グラフィクスとＣＡＤ 108－1
（２００２．８．８）

細分割曲面のフィッティングによるメッシュデータの圧縮

川原田　寛　　杉原厚吉
東京大学大学院情報理工学系研究科数理情報学専攻

概要

本研究では、オリジナルメッシュに細分割曲面をフィッティングし、細分割する前のメッシュデータを圧縮データとみなす圧縮法を提案する。本手法はメッシュを星状形に分割し、その核内の１点からの半直線がメッシュ表面を一度しか通過しないことを利用して、細分割曲面の制御点とオリジナルメッシュの頂点との対応をとり、最小二乗法によってフィッティングする。この方法で圧縮されたデータは概形を保存しており、解凍における Progressive 性とアニメーション性を確保することができる。

Mesh-Data Compression Using Subdivision Surfaces

Hiroshi Kawaharada　　Kokichi Sugihara
Department of Mathematical Informatics,
Graduate School of Information Science and Technology,
University of Tokyo

Abstract

This paper proposes a new method for mesh data compression. In this method, the original mesh is fitted by a subdivision surface, and the mesh data that generate the subdivision surface is regarded as the compressed data. The volume bounded by the mesh data is first partitioned into star-shaped volumes, and then each volume is compressed. For the compression we establish a nearly one-to-one correspondence between the resulting vertices of subdivision surface and the vertices of the original mesh using rays emanating at a kernel point, and fit the surface by the least square method. The resulting compressed data preserves rough shape of the original mesh, and hence can be used for progressive transmission and has advantages for animation.

1　始めに

CAD や CG の分野では、パラメトリック曲線による形状表現が多く用いられるが、アニメーションを考慮すると従来の形状表現ではパッチ間の接続性を保つのが難しいため複雑なモデルが扱いにくい。よって、それに代わる形状表現の方法として、任意の位相のモデルに適応できる細分割曲面の研究が進んでいる。

また、形状表現の手法として細分割曲面を適用することができるのはメッシュデータであり、大規模なメッシュデータを簡略化、圧縮することは、形状データの配信、web 上での３D コンテンツの提供などの目的から研究されてきた。

2　関連研究

メッシュの簡略化では、Hugues らによる評価関数を用いた方法が有名である([4])。この方法では、オブジェクトをレーザースキャンした密なデータと簡略化したメッシュ間で評価していた。それを発展させ圧縮に用いたのが Hugues の論文([3])で、無歪なメッシュの圧縮方法を示した。

細分割曲面のフィッテイングを行ったものとしては、Tony　Derose らによる[5]がある。ここでは、[4]の方法により簡略化されたメッシュの細分割曲面と、密なスキャンしたデータとの間で評価関数をとるものであった。

また、細分割曲面のフィッティングを応用したものとして Aaron らによる[1]がある。ここでは、フィッティングしたメッシュとオリジナルのメッシュの間の誤差を取り、ディスプレイスドマッピングを使うことで解凍後のメッシュの精度を向上させた。

細分割曲面をアニメーションに用いたものでは、Tony Derose らによる[6]がある。ここでは、[5]のタグ付けされた細分割手法と通常の細分割を合わせた角落ちぐあいを変えられる細分割と、それを利用した布のアニメーションとその際のテクスチャマッピングについて述べられている。

3　本手法の概要

本稿では任意の位相をもつ三角形メッシュデータを星状形に分割し、そのそれぞれについて細分割曲面の

フィッティングを考える。フィッティングであるので、この圧縮は、簡略化とも取れる。またこの圧縮は有歪なものとなっている。

ここで扱う対象は、普通のモデリングソフトで作られたデータ（オリジナルメッシュ）であるとし、[5]などの先行研究で使われていたレーザースキャンしたデータのような密なデータは用いない。もちろん、先行研究でもこれらのデータは、オリジナルメッシュの頂点に置き換えることはできるが、対象が複雑な場合や、元々のメッシュデータが粗い場合に誤差が大きくなることがある。これは誤差を取る対応点を、垂線を下ろした足や最も近い頂点としていたからである。今回は星状形にメッシュを分割することにより、星状形の核を利用した対応点をとることにした。

また、[5]では近似的な極限曲面との誤差を取っていたが、本稿では有限回の細分割曲面とオリジナルメッシュとの間でフィッティングを行う。細分割曲面は極限で滑らかさを保証するが、回数を増やすに伴い多くの面数を必要とする。よって解凍されたデータをリアルタイムでレンダリングすることを想定し、有限回との比較とした。

ここで提案する手法では評価関数を用いない。よって誤差を陽に指定することはできないが、反復のワンステップごとに評価関数を計算する手間を省くことができた。

以下、この手法を説明していく。図１はこのアルゴリズムのフローチャートである。

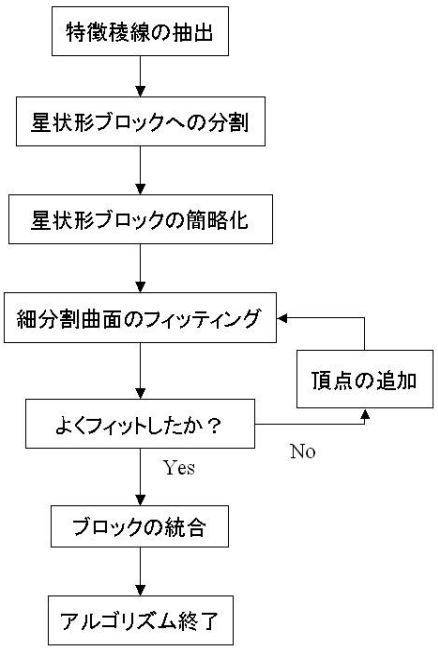


図１．アルゴリズムのフローチャート

3.1 特徴稜線の抽出

圧縮の対象となるオリジナルメッシュデータを M（三角形のメッシュであるとする）とする。M は頂点データ $V(\in \mathbf{R}^3)$ と複体 K (ここでは三次元複体、つまり頂点番号の集合と、２つの頂点番号の組からなる辺の集合と３つの頂点番号の組としての面の集合)から成るものとする。最初に M の特徴稜線集合 C を抽出する。そのために、まず、M に属し隣接する 2 面のなす角度がある閾値 q 以下であるような 2 面の共有する辺を crease edge とし C の要素とする。次に、M のある頂点 v を持つ全ての辺に対して、それらの任意の 2 組の成す角度がある閾値 q' 以下のような頂点 v を corner vertex とし、それを持つ辺を corner edge として C の要素とする。図２は crease edge と corner edge の例を示している。なお、稜線の抽出に関しては[5]において、微分スキームを用いた方法が提案されている。

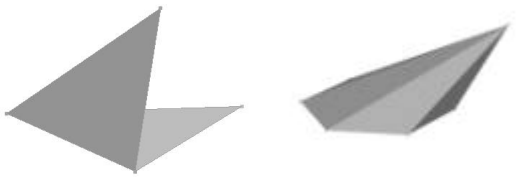


図２． Crease edge と corner edge

3.2 星状形への分割

次にオリジナルメッシュを星状形に分割する。星状形とは、その内部に表面全てを見渡せる点が一点でもあるような図形である。すなわち、図形 S に対して $K(S)=\{x\in S \mid \forall y\in S$ 線分 xy は S に含まれる $\}$（これを核という）とするとき、$K(S)\neq f$ となるならば S は星状形である。$K(S)=S$ の時、S は凸であるので、星状形は必ずしも凸ではない。図３は星状形とそうでない図形の例である。今メッシュで表現される多面体を S と考えれば以下の方法でメッシュを星状形のブロックに分けることができる。

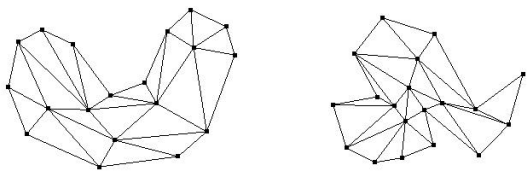


図３．星状形の例

まず M 中の面 f_i の法線を n_i、重心を g_i とし、関数 F_i を $F_i=\mathbf{e}^{n_i\cdot(r-g_i)}$ と定義する。r は位置ベクトルである。今 M の一つの面 f_1 を選び、面集合 f の要素とする。$F_1<1$ となるように r をとる。f_1 の三つ

の辺からなる辺集合を B とする。

次に B の中の 2 辺を持つような面 f_k をとる。取れなければ1つの辺を持つような面を取る。このとき、$F=\sum_{i=1}^{k}F_i$ の微分を取り、最急降下で r を r' まで動かしたとし、そのときの F_i の値が全て1より小さくなっていれば r を更新し、f に f_k を追加する。r が更新されれば、B から選んだ2辺（もしくは1辺）を B から除き、追加した三角形面のその他の1辺（もしくは2辺）を B に加え更新する。この手順で (f,B,r) を更新していく。

r が更新できなければ、それまでの (f,B,r) を一つの星状形のブロックとする。ここで関数 F_i はその面の法線方向に対して単調増加で非負、簡単のため面上で一定の値をとるような関数であればなんでもよい。また閉じた形状が1ブロックとして取られたときは $B=f$ となっている。

B に特徴稜線集合 C の辺が入っている時、その辺を持つような面の追加はしないようにすることもできる。このとき星状形ブロックは、その内部に C の辺は持たない。

このようにしてメッシュ M を星状形のブロック $S=(f,B,r)$ の集合に分解する。ここで r はこの星状形のブロックの核に含まれる点となっている。

3.3 星状形ブロックの簡略化

それぞれの星状形ブロック S に対し、細分割曲面のフィッティングを考えるために、まず S から $S'=(f,B,r,V,K)$ を再構成する。次に、S' に対し legal move [4] を使いメッシュを簡略化する。Legal move とは edge swap 、edge collapse、edge split の3種の操作の内、メッシュの位相構造を変化させない場合の操作のことを言う。図4は legal move の例を示している。

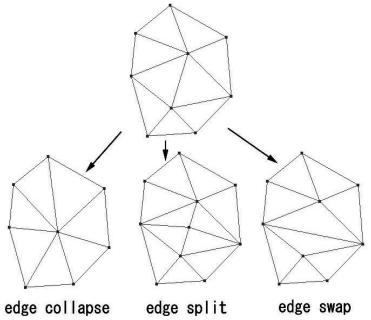


図4．Legal move の例

今、特徴稜線集合 C を考慮して legal move を施す。つまり C に属する辺に関して、edge collapse は行わない。かつ C の辺上の頂点を含む edge collapse はその頂点位置を動かさない。Edge split は行ってもよいが C に分割された辺をそれぞれ追加する。Edge swap は行わない。

特に今回は edge collapse だけを用いた。ここでブロック内の頂点数が目的の細分割回数に適切になるまで減らすことにした。図5は一つのブロックを Legal move で簡略化した例を示している。ここでの簡略化では頂点数のみを考慮しており、特に評価関数を用いて最適な簡略化をしているわけではない。ただし、r が簡略化したメッシュの細分割曲面の内側にくるように簡略化する。今回は判定の容易さのために、r が細分割曲面の核となるように簡略化した。

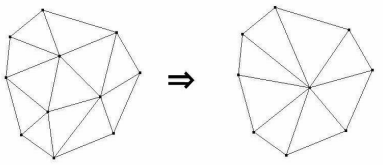


図5．星状形ブロックの簡略化

3.4 細分割曲面のフィッティング

簡略化したメッシュ M と元の星状形ブロック S' のメッシュ M' とで細分割曲面のフィッティングを行う。

まず M に細分割を t 回かけた（$t\geq 1$ だが今回は $t=1$ とした）後のメッシュを M_s とする。適用する細分割は Loop 細分割 [2] とした。このとき C に含まれる辺と、B に含まれる辺は固定した細分割をかける。すなわち Loop 細分割は面を 4 つに分割する自己相似な細分割であるが、その際 C、B の辺に対応する細分割頂点は中点とし、辺上の頂点は動かさないものとする。このような区分的に滑らかな曲面を作る細分割は、[5]に紹介されている。

最小二乗法を用いて 2 つのメッシュ M' と M_s をフィッティングする。このとき、星状形の特徴を用いてお互いのメッシュの頂点同士に対応関係をとり、連立方程式を解く。

r から M_s の各頂点に半直線を伸ばし、その半直線が通過する M' の面で、半直線に最も近い頂点をその M_s の頂点の対応点とする。最も近い頂点が既に他の頂点の対応点として取られていた時は、その面の他の未だ対応点として取られていない頂点の内近い方を取る。3点とも取られているならば最も近い頂点を重複して取る。図6は対応点の取り方を示したものである。

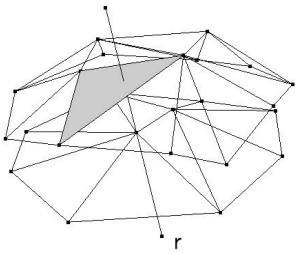


図6．対応点の取り方

今 M' は星状形のブロックで r はその核に含まれる 1 点なので半直線が通過するような M' の面は各々 1 つしかない。ここで問題となるのが M' と M_s の間の領域に r が存在し、通過する面がとれない時があることである。このとき r を 3.2 と同様な最急降下で更新し、r が M_s の内側になるようにする。このような r が取れなければ、そのブロックを２つのブロックに分ける。

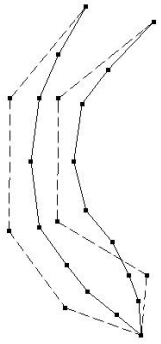
厳密には、よい対応頂点をとるには r が M_s の核でもあった方がよい。今回は星状形分割の方法を、そのまま用いて、M_s の核でもあるようにした。しかし、実際は r が M_s の内側であれば、対応点が取れないことはない。そのとき対応関係が表面上の距離と対応しないような対応頂点も出てくるが、誤差が大きくなるだけで破綻はしない。

M' と M_s の間に対応関係が取れれば、細分割行列 A を使って、

$$V_{M'} = V_{M_s} = AV_M$$

と書けるので、最小二乗法を用いて V_M の頂点位置を求める。ここで $V_{M'}$ は M' の頂点の内 M_s の頂点の対応点として選ばれたものを並べた行列である。

星状形を用いた対応関係では、[5]での最も近い点を取る対応関係で起こる問題を回避できる。メッシュの表面と表面がレーザースキャンのサンプリング幅よりも近いような場合を考えると、図７のようになる。



---: オリジナルメッシュ
——: 細分割曲面

図７．以前の対応決定法の問題点

この場合、最も近い点を取ろうとすると、細分割曲面の点と、向かい側の表面のオリジナルメッシュの点とで対応関係を取ることがある。

また、オリジナルメッシュが自己交差を起こしている場合、最も近い点をとるのでは良い対応関係をとることはできない。

星状形を使った対応決定法では、向かい側の表面メッシュを同じ星状形ブロックとして取ることは無いし、自己交差の起こっている部位も異なる星状形ブロックに別れる。

星状形を使った対応点の取り方を[5]のアルゴリズムに組み込むこともできるが、本稿では有限回の細分割曲面とのフィッティングを考えているので、対応関係に取られたオリジナルメッシュの頂点数をフィッティングの程度として考え、終了条件とする方法を提案する。

3.5 終了条件

今回は反復法を用いているが、[4]や[5]などと違い legal move によって頂点を減らしてはいかず、逆に頂点を追加することを考える。

M は legal move で簡略されているが、M_s の頂点数が M' の頂点数を超えないほどに簡略化している。（ただし、ここでは r が M_s の内側になる範囲で簡略化しているので必ずしも満たされている訳ではない）よって反復では M に頂点を追加していき、適切な簡略化されたメッシュ M にすることがこの手法の目的である。

頂点の追加の判定は、M' のメッシュ上での対応点の取られ方で決める。今 M_s のある面の 3 頂点 v_{s1}, v_{s2}, v_{s3} が M' の頂点 v'_1, v'_2, v'_3 に対応しているとすると、v'_1, v'_2, v'_3 のなす三角形の各辺に M' の複数の辺が対応しているかどうかで、頂点を追加すべきか否かを判断する。どれか１つの辺に３つ以上の辺が対応している場合(1)、３つ全ての辺にそれぞれ２つの辺が対応しており、対応する 2 辺の間の頂点が 3 つ全てで等しくない場合(2)、3 つの辺の内２つに 2 つの辺が対応しており、それらの辺上の頂点が共有されていない場合(3)に、$\{v_{s1}, v_{s2}, v_{s3}\} \in M_s$ を生成する M の面を３つに分割する。図８は頂点の追加判定で追加と判定される(2)の場合の例を示している。今回、分割は重心を用いて行った。

ここで頂点の追加を M' の全ての頂点が対応点として取られるまで行っても良い。この場合、判定のために調べるエッジの数が少なくなり(全ての辺に各々一つの辺が対応しているかどうかを見ればよいので)、計算コストが少なくなるのに加え、テクスチャ座標の移行も考えられるようになる。しかし、今回は追加する頂点数を抑えるためにこのような追加条件とした。この時、対応点に選ばれていない M' の頂点があるが、その頂点は特徴稜線の上には選ばれておらず、その頂点の周りの頂点は対応点に選ばれている（上の(1),(2),(3)の場合に頂点の追加を行えば隣接する頂点が共に対応点として選ばれていないことはない）ので、無視しても概形に影響はないとした（もちろん誤差は大きくなる）。

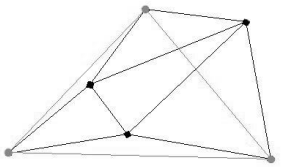


図８．頂点を追加する例

今、頂点の追加は M_s のそれぞれの面ごとに１ステップ中で全て行える。また、Loop 細分割の自己相似により、新たに対応点を求めなおさなければならない頂点は、頂点の追加された M の面によって生成される M_s の４つの面上の頂点であることがわかる。

全ての面において頂点の追加がなくなることを、このアルゴリズムの終了条件とする。

3.6 ブロックの結合

簡略化しフィッティング行った星状形ブロックを集めて結合する。

ここで、各星状形ブロックの境界辺集合 B 上の頂点は、異なるブロックの共有する境界で等しい値を取っている。よって、これらのブロックは分割する前と同じ位相に結合でき、分割前の境界を保存している。

ここで結合したメッシュデータと、細分割をかけて解凍する時にタグ付けしなければならない辺を示した各境界辺集合 B と特徴稜線集合 C を合わせて、圧縮した結果のデータとみなす。

4 適用例

このアルゴリズムを使って圧縮（簡略化）したメッシュデータの例を示す。

図９のような人体のメッシュデータが与えられた時、アニメーションを考慮した稜線として肩の線を C に加えておき、星状形ブロックに分割する際に C の辺を内部に含まないようにすると、図１０のような星状形ブロックが取れた。

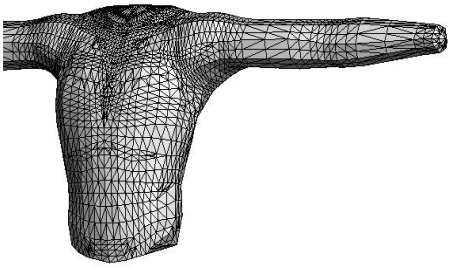


図９．人体のメッシュデータ

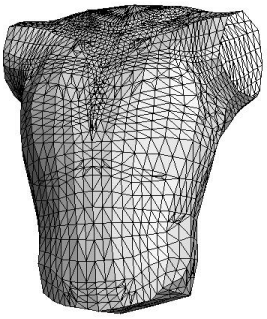


図１０．人体の星状形ブロック

図１０の星状形ブロックに簡略化を施し、細分割曲面をフィッティングしたメッシュデータ、つまり圧縮された星状形ブロックは図１１のようになった。

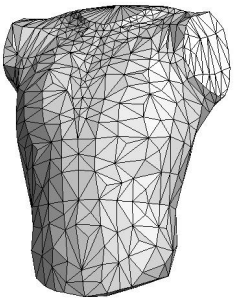


図１１．圧縮された星状形ブロック

解凍されたデータは図１２のようになった。

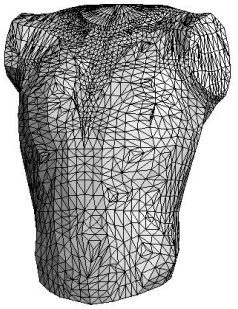


図１２．解凍した星状形ブロック

頂点数はそれぞれ、圧縮前のブロックで 3845 個、圧縮後のブロックで 1369 個であった。このサンプルの場合の圧縮レート（ここでレートは結合したメッシュデータの頂点位置、面情報、各境界辺集合 B、特徴稜線集合 C をバイト換算して比較したものである）は、３７．１%程度であった。

5 結論

本稿で提案した手法を用いれば、任意の位相を持つメッシュデータ（自己交差はあってもよいが、3 つの面以上に共有される辺はないものとしている。もちろんそのような辺を特徴稜線としてやれば対応することはできる）に対して圧縮操作を施すことができる。

また、メッシュの表面同士の距離がレーザースキャンのサンプリング間隔よりも小さいような場合であったり、自己交差があったりするような場合でも、よい対応点をとることができる。

この手法ではフィッティングする細分割曲面の細分割の回数がおおまかな圧縮レートを決める。よって、回数を増やせばレートの向上が期待できるが、反面そのときの誤差は増加していくものと推察できる。実用上では 2 回、もしくは 3 回程度の細分割曲面とフィッティングすれば十分な圧縮が見込めるであろう。

この手法の圧縮結果のメッシュデータは、[5]などと同じくオリジナルメッシュの概形を保存している（簡略化でもある）ので、Progressive JPG のように、圧縮したデータを転送し一旦概形表示してから、解凍しその結果を表示することで精密な画像を得ることができ、web 上での配信に有利である。このような送信の仕方を Progressive transmission という。

また、解凍の細分割は局所的に施こすことができ、選択的なメッシュの精密化としても使える。

また、この星状形を使った対応構成法は、メッシュ表面上で近い位置に対応点をとる。よって、オリジナルメッシュ上の頂点に対して定義できるアニメーション（頂点位置の変化の軌跡）は対応した細分割曲面の上の点に対しても定義でき、さらに細分割の局所性により、細分割の初期メッシュつまり圧縮した結果のメッシュに対しても適用することができる。このとき、圧縮したデータに対してアニメーションを行った結果に、逐次解凍を施せば、オリジナルメッシュと同程度の頂点数でありながら、細分割曲面の滑らかさ（有限回なので厳密に滑らかなわけではないが）を持ったアニメーションを作ることができる。アニメーションを考慮するのであれば、頂点の追加でオリジナルメッシュの全ての頂点が対応点を持つようにしたり、アニメーション上動かない辺を特徴稜線と考えたりと言った工夫もできる。

ここで M' の全ての頂点を対応頂点に取った場合の、フィッティングされたブロックにおいては、M' と M_s にほぼ１対１の対応関係がある。（重複した頂点があるので１対１ではない。）このとき、M_s から M にも対応関係があるので、M' と M の対応を考えることができる。そのとき、星状形での対応の取り方ではメッシュ表面上で近いものに関して対応が取れているので、M' から M へテクスチャ座標の移行を考えることができる。

よって、この手法はアニメーションに有利な圧縮であると言える。

6　今後の課題

今後の課題をあげれば、まず、星状形ブロックを簡略化する際に legal move を用いたが、頂点数しか考慮せずに簡略化してしまった。頂点の追加を抑えられるような簡略化を考えるべきであった。また対応頂点の追加も単純に重心を使ったが、面を３つに分割するのなら重心ではなく、対応頂点が取れるように追加することも出来ると思われる。さらに対応頂点を取るのも、最も近い頂点にしていたが、M_s の１つの頂点が M' の２つの頂点の最も近い頂点となることがある。この時、最も近い頂点同士よりもう一方を選んだ方がよい場合がある。

ここでは頂点追加を面ごとに判定をするように簡単化したため、面を３つに分割することにしたが、それでは圧縮したメッシュの位相構造を固定してしまう。Edge split を併用して追加を行えば、圧縮したメッシュの位相構造に幅がでるだろう。

この手法では r が M_s の内側に取れなければフィッティングできない。r の更新のステップ幅を調節したり、頂点の追加による M_s の変化でフィッティングできないブロックがでたりしないような追加の仕方を考える必要がある。ちなみにフィッティングによって M_s は M' に近づくので、反復の途中で r が M_s の外側に出る可能性は小さい。よって、適用できるよう分けなくてはならないブロックは少ないだろう。

[5]の方法で優れているのは、誤差を指定してやれば同じ曲面に頂点が密に存在しても疎に存在しても、同程度までの簡略化が望める点である。本稿では細分割の回数は 1 回としたが、頂点追加の個数や圧縮結果の誤差（トレードオフだが）を調べるなどして最も良い細分割の回数を見つける必要があるだろう。

参考文献

[1] Aaron Lee, Henry Moreton, Hugues Hoppe: *Displaced Subdivision Surface.* Proc. ACM SIGGRAPH ’00, pp. 85-94, July 2000.

[2] C. T. Loop: *Smooth Subdivision Surfaces Based on Triangles.* Master ’s thesis, University of Utah, Department of Mathematics, 1987. (http://research.microsoft.com/~cloop/thesis.pdf)

[3] H. Hoppe: *Progressive Meshes.* Proc. ACM SIGGRAPH ’96, pp. 99-108, Aug. 1996. (http://www.tecgraf.puc-rio.br/~gattass/cga/pdfs/ProgressiveMesh.pdf)

[4] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W.Stuetzle: *Mesh optimization.* Proc. ACM SIGGRAPH ’93, pp. 19-26, Aug. 1993. (http://www.stat.washington.edu/wxs/Siggraph-93/siggraph93.pdf)

[5] H. Hoppe, T. Derose, T. Duchamp, M. Halstead: *Piecewise Smooth Surface Reconstruction.* Proc. ACM SIGGRAPH ’94, Computer Graphics, Annual Conference Series, pp. 295-302

[6] T. Derose, M. Kass, T. Truong: *Subdivision Surfaces in Character Animation.* Proc. ACM SIGGRAPH ’98, pp. 85-94, July 1998.