画像電子学会・情報処理学会グラフィクスとCAD研究会共催
Visual Computing グラフィクスとCAD 合同シンポジウム 2002, pp. 171-176

# 周波数領域で埋め込む3次元メッシュ電子透かしの改良
# Improving the frequency-domain 3D mesh watermarking algorithm

向山 明夫[1], 大渕 竜太郎[2], 高橋 成雄[3]
Akio Mukaiyama[1], Ryutarou Ohbuchi[2], Shigeo Takahashi[3]
k7186@kki.yamanashi.ac.jp, ohbuchi@acm.org, takahashis@acm.org

[1] 山梨大学大学院工学研究科 Graduate School of Engineering, Yamanashi University
[2] 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
[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

電子透かしと呼ばれる技術は，情報を表現する構造体(透かし)を，埋め込み対象となるコンテンツ(静止画像，動画像，音声データ，3 次元(3D)モデル，等)に付加する(詳しくは[松井 98，Cox02] 等を参照)．透かしを付加する際には，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)を阻害しないように，かつ，透かしがコンテンツから容易に除去できないようにする必要がある．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．

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, Praun99, Yin01, 大渕 01, 向山 01]など)．3 次元形状を対象とする透かし手法においても，画像への透かしの場合と同様，なんらかの変換領域，例えば形状の周波数に相当する領域で埋め込む透かし手法が主流となりつつある．この理由として，変換領域で埋め込む透かし手法に利点があるためである．画像の例でいえば，人が知覚しにくい周波数帯の係数を変えて埋め込めば，妨害耐性を変えずにより気づかれにくい透かしが実現できる．

メッシュに対する電子透かしの中で，変換領域に埋め込む透かしを最初に提案したのは Kanai らである[Kanai98]．Kanai らはメッシュをウエーブレット変換し，その変換領域に透かしを埋め込んだ．この透かしはアフィン変換，および頂点にランダムノイズを付加するような妨害に対して耐性を持つ．しかし，欠点として埋め込み対象が 1 対 4 の再分割接続性を持つメッシュに限られる．Praun らは空間領域における多重解像度基底関数を使って透かしを埋め込んだ[Praun99]．Praunらの透かしは相似変換，切り取り，ランダムノイズの重畳などの妨害に対して耐性がある．しかし，頂点数あたりの埋め込み情報量が少ないため，埋め込み対象が頂点数の多いメッシュに限られる．Yin らはメッシュ上に生成した Burt-Adelson ピラミッド[Guskov99]による多重解像度分解を用い，その変換領域で透かしを埋め込んだ[Yin01]．Yin らの透かしは，Praun らの手法と同様に，さまざまな妨害に対し耐性を持つ．その上，Praun らの手法より頂点数あたりの埋め込み情報量が多い．

我々が先に提案した手法[大渕 01，向山 01]は，メッシュ形状のスペクトル分解[Karni00]を用い，メッシュ形状の低周波成分に透かしを埋め込んだ．この透かしは，相似変換，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．しかし，この透かしには，(1) 複合妨害，例えば相似変換・メッシュ簡単化・部分切り取りを合わせた妨害に対する耐性が無い，(2) スペクトル分解に用いる固有値分解の処理に時間がかかるため大規模なメッシュに対する透かし処理ができない，という 2 つの弱点があった．

本論文では，以前我々が提案した透かし手法[向山 01]を元に，上記 2 つの弱点を改善した電子透かし手法を提案する．第一に，複合妨害の耐性強化のため，モデルの一部分に基づく高精度の位置合わせ手法を用いた．第二に，固有値分解の処理時間の改善のため，より効率的な固有値分解計算アルゴリズムを用いた．改良の結果，本透かしは，相似変換・頂点接続性変更(メッシュ簡単化など)・部分切り取りを合わせた妨害に耐性を持ち，かつ，頂点数数万のメッシュに対しても埋め込めるようになった．例えば実験の結果，本手法を用いることで，頂点数約 48000 のメッシュに透かしを埋め込み，簡単化によって頂点数を半分以上に減らし，さらに相似変換と部分切り取りを行ったメッシュからでも損失なく透かしを取り出せた．

本論文の構成は以下のとおりである．第 2 章では，2.1 節で基本となる透かし処理を，2.2 節で位置合わせ手法の改善手法を，2.3 節で固有値分解処理の効率改善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．

## 2. スペクトル分解による電子透かし

本手法の透かし処理の基本は先行手法[向山 01]と同様である．透かしの埋め込みにおいては，元となるメッシュ(被覆メッシュ)をスペクトル分解[Karni00]して得られた形状の「周波数」表現であるスペクトル係数の値を透かし情報に応じて変更する．透かしの入ったメッシュ(透かしメッシュ)は，固有ベクトルと透かしにより変更したスペクトル係数の一次結合で求まる．透かしの取り出しには，透かしメッシュと被覆メッシュの双方を使用する(このような透かしを「秘密透かし」と呼ぶ)．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，得られた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，本手法および先行研究ではキルヒホフ行列[Bollobás98]を用いる．メッシュスペクトルは，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(低周波成分)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(高周波成分)を，それぞれ表現する．

図 1 に本手法の処理の流れを示す．この図で太線の処理は改良した処理を示す．本手法では，先行手法に対し，以下の 2 つの処理を改良した．

- **位置合わせ**：先行手法の慣性主軸に加えて，より高精度の位置合わせを行うため，最適化アルゴリズムを用いる．高精度の位置合わせを行わないと次のリサンプリングが失敗する．
- **スペクトル分解**：Arnoldi 法を用いることで，先行手法よりも効率的な固有値分解計算を行う．埋め込み処理はほとんどが固有値分解処理なので大幅な改善が行える．

### 2.1. 透かし処理の基本アルゴリズム

透かしを埋め込む際(図 1 左)，まず，被覆メッシュ $M$ に対し領域分割を行う．複数の小領域に分け各小領域ごとに同一の透かしを埋め込めば，部分切り取りに対する耐性が得られる．また，領域分割の情報は取り出しの際に使用するので保存しておく．次いで，透かしデータ $\mathbf{w}$ を拡散率 $c$ 回複製する．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[Hartung98]．複製した透かしデータに応じてスペクトル係数を変調振幅 $\alpha$ だけ変更することで透かしデータを埋め込む．ここで $\alpha$ は，被覆メッシュを囲う最小の直方体である axis-aligned Bounding Box を計算し，その辺の長さの最大長 $\phi$ に対する割合(振幅率) $\beta$ を指定することで $\alpha = \phi \cdot \beta$ のように指定する．透かしを埋め込んだメッシュ $M'$ は透かしで変調された係数を逆変換して得られる．

本手法は秘密透かしであり，取り出しには透かしメッシュ $M'$ と被覆メッシュ $M$ とが必要である(図 1 右)．取り出しでは，まず $M'$ と $M$ の位置・大きさ・向きを揃え(これを「位置合わせ」と呼ぶ)，次いでリサンプリングにより $M'$ 上に埋め込み時と同様の頂点接続性が作られる．さらに，$M$ と $M'$ 上には埋め込み時と同一の領域分割も施される．こうして得られた 2 セットの小領域をスペクトル分解しその係数を比較する．その比較結果と埋め込みの際に使用した拡散率 $c$ とから透かしデータ $\mathbf{w}$ を復元する．

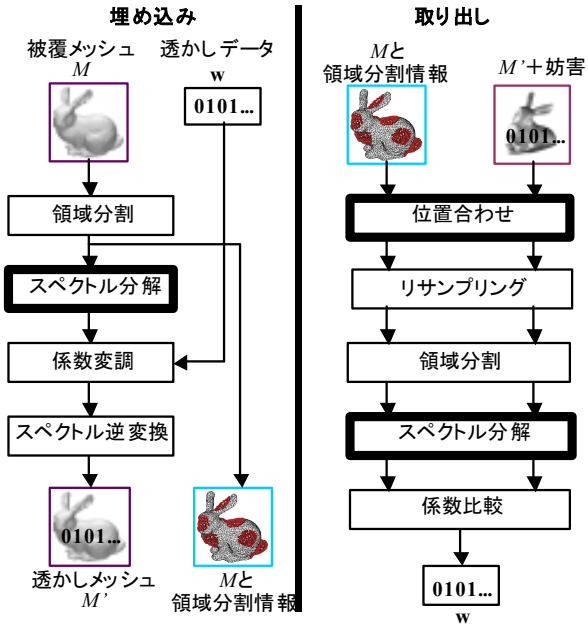



図 1. スペクトル分解を用いた電子透かしの処理．

### 2.2. 位置合わせ手法の改善

今回の改良では，最適化アルゴリズムを使用した高精度の位置合わせを用いる．まず手動で位置合わせをする領域を選択し，慣性主軸を使用したおおまかに位置合わせ[向山 01]を行い，さらにモデルの小領域を用いて最適化アルゴリズムを使用した詳細な位置合わせを行う．本手法を用いることで先行研究の欠点であった，頂点接続性変更・相似変換・部分切り取りを合わせた妨害に対する耐性を改善できる．

本手法では，最適化アルゴリズムを使用した位置合わせには，Praun らが使用した位置合わせ手法[Chen92, Praun99]を用いた．最適化を行う対象となる値は次のように求める(図 2)．まず，透かしメッシュ $M'$(面数 $n$)のある面の重心 $\mathbf{g}_i$ $(i = 1,2,\ldots,n)$ の法線と，被覆メッシュ $M$ との交点 $\mathbf{r}_i$ を求める．次いで，$\mathbf{r}_i$ が存在する面分を含む平面 $f_i$ に直行し，$\mathbf{g}_i$ を通過する直線を求め，その直線と $f_i$ の交点 $\mathbf{h}_i$ を求める．そして，$\mathbf{g}_i$ と $\mathbf{h}_i$ との幾何的な距離 $d_i$ を計算する．後は同様に，他の面の重心から $d_i$ を求め，それらを合計す

る. この合計の値が最適化の対象となる. $M'$に回転, 平行移動, 一様スケーリングを徐々に加え, 合計が最小となるように最適化を行う. 本手法では, 反復最適化を行うには最急降下法の一種である Powell 法 [Press92] を用いた. $M'$の面数が多い場合, メッシュ全体に散らばるように面を選び, 選んだ面の重心を使って最適化を行えば処理時間を短縮できる.

透かしメッシュに対し部分切り取りが行われた場合, どの部分が切り取られず残るかわからない. そこで, 被覆メッシュと透かしメッシュの両方で位置合わせのためのメッシュ領域を生成し(図 3), これら領域を使用して位置あわせし, その後にサンプリングを行う. 各領域の指定は, 領域の中心となる特徴点を1つ手動で指定して行う. 被覆メッシュと透かしメッシュの領域が多少異なっても位置合わせが可能である.

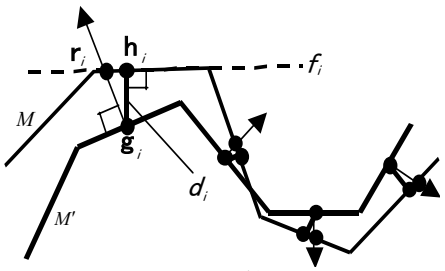
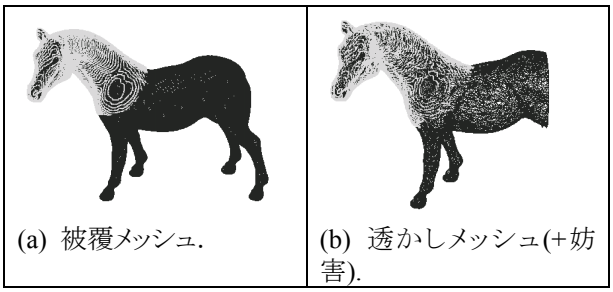



図 2. メッシュ間の誤差の計算(2 次元で図示).



(a) 被覆メッシュ.　(b) 透かしメッシュ(+妨害).

図 3. 位置合わせする領域の選択の例(黒色部以外が選択領域).

### 2.3. 固有値分解処理の効率改善

今回, スペクトル分解の処理時間を改善するために Arnoldi 法 [Golub96] を導入する. Arnoldi 法により頂点数が数千のメッシュに対しても実用的な時間でスペクトル分解を行うことが可能になる.

Arnoldi 法とは, 次元が大きい正方行列に対し, 固有値分解を行う手法である. この手法を行うことで, 値が最も小さい, および大きい固有値とそれぞれに対応する固有ベクトルから求めて行くことができる. この手法は, 次元が数百程度の小さい行列に対しては, 得られる固有値・固有ベクトルが不安定である. しかし, 次元が数千程度の大きな行列に対しては, 小さい, および大きい固有値と対応する固有ベクトルを安定して求めることができる.

我々の先行手法[向山 01]では頂点数数百程度のメッシュが透かしの対象であった. この理由は, スペクトル分解のために行う固有値分解の際, 全ての固有値・固有ベクトルを求めるため計算時間が非常にかかるためであった. 先行手法では全ての固有値・固有ベクトルを求めるために, Housholder 法と Jacobi 法を合わせた手法(Housholder+Jacobi 法)を用いていた. しかし, 本透かし手法では埋め込みに必要な固有値・固有ベクトルだけで十分で, 全ての固有値・固有ベクトルを求める必要はないため Arnoldi 法によりスペクトル分解の処理時間を改善できる.

必要となる固有値・固有ベクトルの数は使用目的およびメッシュの形状に依存する. 例えば, ランダムノイズ付加に対して耐性を高めたいのならば, 拡散率を大きくする必要があるため, 多くの固有値・固有ベクトルが必要となる. スムージングに対して耐性を持たせたいのならば, 影響が小さい低周波成分に透かしを埋め込めば良いので, 小さい固有値に対応する固有ベクトルを求めれば良い. また, 人工物の形状(例えば distcap など)のように, 平らな面の多いメッシュはスムージングの影響を受け大きく形状が変化するため, スムージングに対する耐性を高めるには拡散率を小さくし, 低周波成分に透かしを埋め込む必要がある. そのため, 求める固有値・固有ベクトルの数は少なくて良い.

本手法では, Arnoldi 法を導入するため, Arnoldi 法を含んだライブラリ[Stewart]を用いた. このライブラリの Arnoldi 法を用いるためには, 求める固有値・固有ベクトルの数(大きい固有値から求めるものと小さい固有値から求めるものとの総数)を指定する必要がある. 例えば, $n$ 次元の正方行列に対し, $2m(2m<n)$ の固有値・固有ベクトルを求めるよう指定した場合, 固有値が小さいほうから $m$ 個, 大きいほうから $m$ 個の固有値・固有ベクトルを求める. そのため, $2m \ll n$ になると, $n$ 個全てのの固有値・固有ベクトルを求める場合に比べて大幅に処理時間の改善ができる.

表 1. 固有値分解 2 手法の比較.

| 頂点数 | 使用手法 | 時間 |
|---|---|---|
| 2218 | Householder+Jacobi | 1h46m44s |
| | Arnoldi | 53m42s |

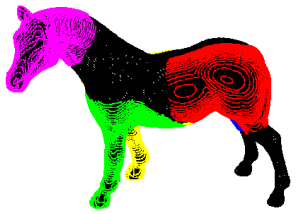
表 2. Arnoldi 法の計算時間.

| メッシュの頂点数 | 求めた固有値の数 | | |
|---|---|---|---|
| | 500 | 1000 | 1500 |
| 2218 | 1m05s | 4m55s | 13m56s |
| 7565 | 3m56s | 13m20s | 30m46s |
| 18957 | 13m38s | 35m25s | 1h13m18s |

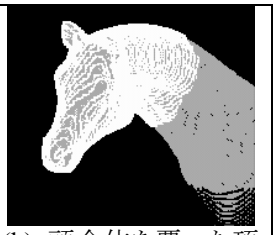
表 1 は, Housholder+Jacobi 法または Arnoldi 法を使用し, 頂点数 2218 のメッシュを固有値分解した時間を比較した結果である. 使用した PC は, CPU が Athlon1900+, メモリが 1.5GB である. 表 1 からわかるように, Arnoldi 法のほうが, 2 倍近く処理が速い.

表 2 に，メッシュの頂点数と求める固有値・固有ベクトルの数との計算時間の関係を示す．使用した PC は表 1 のときと同じである．先行研究の手法では，処理時間が長いために頂点数が数千のメッシュの固有値分解は困難だった．しかし，表 2 からわかるように，Arnoldi 法を用いれば頂点数が数千以上のメッシュの固有値分解が実用的時間で可能である．

上述のように Arnoldi 法を用いることで頂点数が多いメッシュに対して固有値分解を行うことが可能となった．そのため，総頂点数が約 5 万のメッシュの特徴的な部分にも透かしを埋め込める．図 4 はメッシュ horse (頂点数 48485，面数 96966) を 5 つの小領域に疎分割 [向山 01] した例である．各小領域の頂点数は約 7000 である．図 3(b) に示すように，頂点数約 7000 の小領域で頭全体を覆うことができる．このように，Arnoldi 法を用いることで大規模なメッシュの特徴的な部分に固有値分解を行うことができ，透かしが埋め込めるようになった．



(a) 5 つの小領域に疎分割を行ったメッシュ(黒色部以外が小領域).



(b) 頭全体を覆った頂点数 7063 の小領域（白色部分）.

図 4. 特徴的な部分を含んだ疎分割を行ったメッシュ horse(頂点数 48485，面数 96966).

## 3. 実験結果

我々は 2 節で述べたアルゴリズムを C++を用いて実装し，実験を行った．

### 3.1. 改良した位置合わせによる耐性強化

今回最適化アルゴリズムを用いることで位置合わせを改良した．そこで最適化アルゴリズムの有無で妨害に対する耐性の変化を調べた．

図 6 は，bunny1 (図 5)に対して，振幅率 $\beta$ と拡散率 $c$ とをある値にし，無分割で 32 ビットの透かしを埋め込み，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．ここでは 3 種類の透かし(全て 0，全て 1，ランダム)を使用した．図 6 には，その 3 種類の透かしの平均ビットエラー率を示した．図 6(a) は，位置合わせの際には慣性主軸の位置合わせだけを，図 6(b)は慣性主軸と最適化アルゴリズムを用いた位置合わせを行った結果である．

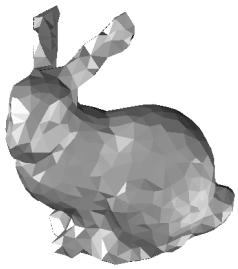
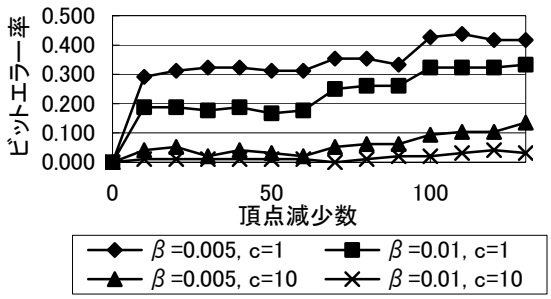
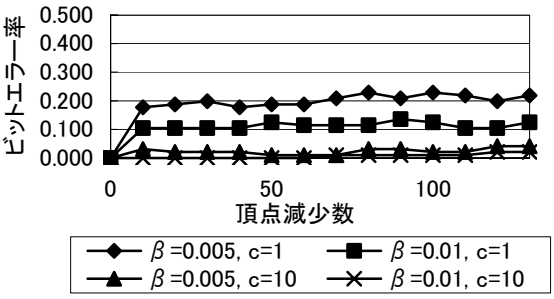


図 5. メッシュ bunny1(頂点数 646，面数 1288).




(a) 相似変換とメッシュ簡単化．慣性主軸の位置合わせのみを実行.




(b) 相似変換とメッシュ簡単化．慣性主軸と最適化アルゴリズムの位置合わせを実行.

図 6. 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.

図 6(a) (b)からわかるように，慣性主軸と最適化アルゴリズムを用いた位置合わせを行うことでビットエラー率がさらに減少する．

### 3.2. 大規模メッシュに対する透かし埋め込み

頂点数が数万の比較的大きなメッシュに対し実験を行った．図 6 にメッシュ horse (頂点数 48485，面数 96966)を，図 7 にメッシュ bunny2 (頂点数 34839，面数 69674) を用いた実験結果の例を示す．図 6(a)のメッシュを図 6(b)のように疎分割を行い，各小領域に透かしを32ビット埋め込んだメッシュが図6(c)である．このとき振幅率 $\beta = 0.001$ ，拡散率 $c = 10$ である．同様に，図 7(b)のように疎分割を行い図 7(a) に透かしを埋め込んだメッシュが図 7(c)である．図 6(a) (c)，図 7(a) (c) のように，透かし埋め込み前後では見た目にそれほど違いがない．

図 6(d)~(i)は図 6(c)のメッシュにメッシュ簡単化やリメッシュ(MeshToss [MeshToss01] を使用)，それらの

操作と他の操作を合わせた妨害を与えたメッシュである．同様な妨害を図 7(c)に与えた例が図 7(d)~(f)である．本手法を用いれば，図 6(d)~(i)，図 7(d)~(f)のようなメッシュからでも，損失なく透かしを取り出すことができた．

透かしを取り出す際，複数の領域から取り出した複数の透かしを使って簡単な誤り訂正を行い，最終的な透かしを決定した．修正方法は，複数の透かしの同じ桁のビットを比較し，多数決により，0と1のうち数が多いものをそのビットの最終的な透かしとする．図 6(f)(i)，図 7(e)(f)のような部分切り取りされたメッシュの場合は，切り取られずに残った領域から透かしを取り出し，修正を行った．

取り出しの際には，図 6(d)(g)と図 7(d)にはリサンプリングのみを，図 6(e)(h)には位置合わせとリサンプリングを使用した．また，図 6(f)(i)と図 7(e)(f) のように部分切り取りが行われたメッシュの場合は，被覆メッシュと透かしメッシュの両方においてそれぞれ手動で領域を決め，これら領域を使用して位置あわせし，その後サンプリングを行った．

メッシュ簡単化やリメッシュ，部分切り取りによって透かしメッシュの頂点数を大幅に減らすと，透かしの取り出しに失敗した．この原因は，メッシュの形状が大きく変形してしまい位置合わせがうまく行かなかったためである．

## 4. まとめと今後の課題

本研究では，以前我々が提案した手法[向山 01]を改善した．今回の改善により，相似変換・頂点接続性変更・部分切り取りを合わせた妨害に対する耐性が得られ，また大規模メッシュに透かしを埋め込めることができるようになった．具体的には，位置合わせ手法を改善してほぼ任意の部分メッシュを用いて位置合わせが可能となったこと，また，Arnoldi 法で固有値分解の計算を行ったことである．その結果，例えば頂点数が約48000の大きなメッシュに透かしを埋め込むことができ，さらに，その透かしメッシュを簡単化して頂点数を半分以上に減らし，相似変換と部分切り取りを行ったメッシュからでも損失なく透かしを取り出すことができた．

今後，位置合わせ手法を含め，リサンプリング手法の更なる改善が必要である．例えば，現在，メッシュ簡単化，部分切り取り，幾何変換の 3 種を合わせた妨害が加えられた場合，手動で位置合わせを行う領域を選択する必要がある．このような複合的な妨害が加えられた場合でも自動で位置合わせが行えるようにしたい．また，現在は本透かし手法では対応できないアフィン変換に対する耐性も付加したい．

また，3 次元メッシュ電子透かしの研究を促進するためには，複数の手法を同じ条件で評価し比較し，結果を数値化するための枠組みが必要である．画像の透かしには事実上の標準といえる画像群や攻撃手法，画像の歪の測定基準などがある．3 次元メッシュの透かしについても，標準となる 3 次元モデルのセット，標準な攻撃のセット，幾何形状や接続性を考慮したモデルの「歪」の尺度，等を用意する必要があるだろう．

**謝辞**：本研究は文部化学省科学研究費補助金(課題番号 14580369)からの助成による．

## 参考文献

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

[Chen92] Y. Chen, G. Medioni, Object modelling by registration of multiple range images, *Image and Vision Computing* 10, 3(Apr. 1992), pp. 145-155.

[Cox02] I. J. Cox, M. L. Miller, J. A. Bloom, *Digital Watermarking*, Morgan Kaufman Publishers, 2002.

[Golub96] Golub, G. H., Van Loan, C. F., *Matrix Computations*, Third Edition, Johns Hopkins University Press, 1996.

[Guskov99] I. Guskov, W. Swelden, P. Schröder, Multireslution signal processing for meshes, Proc. *SIGGRAPH '99*, pp. 325-334, 1999.

[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.

[MeshToSS01] 金井 崇, *MeshToss*, Version 1.0.1, http://graphics.sfc.keio.ac.jp/MeshToSS/indexE.html.

[Praun99] E. Praun, H. Hoppe and A. Finkelstein, *Robust Mesh Watermarking*, Proc. *SIGGRAPH '99*, pp. 49-56(1999).

[Press92] W.H. Press et al, Numerical Recipes in C-The Art of Scientific Programming, 2nd Ed., Cambridge University Press, Cambridge, UK, 1992.

[Stewart] David E. Stewart, Zbigniew Leyk, *Meschach Library* Version 1.2b, http://www.netlib.org/c/meschach.

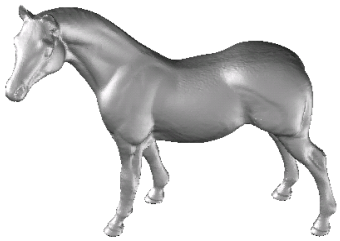
[Taubin95] G. Taubin, A Signal Processing Approach to Fair Surface Design, Proc. *SIGGRAPH '95*, pp. 351-358, 1995.

[Yin 01] K. Yin, Z. Pan, J. Shi, Robust mesh watermarking bassed on multiresolution processing, *Computer and Graphics*, 25, pp. 409-420, 2001.

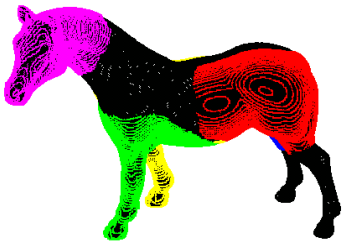
[大渕01] 大渕 竜太郎，高橋 成雄，宮澤 貴彦，向山 明夫，スペクトル分解を用いた3次元メッシュへの電子透かしの埋め込み，情報処理学会論文誌，2001年5月，**42**(5), pp. 1103-1114.

[松井98] 松井 甲子雄，電子透かしの基礎，森北出版 (1998)

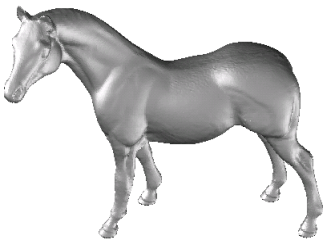
[向山01] 向山 明夫，宮澤 貴彦，高橋 成雄，大渕 竜太郎，スペクトル変換領域で埋め込む3次元メッシュの電子透かし，画像電子学会 情報処理学会 Visual Computing グラフィクスとCAD合同シンポジウム2001，2001年6月．



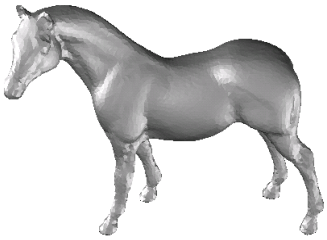
(a) 被覆メッシュ(頂点数 48485，面数 96966).



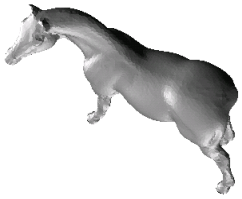
(b) 5 つの小領域に疎分割(黒色部以外が小領域，各小領域の頂点数約 7000).



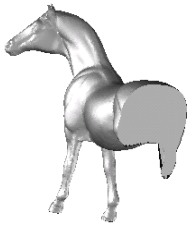
(c) 透かしメッシュ(各領域において $\beta = 0.001$ , $c = 10$ ).



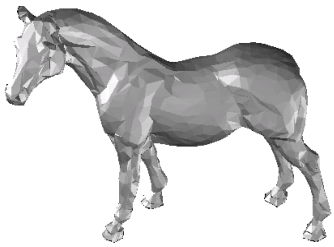
(d) メッシュ簡単化のみ(頂点数 7485，面数 14966).



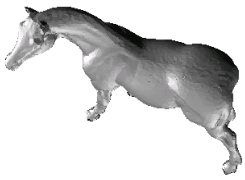
(e) メッシュ簡単化＋相似変換(頂点数 7485，面数 14966).



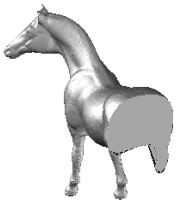
(f) メッシュ簡単化＋部分切り取り＋相似変換(頂点数 25000 にした後切除).



(g) リメッシュのみ(頂点数 2000，面数 5994).

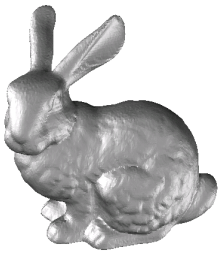


(h) リメッシュ＋相似変換 (頂点数 8000，面数 15996).

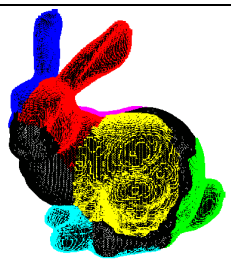


(i) リメッシュ＋部分切り取り＋相似変換(頂点数 30000 にした後切除).

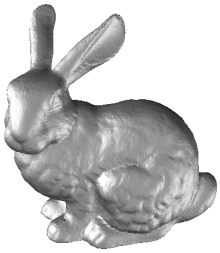
図 6. メッシュ horse を使用した実験結果の例.



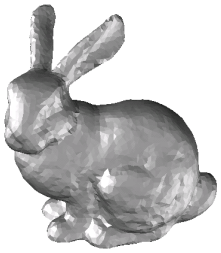
(a) 被覆メッシュ(頂点数 34839，面数 69674).



(b) 6 つの小領域に疎分割 (黒色部以外が小領域，各小領域の頂点数約 4000).



(c) 透かしメッシュ (各領域において $\beta = 0.001$ , $c = 10$ ).



(d) メッシュ簡単化のみ(頂点数 4839，面数 9674).

(e) メッシュ簡単化＋部分切り取り＋相似変換(頂点数 17839 にした後切除).

(f) リメッシュ＋部分切り取り＋相似変換(頂点数 18839 にした後切除).

図 7. メッシュ bunny2 を使用した実験結果の例.