転送する際に、三角形の連結情報を三つのコードを用いて 2 bit で記述していた。これらは、頂点列に対して、(R):開始、(O):直前の 2 個を用いて三角形を生成、(M):3 個前の頂点を起点として扇状に三角形を生成、としていた。しかしこの方法を用いても、座標を複数回転送する必要がある。

Deering は従来の一般化三角形ストリップを拡張し、さらに冗長性が削減できる一般化三角形メッシュを提案した [1]。この考え方を Fig.**1** を用いて示す。Deering は、16 個の頂点バッファを用意し、再利用され得るものを明示的に push 記号 (p) をつけて示し、過去の座標を再利用するときには、このバッファの位置をオフセット値で指定した。バッファは 16 個なので、オフセット値は 4 ビットで記述される。図では、オフセットは、マイナス記号をつけて示されている。Deering によれば、この方法によって冗長性の 94 % が削減できたとしている。

この手法は、処理が単純で使用するメモリも限られているのでハードウェア化に向いている。また、この圧縮手法は、JAVA3D の Compressed Object で利用されている [12]。

2.2 Topological Surgery

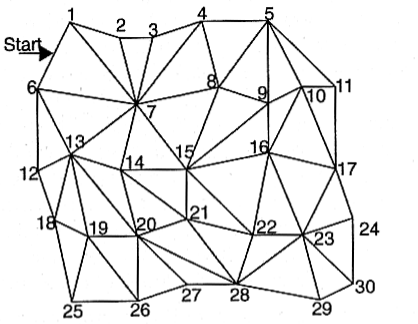
Taubin らの提案した手法 [5] は、Topological Surgery と呼ばれ、VRML や MPEG-4 で用いられている [13]。

この方法では、Fig. **2** (a) のように、メッシュ上に全域木（すべての頂点を通る木）を作る。全域木は、Fig.**2**(b) のように、深さ優先または幅優先の探索によって頂点の順序付けにも使えるので、座標列はこの順番で格納しておく。

次に、全域木に沿って切れ目を入れて、メッシュモデルを展開する。このとき、展開した図形は Fig. **2** (c) のような三角形ストリップの集合になる。この三角形ストリップの外周と図 (b) の全域木との対応を記述しておけば、受け取り側は、全域木を参照しながら三角形ストリップを張り合わせ、元のメッシュモデルを復元することができる。圧縮データとして転送するのは、座標列と位相情報（全域木とストリップ）である。

位相情報は非常に少ないデータ量で記述できる。この例で

＊原稿受付　平成 14 年　月　日
＊＊正 会 員　東京大学大学院工学系研究科（東京都文京区本郷 7-3-1）
1987 年 東京大学大学院工学系研究科修士課程修了．工学博士．同年日本アイ・ビー・エム（株）入社，東京基礎研究所に勤務．1998 年東京大学大学院工学系研究科助教授．1999 年 同 人工物工学研究センター助教授．2003 年より 同 大学院工学系研究科助教授．研究分野：3 次元形状処理，情報視覚化、設計支援システム．



Generalized Triangle Mesh:
R6p, O1, O7p, O2, O3, M4, M8p, O5, O9p, O10, M11,
O17p, M16p, M-3, O15p, O-5, O6, M14p, O13p, M-9,
O12, M18p, M19p, M20p, M-5, O21p, O-7, O22p, O-9,
O23, O-10, O-7, M30, M29, M28, M-1, O-2, M-3,
M27, O26, M-4, O25, O-5

Fig. 1 Generalized Triangle Mesh.

は、全域木は５本の線分列が連結してできている。各線分列は、(頂点数、分岐フラグ、端点フラグ) によって符号化する。分岐フラグと端点フラグは、線分列の始点と終点のそれぞれで辿っていない分岐を持つとき T (true)　となる。この全域木では、■ から始めて、(2,F,F) (1,T,T) (1,F,F) (1,T,T) (6,F,T) と記述される。

また、三角形ストリップは、出発点の三角形からみて、連結する次の三角形が右 (1)、左 (2)、分岐 (3), 終端 (0) であるかを記述する。すなわち、三角形１個につき、２ビットで符号化される。この例題では、破線で示した一番長いストリップの連結情報は、[221222221] の 18 bit で記述される。

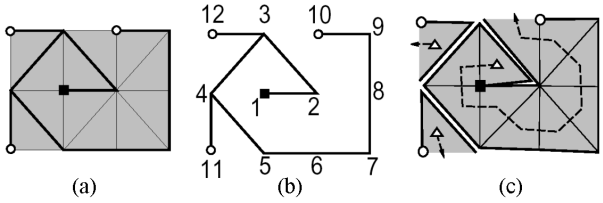


Fig. 2 Topological Surgery.

2.3　Touma らの圧縮法

Touma らは、向き付け可能な境界のない多様体に対して、頂点に接続するエッジを反時計周りに辿れることを利用した符号化手法を示した [6]。また、この手法で Taubin らの手法より、位相データをさらに 1/2 - 1/4 程度にできることを示した。

Fig.**3** を用いて、この手法の概略を示す。まず、最初の頂点 (●) は４個のエッジに接続するので、その次数を [add 4] と記述する。順次、反時計周りにエッジを除去しながら、エッジの端点の次数を [add 4, 6, 5, 7] のように記録していく。また、辿った頂点を頂点リストに格納する。すべての頂点が辿られたら、その頂点を除去する (Fig. **3**(b))。次に頂点リストの先頭の頂点を取り出し、それを基点として同様の処理を行っていく。

処理の過程で、既に頂点リストに格納された頂点に出会った場合には、頂点リストをいくつ遡った頂点であるかを求め、[split N] のように記述する。この場合はループが生成されたことを意味するので、分割された領域に応じて頂点リストを二つに分割し、それぞれ独立に処理を進める。

受け取り側では、頂点の次数の情報を用いて、圧縮したときと同じ順で頂点周りのエッジを再生していく。なお、この手法は境界のない場合を前提にしているため、境界がある場合には仮想的なエッジを生成し、閉じた２多様体にしてから処理を行なう必要がある。

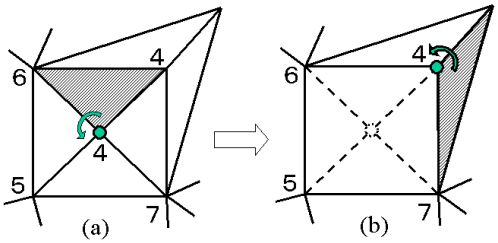


Fig. 3 Touma らの方法.

2.4　EdgeBreaker

Rossignac は、Fig. **4** に示すように、三角形メッシュを辿る際の起点となる三角形の一辺を gate と呼び、gate を底辺する三角形の頂点との関係を分類して符号化する方法を提案した [3]。実装上は、half-edge 構造を用いており、gate は half-edge に相当する。この方法は、Edge Breaker と呼ばれる。

符号は、図に示すような、C, L, E, R, S の五つである。L, R, S は、gate を底辺とする三角形の頂点が、gate を反時計周りに回ったときの直前にあるか、直後にあるか、またはそれ以外かによって区分される。図に、EdgeBreaker による符号化の例を示す。分岐においては S、終端では E が用いられる。

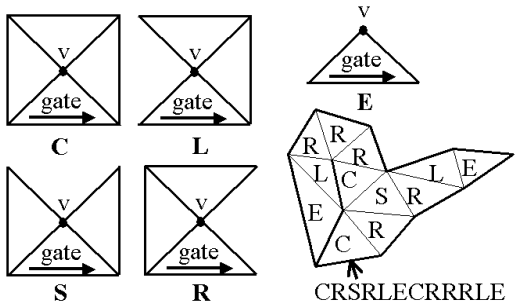


Fig. 4 EdgeBreaker.

2.5　位相的な定義域の拡張

形状圧縮を CAD モデルに適用するときには、必ずしもきれいなポリゴンメッシュにならないことがある。Masuda らは、穴や空洞のあるソリッドモデルの圧縮を行うために、可逆なオイラー操作を用いてメッシュの圧縮問題に帰着させる方法を提案した [10]。この方法では、オイラー操作によってソリッドモデルをサーフェスモデルに変換した後、Fig. **5** のようなオイラー操作によってすべての面を多面体に変換する。受け取り側では、復元したメッシュモデルに逆オイラー操作を施して、元のソリッドモデルを復元する。

多面体モデルの圧縮では、三角形分割を施して、三角形メッシュ圧縮の問題に帰着されるものが多いが、Isenbueg らは、多面体モデルを直接に圧縮できる方法を提案した [7]。。この方法は、Face Fixer と呼ばれ、Edgebreaker で用いられた考え方を多面体に拡張したものである。この方法では、Fig. **4** の分類を多面体に適用できるように拡張し、７通りのコードによって

多面体メッシュを記述している。

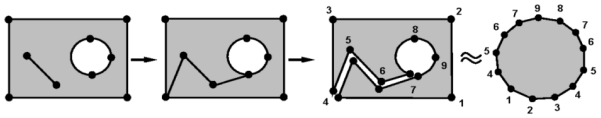


Fig. 5 Euler Operators.

3. 幾何圧縮

次にポリゴンモデルの頂点座標を圧縮する方法を示す。

3.1 量子化

ポリゴンモデルに使われる座標や法線などのデータは浮動小数点 (float:32 ビット) で記述することが多い。しかし、Deering は、この精度が表示目的には著しく過剰であることを指摘した [1]（2^{-46} は、量子レベルの精度である）。ポリゴンモデルは元々近似表現であることを考えれば、この割り切り方は十分妥当な考え方である。

Deering が示した、実用上品質が劣化しない精度は、座標 x, y, z それぞれに対して 16 bit, 法線は 9 bit、また、色は RGB それぞれに対して 5 bit である。法線は、単位球面を 48 区画に分割し、どの区画に入るかで表現することで記述している。いくつかの例題を提示し、これらの精度でも見かけ上、ほとんど区別がつかないことを示している。

3.2 予測符号化

頂点列を転送するとき、x, y, z それぞれに対して、数値列が作られる。このとき、数値の並びの規則性を利用することで必要なデータ量を削減できる。

予測符号化は、すでに転送した過去のデータから、次のデータを予測する方法である。予測値との残差は整数に量子化され、Huffman 符号化などのエントロピー符号化が施される。Huffman 符号化は、データの出現頻度を調べ、出現頻度の多い文字ほど少ないビット数を割り当てる方法である。もし、予測が十分適切なものであれば、残差は小さい値に偏るので、エントロピー符号化が有効に機能する。

予測方法は、メッシュの頂点をどのような順序で並べるかに依存する。Taubin の圧縮手法では、頂点を全域木の深さ優先順に格納するので、頂点は直線状に並ぶ傾向がある。そこで、頂点の予測には線形予測符号化が有効である [5]。線形予測は、頂点列 $\{v_i\}$ $(i = 1, 2, ..)$ に対して、

$$\mathrm{v_i} = \epsilon_\mathrm{i} + \sum_{j=1}^{K} \lambda_\mathrm{j} \mathrm{v_{i-j}}$$

を用いる。K は予測に用いる頂点数、ϵ_i は残差、λ_j は線形予測係数である。係数 λ_j は、残差 ϵ_i の 2 乗和が最小になるように、最小自乗法を用いて算出される。復元の際には、頂点列ごとに　K 個の　λ_j の値を転送する。

Taubin らの示した実験では、ポリゴンモデルの精度が低くてよいときは非常に大きな圧縮率が実現できることを示している。これは、大きな桁ほど規則性があるので圧縮が効き易いが、小さい桁は乱雑性が高く、場合によってはランダムノイズに近くなるため、圧縮が効きにくくなるためである。

一方、Touma らの方法では、頂点周りに一周辿る形で頂点を格納していく。この場合、頂点が一個転送される度に三角形が一個決まる。そこで、辺を共有する二つの三角形を考え、一方の三角形の頂点を他方から予測する。隣接三角形の 4 頂点が平行四辺形の頂点であると仮定し [6]、三角形の頂点が (**u**, **v**, **w**) のとき、隣り合った三角形の頂点を **v** + **u** - **w** で予測する。格納する値は、予測値との残差である。

4. プログレッシブ圧縮

プログレッシブ圧縮転送は、最初に粗いポリゴンデータを転送し、その後に送られるデータによって徐々にポリゴンモデルを詳細化していく方法である。すべてのデータが転送される前に表示ができ、また必要に応じた精度で詳細化ができるという利点がある。

4.1 プログレッシブメッシュ

Hoppe はプログレッシブメッシュ (PM) によってポリゴンモデルのプログレッシブ転送を実現した [2]。この方法では、Fig.**6** に示すような edge collapse という操作でメッシュの個数を削減するポリゴン簡略化を行ない、その履歴を記録しておく。データ転送の際は、まず粗いモデルを送り、続いて edge collapse の逆操作である vertex split 操作を逆順に転送することで、モデルを徐々に詳細化していく。

vertex split を施すには、図に示したように、詳細化すべき頂点 v_s と、split する方向を決める頂点 v_r と v_l、また、新たに生成される 2 頂点の座標の差分ベクトルが必要である。n 個の頂点があるときの一個の頂点の位置指定に log (n) bit 必要なので、全体としては多くのデータ量が必要となる。そのため、プログレッシブ転送に要するデータ量を削減するためには、できるだけまとまった単位で詳細化処理を行なう必要がある。

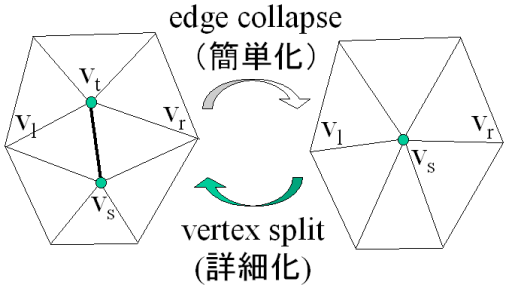


Fig. 6 Progressive Mesh.

Pajarola らは、基本的には PM を用いるが、操作を逐次送るのではなく、グルーピングした上で転送することでデータ量を減らす方法を示した [9]。目安としては、一連の操作群で、頂点が 50 %増加するような単位で転送する。このとき、どの頂点に対して vertex split を施すかを 1 ビットでマークしておき、近傍の頂点は連結情報を使った局所的な index によって記述することで必要なデータ量を減らしている。

4.2 Progressive Forest Split

Taubin らは、Topological Surgery (TS) を利用した Progressive Forest Split (PFS) と呼ばれる手法を示した [4]。基本的な考え方を Fig. **7** に示す。まず、TS と同様に、メッシュ上のエッジからなる木を定義する。このとき、図のような頂点 v, エッジ e, 面 f を記録しておく。次に、図 (b) のように木に沿って切り開き、記録した v, e, f を図のように付け替える。この隙間に対して、図 (c) のような三角形ストリップを埋め込んで詳細化する。ストリップには、位置合わせを行なうために頂点と

エッジをマークしておく。埋め込む三角形ストリップは、TS と同様の方法で符号化できるので、Hoppe の PM に比べて少ないデータ量で符号化ができる。

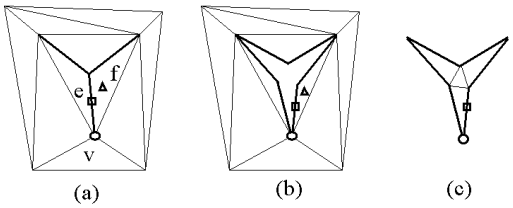


Fig. 7 Progressive Forest Split.

4.3 Alliez らの圧縮法

ポリゴン簡略化には、edge collapse に基づくものの他に、頂点除去に基づくものがある。Alliez らは頂点除去に基づいたプログレッシブ圧縮方法を提案した [11]。

Fig.**8** に頂点除去に基づくポリゴン簡単化を示す。この方法では、まず、矢印で示した gate を指定し、それを含む三角形の頂点周りの面を除去する。頂点周りのエッジが六本なので、六角形の穴ができる。六角形を三角形分割して穴を埋めることで、三角形の個数は 2 個減少する。

Alliez らの圧縮方法は、まず、図の a, b, c の順に gate を移動し、頂点の次数を記録しながら頂点除去操作を行なっていく。次に、穴を三角形分割するが、元のメッシュを復元することを考えると、三角形分割後も多角形の境界が算出できなければならない。そこで、Alliez らは多角形の三角形分割の仕方にルールを導入した。六角形の場合を図に示す。まず、gate となるエッジの始点と終点に + か - のフラグをつける。gate は向き付きのエッジなので、その組み合わせは 4 通りである。そのそれぞれについて、三角形分割の仕方を固定した。頂点除去を行なうのは頂点の次数が 6 以下の場合とし、三角形、四角形、五角形、六角形の場合のそれぞれに 4 通りの分割方法を用意した。復元のときは、エッジのフラグによって元の多角形の境界を知ることができるので、再度、多角形の穴を復元し、中心座標を追加していくことでメッシュを詳細化することができる。

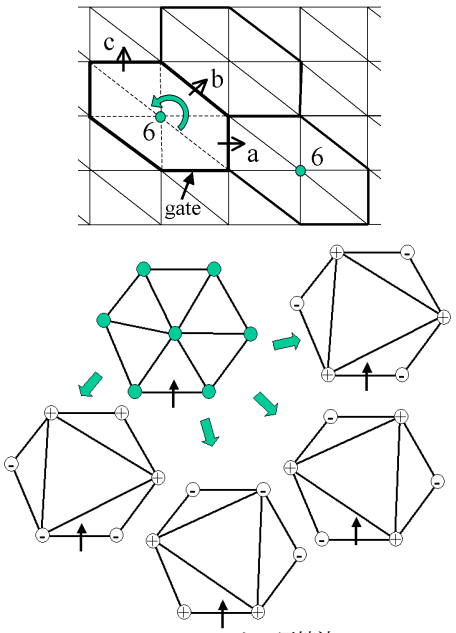


Fig. 8 Alliez らの圧縮法.

5. まとめ

本稿では、ポリゴンモデルの圧縮手法について、位相圧縮、幾何圧縮、また利用形態の一つとしてプログレッシブ圧縮について解説した。さらに詳細な情報を必要とする場合には、参考文献 [14] が詳しいので適宜参照して頂きたい。

1995 年以降の非常に多くの研究の結果、ポリゴン圧縮は既に相当よいレベルに到達し、汎用的な手法に関していえば、既存手法を大きく改善するのは簡単ではなくなった。しかし、ポリゴンモデルの利用方法や表現対象が違えば、観測される規則性も異なるので、別の圧縮法が有利になることがあり得る。また、CAD や CG では、ポリゴンに加えて、パラメトリック曲面や陰関数表現、ボリュームデータなど、多様な表現が存在する。これらの点からも、今後も様々な圧縮手法のバリエーションが研究されると思われる。

参 考 文 献

1) M.Deering. Geometry Compression. SIGGRAPH 95, pp.13-22, 1995.
2) H.Hoppe. Progressive meshes. SIGGRAPH 96, pp.99-108, 1996.
3) J.Rossignac. Edgebreaker: Connectivity Compression for Triangle Meshes. IEEE Transactions on Visualization and Computer Graphics, 5(1):47-61, January-March 1998.
4) G.Taubin, A.Gueziec, W.Horn, and F.Lazarus. Progressive Forest Split Compression. SIGGRAPH 98, pp.123-132, 1998.
5) G.Taubin and J.Rossignac. Geometry Compression through Topological Surgery. ACM Transactions on Graphics, 17(2):84-115, 1998.
6) C.Touma and C.Gotsman. Triangle Mesh Compression. Graphics Interface 98, 1998.
7) M.Isenbueg and J.Snoeyink. Face Fixer: Compressing Polygon Meshes with Properties. SIGGRAPH 2000, pp.263-270, 2000.
8) A.Khodakovsky, P.Schroeder, and W.Sweldens. Progressive Geometry Compression. SIGGRAPH 2000, pp.271-278, 2000.
9) R. Pajarola and J. Rossignac. Compressed Progressive Meshes. IEEE Trans. of Visualization and Computer Graphics, 6(1):79–93, Jan.-March, 2000.
10) H.Masuda and R.Ohbuchi, Coding Topological Structure of 3D CAD Models. Computer-Aided Design, 32(5-6):367-375, 2000.
11) P.Alliez and M. Desbrun. Progressive Compression for Lossless Transmission of Triangle Meshes. SIGGRAPH 2001.
12) JAVA3D API. http://java.sun.com/products/java-media/3D.
13) A. Guiziec, G. Taubin, F. Lazarus, and W. Horn. A Framework for Streaming Geometry in VRML. IEEE CG&A, 19(2), 1999.
14) 3D Geometry Compression. (Organized by G. Taubin and J. Rossignac), Course Notes for SIGGRAPH 2000.