スペクトル変換領域で埋め込む 3 次元メッシュの電子透かし
Watermarking 3D Meshes in the Mesh Spectral Domain

向山 明夫[1], 宮澤 貴彦[2], 高橋 成雄[3], 大渕 竜太郎[1]

Akio Mukaiyama[1], Takahiko Miyazawa[2], Shigeo Takahashi[3], and Ryutarou Ohbuchi[1]

k7186@kki.yamanashi.ac.jp, taka-m@nagano.fujitsu.com, takahashis@acm.org, ohbuchi@acm.org

[1] 山梨大学工学部コンピュータ・メディア工学科 Computer Science Department, Yamanashi University
[2](株)富士通長野システムエンジニアリング Fujitsu Nagano Systems Engineering
[3] 東京大学大学院総合文化研究科 Graduate School of Arts and Sciences,University of Tokyo

1. はじめに

データ埋め込み, または電子透かしと呼ばれる技術は, 情報を表現する watermark(透かし)と呼ばれる構造体を, 透かしの埋め込み対象となるコンテンツ(文字文書, 静止画像, 動画像, 音声データ, 3 次元モデル, 等)に付加する(詳しくは[松井 98, Katzenbeisser00] 等を参照). ここで, 透かしの存在が埋め込み対象コンテンツの本来の目的(例えば人による表示・鑑賞)を阻害しないこと, かつ, 透かしがコンテンツから容易には除去できないこと, が要件である. 埋め込まれた透かしは, 改ざんの検出, 正規の購入者の認証など, そのコンテンツを何らかの形で管理する目的で用いられることが多い.

VRML や MPEG4, 3 次元形状 CAD データ等の 3 次元(3D)データ, 特にその形状を対象とする透かし手法の多くは 3D ポリゴンメッシュの形状を対象としている([Kanai98, Benedens99, Yeo99, 大渕 01]など). このなかで, 大渕らが先に提案した 3D メッシュの形状を対象とする透かし [大渕 01]は, そのメッシュの変換領域で透かしを付加する手法である (図 1). 用いる変換は, ポリゴンメッシュの頂点の接続関係から定義されるメッシュラプラシアン行列を元にした, ポリゴンメッシュ形状のスペクトル分解 [Karni00] である. この透かし手法は, 相似変換(回転, 並行移動, 一様スケーリングの組み合わせ), 頂点座標へのランダムノイズ重畳, 形状のスムージング[Taubin95] , メッシュの部分的切り取りに対する耐性を持った.

この手法に残された課題の中で最も重要と考えられる 2 つは (1)メッシュ簡単化に対する耐性付与, および (2) 処理時間の短縮, である. この手法で埋め込まれた透かしはメッシュ簡単化操作に対する耐性を備えていない. 接続性を変更するメッシュ簡単化処理がスペクトル分解の基となるラプラシアン行列を変えてしまうためである. また, スペクトル分解の処理時間はメッシュの頂点数のほぼ 3 乗に比例して増えるため, 頂点数が 1000 程度を超えるメッシュでは透かし処理時間が実用範囲を超える. 先行研究 [大渕 01] では処理時間を減らすためメッシュを領域に分割する手法を導入したが, 領域の大きさがばらつくために固有値分解の手間があまり減らず, また領域の配置を人手で決めるためその試行錯誤に手間がかかった.

本論文では, これら 2 つの課題を解決する手法を提案する. まず, メッシュ簡単化に対する耐性を付与するためには形状のリサンプリングを用いる. 形状のリサンプリングを用いて被覆メッシュの接続性と透かしメッシュの幾何形状を持つメッシュを作り出す. これにより, 簡単化により接続性が変わり, かつ剛体変換の加わった透かしメッシュから透かしを取り出すことができる.

処理時間を減らすには, 領域分割手法の改善で対応する. 領域分割に要する人の関与を減らすため, 領域の中心の指定を自動化した. また, 領域の間に隙間を許す疎分割(図 7 参照)を導入し領域数と領域サイズのばらつきを減らすことで, 領域分割とその後の固有値分解に必要な計算機時間を減らす.

本論文の構成は以下のとおりである. 第 2 章では, 基本となる透かし処理, メッシュ簡単化に対する耐性強化法, 改善された領域分割手法について述べる. そして, 第 3 章では実装と実験の結果を紹介し, 第 4 章でまとめと今後の課題について述べる.

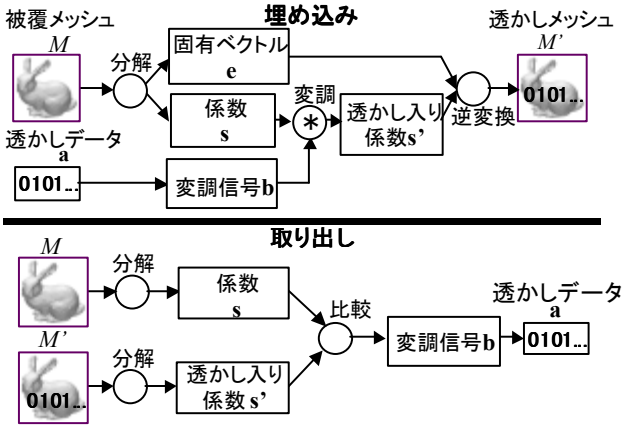


図 1. スペクトル分解を用いた電子透かし処理の基本的流れ.

2. スペクトル分解による電子透かし

2.1. 透かし処理の基本アルゴリズム

大渕らの手法[大渕 01]では, メッシュをスペクトル分解して得られるスペクトル係数の振幅を透かし情報に応じて変更し, 透かしを埋め込む. 透かしの入った形状(透かしメッシュ)は, 固有ベクトルと変更したスペクトル係数の一次結合で求まる. 透かしの取り出しの際には, 透かしを付加したメッシュ(透かしメッシュ)と透かし付加前のメッシュ(被覆メッシュ)の双方を使用する(秘密透かし). 透かしの取り出しは, 透かしメッシュと被覆メッシュを共にスペクトル分解し, そのスペクトル係数を比較して行う.

メッシュスペクトルは, 頂点の接続関係で定義された

メッシュラプラシアン行列に対し固有値分解を施し，得られた固有ベクトルに頂点座標を射影して得られたスペクトル係数からなる．メッシュラプラシアン行列にはいくつかの定義があるが，その中で，我々は[Bollobás98]のメッシュラプラシアン行列 **K** (別名 Kirchhoff (キルヒホフ)行列)を用いた．

$$\mathbf{K} = \mathbf{D} - \mathbf{A}. \tag{1}$$

A はポリゴンメッシュの頂点の隣接行列で，

$$A_{ij} = \begin{cases} 1 & \text{頂点}i\text{と}j\text{が隣接}, \\ 0 & \text{その他}, \end{cases} \tag{2}$$

のように定義される．**D** は対角行列で，その対角要素 $D_{ii} = d_i$ は頂点 i の次数である．

スペクトル分解の結果は，およそ，絶対値の小さな固有値に対応する固有ベクトルがメッシュの概形を，大きな固有値に対応する固有ベクトルがメッシュの詳細を，それぞれ表現する．前節で述べたように，この手法で埋め込まれた透かしはメッシュ簡単化操作に対する耐性を備えていない．接続性を変更するメッシュ簡単化処理がスペクトル分解の基となるメッシュラプラシアン行列そのものを変えてしまうためである．

スペクトル係数の値を変更して透かしデータ **a** (図 1 上)を埋め込む際，埋め込むデータビット列の j 番目のビット a_j (値は{0,1}を取る)は拡散率 c 回複製され，変調信号 **b** に変換される．

$$b_i = a_j, \quad j \cdot c \le i < (j+1) \cdot c. \tag{3}$$

同じデータを繰り返し埋め込むことで，ランダムノイズ重畳などに対する耐性を得る．次いで，b_i は{0, 1}から{-1, 1}の値をとる透かしシンボル b_i' に変換される．変更されたスペクトル係数 s_j' (x, y, z 成分がある)は次のように計算する．

$$s_i' = s_i + b_i' \cdot p_i \cdot \alpha, \tag{4}$$

s_i : 変更する前のスペクトル係数,
p_i : あるシード値から生成された{-1, 1}の値をとる擬似乱数列,
α : 変調振幅.

ここで，変調振幅 α は，被覆メッシュを囲う最小の直方体である axis-aligned Bounding Box (BBox)を計算し，その辺の長さの最大長 ϕ に対する割合（振幅率）β を指定することで $\alpha = \phi \cdot \beta$ のように求める．ここで用いた，埋め込むビット列でスペクトル係数列を変調する手法は，Hartungらが MPEG4 SNHC 顔アニメーションパラメタ数値列の変調に用いたものと類似する[Hartung98]．

本透かしは秘密透かしであり，透かしの取り出しの際，透かしメッシュと被覆メッシュとを比較して透かしを取り出す(図 1 下)．取り出し処理(詳細は[大渕 01]を参照)では，まず透かしメッシュに相似変換がかけられたと仮定し，透かしメッシュと被覆メッシュから得られる固有ベクトルを使って 2 つのメッシュの位置合わせを行う．位置合わせが終わると，埋め込みの式(3)(4)で使用した，拡散率 c ，被覆メッシュから計算した参照となるスペクトル係数 s_i ，埋め込み時と同じシードから生成した同一の擬似乱数系列 p_i ，を使用して q_j を求める．スペクトル係数には x, y, z に対応する 3 つの成分があるので，以下の式のように 3 つの成分の寄与を平均する．

$$q_j = \frac{1}{3}\sum_{i=j\cdot c}^{(j+1)\cdot c-1}(s_i' - s_i)\cdot p_i = \frac{1}{3}\sum_{i=j\cdot c}^{(j+1)\cdot c-1} b_i' \cdot \alpha \cdot p_i^2, \tag{5}$$

s_i' : 透かしを埋め込んだモデルから計算されたスペクトル係数.

ランダムノイズなどの妨害を無視できるとすると，q_j は，式(6)のようになる．

$$q_j = c \cdot \alpha \cdot b_i' \tag{6}$$

式(6)より，拡散率 c と振幅 α は常に正の数であるから，q_j の正負を判定することによって透かしデータ a_j が取り出せる．

$$a_j = sign(q_j) \tag{7}$$

スペクトル分解に必要な固有値分解の時間は，メッシュの頂点数 n に対して約 $O(n^3)$ で増える．実験によると，例えば 2218 頂点のメッシュを固有値分解するには3時間近い時間がかかった(表 1)．そこで，先行研究では，頂点数が 1000 程度以上のメッシュに透かしを埋め込む場合はメッシュを頂点数数百程度以下のメッシュからなる領域に分割して処理した [大渕 01]．

表 1. 固有値分解の計算時間.

モデル	頂点数	面数	時間
tiger	254	504	20s
distcap	686	1368	6m19s
bunny1	1197	2390	33m16s
bunny2	2218	4432	2h54m40s

分割されたそれぞれの領域に対して個別にスペクトル分解を施し透かしを埋め込むことで，全体の処理時間を減らせる．また，同じ透かしを各領域に繰り返し埋め込めば，メッシュの切り取りに対する耐性も付与できる．領域ごとに同じ内容の透かし情報を埋め込むため，透かしを埋め込んだ領域がどれか一つでも切り取られずに残っている場合，透かしが取り出せるためである．透かしの取り出しではその埋め込みと同一の領域分割が必要となるが，これは被覆メッシュから生成できる．（本透かしは取り出しに被覆メッシュを用いる秘密透かしである．）

領域分割は，与えられたメッシュの頂点群から選んだ頂点（特徴点）を出発点とし，隣接する頂点へと領域を拡張していく．この拡張操作を，領域に含まれる頂点数がメッシュ上のすべての頂点を覆い尽くすまで行う．ここで，特徴点は，各領域に埋め込む情報量を最大にするため，全領域の頂点数がほぼ同じになるように選びたい．これにはメッシュ上で特徴点が等間隔になるように特徴点を選べばよい．先行論文[大渕 01]では，特徴点を手動で選んでいた．手動で特徴点を指定すると，領域分割に形状特徴をある程度反映できるが，その指定に手間がかかる．特に，多数の領域に含まれる頂点数がほぼ等しくなるように特徴点を配置するのは困難であった．

2.2. メッシュ簡単化に対する耐性の強化

メッシュ簡単化は近年広く用いられる様になってきた．この操作を加えられると，先行研究の透かしは壊れた．これは，本手法の透かしの基本となるメッシュラプラシアン行列がメッシュの頂点接続性のみによって

定まるため，形状を保存しつつ頂点数を減らすメッシュ簡単化処理([金井 00]を参考) を受けると透かしが取り出せなくなるためである．

メッシュ簡単化に対する耐性を付与するため，我々は形状リサンプリングを用いた [宮澤 01] . 形状リサンプリングとは，透かし付加(と簡単化)後のメッシュ M' から，M' の幾何形状と透かし付加前のメッシュ M の頂点接続性とを持つ新たなメッシュ M'' を生成する処理である．この M'' はメッシュラプラシアン行列が元メッシュ M と同じで，かつ，幾何学的形状が M' とほぼ同じため，透かしが取り出せる．リサンプリングの手順としては，メッシュ簡単化に加えて幾何変換が加えられたと仮定し，まずメッシュ同士の幾何的な位置・向き合わせを行う．本研究では，幾何変換として剛体変換を仮定し，これに対する位置合わせを実現した．位置合わせが済めば，後は光線追跡による幾何形状のリサンプリングを実行すればよい．

2.2.1. 頂点接続性が異なるメッシュ間の位置合わせ

先行研究では接続性が変わらないと仮定し，2.1 節で述べたメッシュラプラシアン行列の固有ベクトルによる位置合わせを使った[大渕 01]．(本論文では，被覆，透かしの 2 つのメッシュの向きと重心を合わせることを位置合わせと呼ぶ．)しかし，メッシュ簡単化を仮定した場合にはこの手法は使えない．

位置合わせにはメッシュの頂点を質点として慣性主軸を求めるGottschalkらの手法[Gottschalk96]があるが，メッシュ簡単化は頂点の分布を変えるため，この手法はうまく行かない．我々は，頂点ではなくメッシュの面内に質量が連続的に分布するとし，これをモンテカルロ法で近似し，その慣性主軸を求めることで向き合わせを行うこととした．具体的にはメッシュのそれぞれの面上にランダムに質点を配置し，その質点にGottschalk らの手法[Gottschalk96]を当てはめる．質点の個数は面の面積に比例して決める．モデルの重心位置を合わせるには，このようにして配置した質点群の重心を求めればよい．

面上にランダムな点 $\mathbf{P}$ を発生させるには，三角形の 3 頂点の座標 $\mathbf{t}_1$，$\mathbf{t}_2$，$\mathbf{t}_3$ より次式で求める[Osada01] . ここで r_1，r_2 は，定義域[0, 1]の乱数値である．

$$\mathbf{P} = \left(1-\sqrt{r_1}\right)\mathbf{t}_1 + \sqrt{r_1}\left(1-r_2\right)\mathbf{t}_2 + \sqrt{r_1}r_2\mathbf{t}_3 . \qquad (8)$$

本手法の位置合わせ精度を確かめるために行った実験結果の一つを，向き合わせの場合について示したのが図 2 である．図 2 はモデル bunny1(図 3(a)，頂点数 1197，面数 2390)に簡単化を行ったメッシュ(相似変換など，他の妨害はいっさい加えてない)と簡単化を行う前のメッシュとで主軸の誤差を調べた結果である．簡単化を行い，頂点を 400 取り除くまで実験を行った(頂点を 400 取り除いたメッシュが図 3(b))．ここで言う誤差とは，簡単化前と後の 3 つの主軸(第 1 主軸，第 2 主軸，第 3 主軸がある)の差(単位は度)を平均したものである．図 2 からわかるように，簡単化が行われた場合，提案手法の向き合わせ精度は Gottschalk らの手法によるものよりも良い．

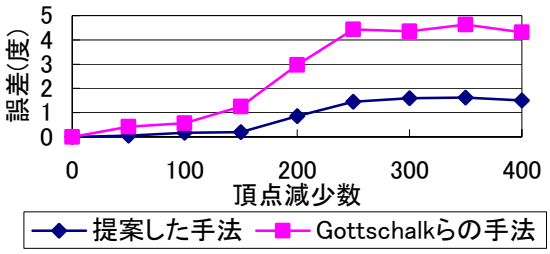
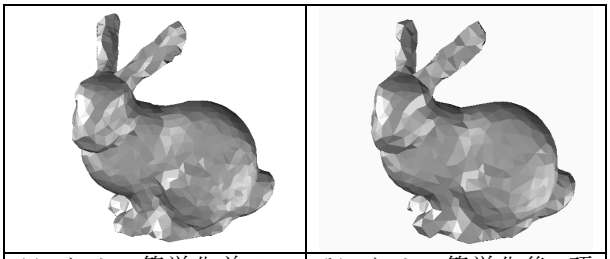


図 2. 主軸の誤差.



(a) メッシュ簡単化前のモデル(頂点数 1197，面数 2390).

(b) メッシュ簡単化後．頂点を 400 除去したモデル．(頂点数 797)

図 3. 誤差実験に使用したモデル.

2.2.2. 形状のリサンプリング

位置合わせがすむと，以下のようにして光線追跡により幾何形状のリサンプリングを行う．

1) 被覆メッシュ M と全く同じ頂点の座標値と接続性を持ったメッシュ M_c を作成する．
2) M_c の頂点 $\mathbf{v}_i$ $(i = 1,2,\ldots,n)$ に接する面の法線ベクトルの平均を，$\mathbf{v}_i$ の法線ベクトル $\mathbf{N}_i$ とする．
3) $\mathbf{v}_i$ を通り，$\mathbf{N}_i$ を方向ベクトルとする直線 l_i を考え，透かしメッシュ M' の表面と l_i との交点を求める(図 4)．
4) 先の求めた交点のうち，幾何的な距離が $\mathbf{v}_i$ に最も近い点の座標で $\mathbf{v}_i$ の座標を置き換える．

上述した方法でリサンプリングを行うとき，問題が 2 点ある．第 1 点は，被覆メッシュの頂点 $\mathbf{v}_i$ に対するリサンプル点 $\mathbf{v}_i'$ が求まらない場合(例えば，図 5 頂点 **a** の法線方向には透かしメッシュがなく，**a** からリサンプル点が求まらない)である．この場合，$\mathbf{v}_i'$ の座標値を $\mathbf{v}_i$ の座標値と同じにする．第 2 点は，$\mathbf{v}_i'$ が適切に求まらない場合(例えば，図 5 頂点 **b** に対する理想的なリサンプル点は，形状が類似した頂点 **b**′ であるが，実際は **b** の法線方向に **b**′ はなく，**b** とは形状の対応しない点 **c** がリサンプル点となってしまう)である．この場合，$\mathbf{v}_i$ と $\mathbf{v}_i'$ との幾何的距離が，あるしきい値を超えたとき，$\mathbf{v}_i'$ の座標値を被覆メッシュの頂点 $\mathbf{v}_i$ の座標値に変更する．

リサンプリング手法の計算時間は，被覆メッシュの頂点数 n と透かしメッシュの面数 m とに依存し，$O(n \times m)$ で処理時間が増加する．そこで，我々は，空間の等方分割を用いた光線・物体交差判定高速化技術[GraphicsGems94]を応用し，ある法線と交差判定を必要とする面数を刈り込むことでリサンプリング処理を高速化した．具体的には，まず，被覆メッシュ M と透かしメッシュ M' とが存在する空間を 3 次元正方格子状に分割する(図 6 に 2 次元で図示)．次いで，M の頂点の法線が触れる格子(複数)に所属する面(複数)と法線と

の間でのみ詳細な交差判定を行う.

上述した手法で交差判定を行う面数を刈り込んだ場合と, 刈り込みを行わず全ての面と交差判定を行った場合との処理時間を調べた結果を表 2 に示す. ここでは, 同じメッシュを被覆メッシュと透かしメッシュとして扱い, 時間を測定した. 表からわかるように, 交差判定の面数を刈り込むことによって処理時間が大幅に短縮された.

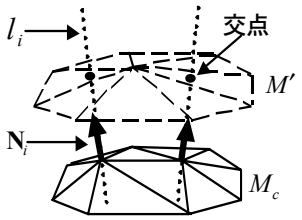


図 4. 光線追跡により幾何形状をリサンプリングする.

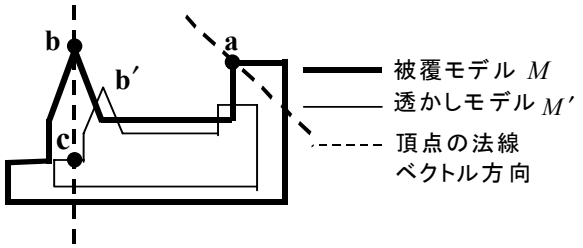


図 5. リサンプリングの失敗の例(2 次元で図示).

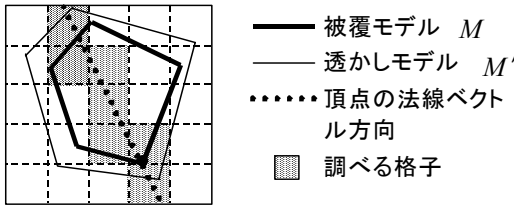


図 6. 計算する面数を減らして, 計算時間を短縮(2 次元で図示).

表 2. リサンプリング処理時間の例.

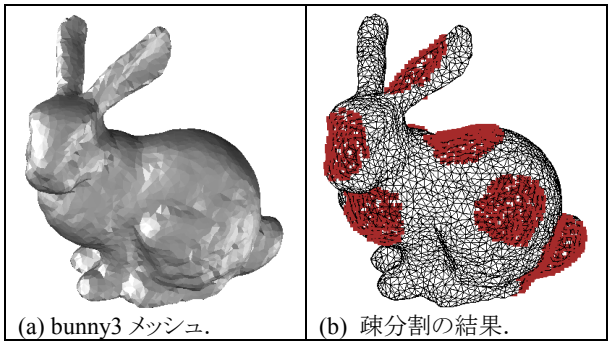
モデル	頂点数	面数	処理時間	
			刈込みなし	刈込みあり
bunny2	2218	4432	22s	0s
bunny4	13990	27976	15m04s	7s
bunny5	25849	51694	44m57s	20s

2.3. 領域分割手法の改善

先行研究の領域分割の問題点は, 3 点あった. 第 1 に, 領域の大きさのより均等な良い分割が得られるように特徴点を選ぶ作業は試行錯誤が必要で, 人の手間がかかること. 第 2 に, 特に頂点数の多いメッシュで領域数が多くなり, 固有値分解の処理時間が(領域数に線形な増加ではあるが)大きくなること. 第3に, 領域サイズにばらつきが大きいため, 小さめの領域のサイズを固定すると大きめの領域の固有値分解処理時間が大きくなり, 全体として透かしの処理時間が増大すること, である.

第 2, 第 3 の問題点を解決するため, 我々は領域の疎分割を導入する. 先行論文[大渕 01]で提案した手法では, メッシュの全ての頂点が必ずどれかの領域に属する密分割を用いていた. それに対し, 疎分割の場合, どの領域にも属さない頂点が存在する (図 7(b)). Karni らのメッシュ圧縮の場合は全ての頂点を含むように分割する必要があった[Karni00]が, 透かしの場合はその必要がない. メッシュ切り取りに対する透かしの耐性低下が許容できる程度の密度で透かし領域が配置されていれば十分である.

疎分割を行うには, 与えられたメッシュの頂点群から選んだ頂点(特徴点)を出発点として, 隣接する頂点へと領域を拡張していく. この領域の拡張操作を, 領域に含まれる頂点数があるしきい値を超えるまで行う. 疎分割により領域数が減り, かつ, 領域の大きさが均等となる. 領域の大きさが均等になるのは, 分割処理時の領域間の競合が殆ど無いためである. これらの結果, 疎分割の導入により, 領域分割と固有値分解の双方の処理時間が短縮できる.



(a) bunny3 メッシュ. (b) 疎分割の結果.

図 7. メッシュ bunny3 (頂点数 4114, 面数 8224)を 10 の小領域(塗りつぶした部分)に疎分割した例.

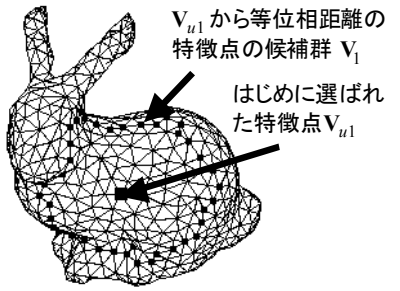


図 8. 正三角形の充填による特徴点の自動探索.

領域分割における第 1 の問題点である, 人が特徴点を指定する手間を減らすため, 我々は(半)自動的に特徴点を決定する手法を導入した[宮澤 01]. 平面に正三角形を充填すると, その三角形の頂点を中心に面積が互いに等しいボロノイ領域が作られることに注目した. 我々はこれを 3 次元メッシュに近似的に当てはめ, 与えられたメッシュの表面に擬似的に正三角形を充填することで領域分割を行うアルゴリズムを考案した. 以下にそのアルゴリズムを示す. ユーザが 2 つの特徴点(初期頂点 $\mathbf{v}_{u1}$ と, 集合 V_1 の中の 1 点)を選ぶと残りの特徴点が自動的に決まる.

1) ユーザに, 特徴点の個数 F (分割する領域数)と, 最初の特徴点 $\mathbf{v}_{u1}$ を指定させる.
2) $\mathbf{v}_{u1}$ から位相距離 R にある頂点の集合 V_1 を求める (図 8).
3) V_1 から 1 つの頂点 $\mathbf{v}_{u2}$ を特徴点として, ユーザに選ばせる.
4) $\mathbf{v}_{u2}$ から R にある頂点の集合 V_2 を求める.

5) V_1とV_2が重なるところを次の特徴点にする.
6) 同様に, V_i ($i=3, \ldots, F$) を求め, V_{i-1}と重なる頂点を次の特徴点とする(このとき, 今まで決めてきた特徴点と近過ぎない点を選ぶ).
7) 操作 6)を繰り返し, すべての特徴点を決める.

本研究では, 操作 2)で指定する位相距離をRとして, 各領域に含まれる頂点数を元に実験的に求めた値を使っている.

3. 実験結果

我々は 2 節で述べたアルゴリズムを C++と GUI ツールキットfltk (http://www.fltk.org) を用いて実装し, 実験を行った.

3.1. 妨害に対する耐性

ポリゴンメッシュに加えられる各種の変更に対する頑強性について実験を行った.

先行論文[大渕 01]で示したように, 本手法で埋め込まれた透かしは頂点接続性を変更しない妨害(相似変換, ランダムノイズ付加, スムージング等)に対して耐性を持った. 相似変換に対して, 本透かしは頑強である. また, 拡散率cを大きくするとランダムノイズ付加に, 小さくするとメッシュスムージングに対する耐性が増加する. 詳しくは先行論文[大渕 01]を参照して欲しい.

また, 同じく先行論文[大渕 01]で述べたように, 透かしを埋め込む際に, メッシュに領域分割を施しておけば, メッシュ形状を部分的に削除されても, 残った領域から透かしを抽出することができた. また, メッシュ形状を部分的に削除された後, 相似変換を加えらたメッシュに対しても, 削除されなかった領域で相似変換を補正すれば, 透かしを取り出すことができた.

3.1.1. メッシュ簡単化に対する耐性

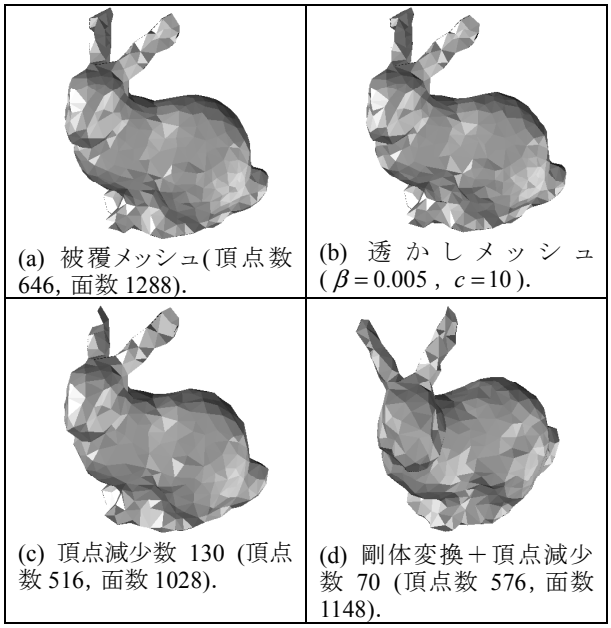
本手法で埋め込まれた透かしは, リサンプリングを用いれば, メッシュ簡単化によってある程度幾何形状が変化し, さらに剛体変換を加えられた透かしメッシュからでも取り出せた.

図 9 は, bunny0(図 9(a)頂点数 646, 面数 1288)に透かしを無分割で埋め込んだ後(図 9(b)), メッシュ簡単化によって頂点数を 130 減らしたメッシュ(図 9(c)), メッシュ簡単化によって頂点数を 70 減らし剛体変換を行ったメッシュ(図 9(d))である. 透かしを埋め込む際には振幅率$\beta=0.005$, 拡散率$c=10$とした.

図 9(c)の場合はリサンプリング行うことで, 図 9(d)の場合は位置合わせの後リサンプリングを行うことで, 損失なく透かしを取り出すことができた. ここで, 図 9(c)に比べて, 剛体変換を加えた場合である図 9(d)の頂点減少数が少ない. これは, 剛体変換が加わると位置合わせの必要があり, メッシュ簡単化によって頂点数を多く減少させるとリサンプリングに必要な位置合わせがうまく行かなくなるためである.

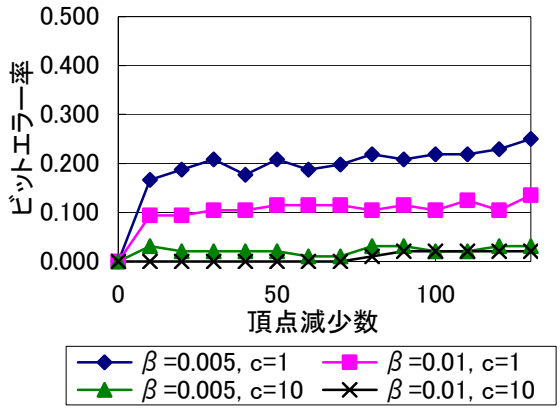
図 10 は, bunny0 (図 9(a))に対して, 無分割で透かしを埋め込み, メッシュ簡単化によって頂点数を減らした場合のビットエラー率を示している. ここで, ビットエラー率とは, 取り出しに失敗したビットの数を, 埋め込んだ透かしのビット数で割った値である. 取り出した値に誤りがなければ 0 となる.

図 10 からわかるように, メッシュ簡単化のみの場合(図 10(a)), 剛体変換とメッシュ簡単化と合わせた場合(図 10(b)), いずれも変調振幅率βや拡散率cが大きいほど, 頑強性が増した. ただし, モデルの形状によってはビットエラー率が大きくなることがある.

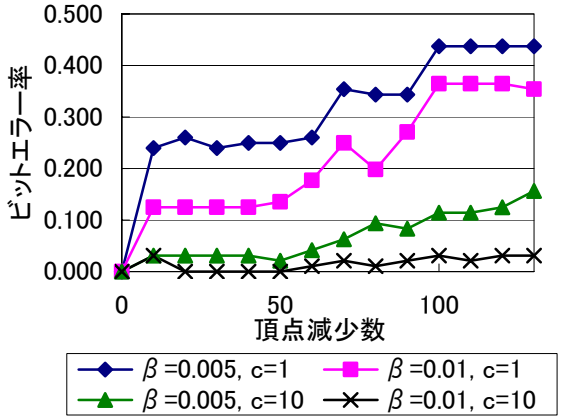


(a) 被覆メッシュ(頂点数 646, 面数 1288).
(b) 透かしメッシュ($\beta=0.005$, $c=10$).
(c) 頂点減少数 130 (頂点数 516, 面数 1028).
(d) 剛体変換＋頂点減少数 70 (頂点数 576, 面数 1148).

図 9 透かしメッシュに簡単化を行った例(bunny0).



(a) メッシュ簡単化のみ.



(b) 剛体変換とメッシュ簡単化.

図 10 メッシュ簡単化を加えた際の頂点減少数とビットエラー率.

3.2. 領域分割による処理時間の短縮

表 3 は透かし埋め込みに要する時間をメッシュの大きさ(頂点数)と領域分割数で比較したものである. 表中「(疎)」は疎分割で領域分割を行った結果を示す. 領域分割の処理時間の大半は特徴点を決定するのに費やされている.

領域分割無しの場合, 2218 頂点のメッシュ bunny2 の透かし処理時間は 3 時間弱かかり, 実用的でない. しかし, 領域分割を施し, 分割後の領域の大きさを一定値以下(例えば 500 頂点程度以下)に押さえれば, 大きなメッシュに対する透かし埋め込みが可能である. 表 3 からわかるように, 上述の bunny2 に対する埋め込みも, 5 つの領域への密分割で 8 分強, 3 領域への疎分割では 4 分弱となった. また, 疎分割を用いると, 頂点数が 10000 を超えるメッシュの透かし処理も 10 数分程度と実用的な処理時間の範囲内である.

スペクトル分解の処理時間は, 分割することで得る領域の大きさに大きく依存する. 分割数を増やして領域を小さく取ると全体の処理時間は減り, また部分的切り取りに対する耐性は高くなる. 反面, 埋め込み情報量が減り, ノイズやスムージングに対する耐性も低下する.

表 3. スペクトル分解と領域分割に要した時間.

モデル	頂点数	領域数	処理時間	
			スペクトル分解	領域分割
bunny2	2218	無分割	2h54m40s	
bunny2	2218	5	8m08s	1s
bunny2	2218	3(疎)	3m46s	0s
bunny3	4114	10	21m16s	11s
bunny3	4114	7(疎)	5m17s	5s
bunny4	13990	34	53m12s	31m26s
bunny4	13990	20(疎)	13m20s	24m44s

4. まとめと今後の課題

本論文では, 3 次元メッシュに対しそのメッシュスペクトル領域で透かしを埋め込む先行論文の手法[大渕 01]を改善する提案を行った.

先行論文では, ポリゴンメッシュで定義された 3 次元形状をスペクトル分解し, そのスペクトル(周波数成分)を変更することで透かしを埋め込む手法を提案した. この透かし手法は, 秘密透かしに分類され, メッシュに加えられる相似変換(回転, 並行移動, 一様スケーリング), 頂点座標へのランダムノイズ重畳, 形状のスムージング, メッシュの部分的切り取りに耐性を持った. しかし, ポリゴン簡単化を加えられ頂点の接続性を変更されると透かしが壊れる, 頂点数の多いモデルにおいて処理時間が必要以上にかかる, という問題があった.

改善策として, メッシュ簡単化に対する耐性を付与するため, 形状のリサンプリングを用いた. これにより, 被覆メッシュと同一接続性で, かつ, 透かしメッシュの幾何形状を持つメッシュを作り, これから透かしを取り出すことで対応した. リサンプリングの必要な被覆メッシュと透かしメッシュの向き合わせは, メッシュの面に質量が分布すると仮定し, これをモンテカルロ近似して慣性主軸を求めた. また, 光線追跡による幾何形状のリサンプリングを高速化するため, 空間の均等分割による交差判定の刈り込みを導入した. さらに, 透かし処理の時間を短縮するため, どの領域にも含まれない頂点を存在させる疎分割を導入した. 疎分割により領域数が減り, 各領域の頂点数がほぼ均一になるため, 全体の処理時間が短縮された.

今後, リサンプリングのための位置合わせ手法の更なる改善が必要と考える. これにより, 現在の剛体変換よりも広いクラスの幾何変換(例えば相似変換)とメッシュ簡単化との組み合わせ, あるいは, メッシュ簡単化, 切り取り, 幾何変換の 3 種の組み合わせを加えても透かしが取り出せるようにしたい.

参考文献

[Bollobás98] B. Bollobás, *Modern Graph Theory*, Springer, 1998.

[Benedens99] O. Benedens, Geometry-Based Watermarking of 3D Models, *IEEE CG&A*, pp. 46-55, January/February 1999.

[Gottschalk96] S. Gottschalk, M.C. Lin, D. Manocha, OBBTree: A Hierarchical Structure for Rapid Interference Detection, *Proc. SIGGRAPH '96*, pp. 171-180, 1996.

[GraphicsGems94] Paul S. Heckbert. Ed, *Graphics Gems IV,* AP Professional, 1994

[Hartung98] F. Hartung, P. Eisert, and B. Girod, Digital Watermarking of MPEG-4 Facial Animation Parameters, *Computer and Graphics*, **22**(4), pp. 425-435, Elsevier, 1998.

[Kanai98] S. Kanai, H. Date, and T. Kishinami, Digital Watermarking for 3D Polygons using Multiresolution Wavelet Decomposition, Proc. of the *Sixth IFIP WG 5.2 GEO-6*, pp. 296-307, Tokyo, Japan, December 1998.

[Karni00] Z. Karni, C. Gotsman, Spectral Compression of Mesh Geometry, Proc. *SIGGRAPH* 2000, pp. 279-286, 2000.

[Katzenbeisser00] S. Katzenbeisser, F. A. P. Petitcolas, *Digital Watermarking*, Artech House, London, 2000.

[Ohbuchi98] R. Ohbuchi, H. Masuda, and M. Aono, Geometrical and Non-geometrical Targets for Data Embedding in Three-Dimensional Polygonal Models, *Computer Communications*, Vol. 21, pp. 1344-1354, Elsevier (1998).

[Osada01] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin, Matching 3D Models with Shape Distributions, Proc. *International Conference on Shape Modeling and Applications 2001* (*SMI 2001*), pp. 154-166, Genova, Italy, May 7-11, 2001 (IEEE Computer Society Press).

[Taubin95] G. Taubin, A Signal Processing Approach to Fair Surface Design, Proc. *SIGGRAPH '95*, pp. 351-358, 1995.

[Yeo99] B-L. Yeo and M. M. Yeung, Watermarking 3D Objects for Verification, *IEEE CG&A*, pp. 36-45, January/February 1999.

[大渕01] 大渕 竜太郎, 高橋 成雄, 宮澤 貴彦, 向山明夫, スペクトル分解を用いた3次元メッシュへの電子透かしの埋め込み, 情報処理学会論文誌, 2001年5月, **42**(5), pp. 1103-1114.

[金井00] 金井 崇, 表示・変形・圧縮のための多重解像度表現技術, 情報処理, 41巻, 10号, pp. 1108-1112, 2000年10月.

[松井98] 松井 甲子雄, 電子透かしの基礎, 森北出版, 1998年8月.

[宮澤01] 宮澤 貴彦, 向山 明夫, 高橋 成雄, 大渕 竜太郎, スペクトル分解を用いた3次元メッシュ電子透かしの耐性強化法, 情報処理学会 グラフィクスとCAD研究会, 2001年2月, **2001**(14), pp. 55-60.