グラフィクスとCAD 96－2
（1999．8．23）

円盤と位相同型の任意の三角形メッシュ間の高速な合成法

金井 崇　鈴木 宏正†　木村 文彦†
理化学研究所 素形材工学研究室
〒351-0198 埼玉県和光市広沢 2-1
† 東京大学大学院工学系研究科 精密機械工学専攻
E-mail: *kanai@postman.riken.go.jp*

本論文では，コンピュータグラフィックスの分野で一般的に用いられている形状の表現形式である，三角形メッシュを合成するための手法について議論する．メッシュを合成するという操作は，三次元形状モーフィングにおいて一般的に用いられている．筆者らの手法は，調和写像を用いて二次元平面内に三角形メッシュを展開することを基本としている．円盤に位相同型な形状に限定されているものの，形状の幾何学的性質には依存しない，というのが特徴である．本論文では，特に合成のためのアルゴリズムについての詳細を述べ，主に計算速度，頑健性について改良されていることを示す．

A Fast Combining Method of Two Arbitrary Triangular Meshes Homeomorphic to a Disk

TAKASHI KANAI　HIROMASA SUZUKI†　FUMIHIKO KIMURA†
Materials Fabrication Laboratory, RIKEN
2-1 Hirosawa, Wako-shi Saitama, 351-0198, Japan.
†Dept. Precision Engineering, University of Tokyo.
E-mail: *kanai@postman.riken.go.jp*

This paper discusses a combining method of two triangular meshes that are frequently used in the Computer Graphics area. The operation of combining meshes is used for the three-dimensional (3D) morphing. The basic idea of our method is to embed each of two meshes to a polygon into a unit circle in $\mathbf{R}^2$ using harmonic mapping. While the mesh we can treat is limited to a topological disk, but not depend on the geometric property. This paper mainly describes the detail of the algorithm for combining two embedded meshes, and shows that our algorithm is numerically robust and is fast especially for the case that the number of faces is large.

1　はじめに

三角形ポリゴンの集まりで構成される三角形メッシュ（以下略してメッシュと呼ぶ）は，コンピュータグラフィックス（CG）の分野などで扱われる基本的な表現形式であり，多くの CG モデラで生成，操作することができる．本論文では，このうち円盤と位相同型である任意の二つのメッシュを合成するための手法について述べる．ここでいう“合成”というのは，同じ位相を持つ二つのメッシュのグラフ構造を併せ持つような，一つのメッシュを生成することである．以下，このメッシュのことを合成メッシュと呼ぶことにする．この際，生成される合成メッシュはもとのメッシュと同じ位相（すなわち円盤と位相同型）を持つことが条件である．

二つのメッシュを合成するという操作は，主に二つの形状間を連続的に補間するモーフィング（*morphing* あるいは *metamorphosis*）で用いられる．画像の補間を基本とする二次元のモーフィングに比べ，三次元の形状を直接扱った三次元モーフィングは，フレームごとに視点が変わるような状況にも対応できることや，色情報やテクスチャ座標をもつような形状に対しても補間可能である，などいくつもの利点を持つ[3, 7].

三次元モーフィングを実現するためには，グラフ構造のような位相的要素，および頂点の位置などの幾何学的要素の異なる二つのメッシュ間に一対一対応を構築する必要がある[1]．この問題を解決する手法として，上記の合成メッシュを用い，各頂点に双方のメッシュ上の座標値を割り当てることで，二つのメッシュ間の対応を構築する方法が一般的である．

筆者らの三次元モーフィングの実現手法 [4] の基本

[1]この問題は一般に対応問題（*correspondence problem*）と呼ばれる [6]．

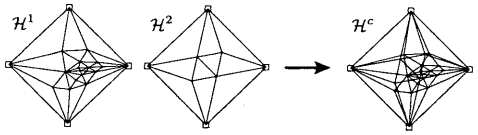

図 1: 埋め込みメッシュの合成

となる考えは，まず頂点数や面数の異なる二つのメッシュを二次元空間上の同じ凸多角形領域内に写像する．写像は調和写像 [2] を求めることにより実現する．この写像されたメッシュを埋め込みメッシュと呼ぶことにする．そして，二つの埋め込みメッシュ同士を合成したもの (これを合成埋め込みメッシュと呼ぶ) をもとに合成メッシュを構築する．

本論文ではそのうち，埋め込みメッシュの合成方法に関して，我々が以前提案した手法 [4] に比べ，処理速度および計算の頑健性が改良されたので，報告するものである．本手法全体の処理は，頂点の数を n とすれば，特に頂点の数が多くなればなるほど $O(n)$ に近い計算量で行なうことができる．また，一般に多角形同士の演算を行なう場合は，数値計算がもたらす誤差などが，アルゴリズムの破綻やグラフ構造の矛盾を引き起こすという問題がある [9] ．本手法でグラフ構造の合成に必要な数値計算は，エッジ間の交差判定だけであり，そこから得られるもとの埋め込みメッシュのグラフ構造の情報を利用しているのが特徴である．

なお本論文では，紙面の都合上，埋め込みメッシュの合成を中心に議論する．三次元モーフィングの実現方法に関しては [4] に基づいているので，こちらの方を参照されたい．

2 埋め込みメッシュの合成手法

図 1 のように，平面上の同じ凸多角形の中に定義されている二つの埋め込みメッシュ $\mathcal{H}^1, \mathcal{H}^2$ を合成し，合成埋め込みメッシュ $\mathcal{H}^c$ を生成するアルゴリズムについて述べる．合成のためのアルゴリズムは: 1. $\mathcal{H}^1, \mathcal{H}^2$ のエッジ間の交点を求める， 2. 各エッジに対し，算出された交点をソートして分割する， 3. 分割された $\mathcal{H}^1, \mathcal{H}^2$ のエッジと頂点，および交点をもとに，$\mathcal{H}^c$ の頂点とエッジ，さらに頂点に隣接するエッジサイクルを求める， 4. $\mathcal{H}^c$ の面を生成する，の四つのステップから構成される．

1.,2. のアルゴリズムは，Kent らの手法 [6] を拡張，改良したものである．Kent らの手法は，球に写像されたメッシュ同士の合成法であり，合成する上で二つの頂点やエッジが一致しない，という仮定のもとでアルゴリズムが設計されている．本アルゴリズムは，同じ凸多角形の中にある $\mathcal{H}^1, \mathcal{H}^2$ を合成するので，境界上の頂点，エッジが必ず一致することを考慮する必要がある．また 3.,4. は独自の考えにもとづいている．

```
procedure エッジ間の交点算出アルゴリズム
( 入力: H^1, H^2 のエッジ, 出力: エッジ間の交点 )
{
  H^1 の全てのエッジのフラッグに NOTUSED をたてる;
  e^1 ← H^1 の先頭のエッジ;
  f^2 ← e^1 の始点の包含面;
  始点に接続しているエッジ群を S^1 にプッシュ,
  それらのエッジのフラッグに USED をたてる;
  while ( S^1 が空でない ) {
    e^1 ← S^1 の先頭のエッジ;
    f^2 を構成するエッジを S^2 にプッシュ;
    while ( S^2 が空でない ) {
      e^2 ← S^2 の先頭のエッジ;
      if ( 交差判定 (e^1, e^2) = TRUE ) {
        交点を求め，双方のエッジの交点リストに加える;
        f^2 ← e^2 に対し f^2 の反対側にある面;
        f^2 の持つエッジのうち e^2 以外を S^2 にプッシュ;
      }
      S^2 をポップ;
    }
    S^1 をポップ;
    e^1 の終点に接続しているエッジのうち, NOTUSED
    のフラッグがたっているエッジを S^1 にプッシュ;
    格納したエッジのフラッグに USED をたてる;
  }
}
```

図 2: エッジ間の交点算出アルゴリズム [6]

2.1 エッジ間の交点算出アルゴリズム

図 2に交点算出アルゴリズムを示す．ここで，S^1 (S^2) は $\mathcal{H}^1$ ($\mathcal{H}^2$) のエッジを格納するためのスタックである．$\mathcal{H}^1$ のエッジは S^1 に格納されたかどうかを判別するフラッグを持つ．フラッグが NOTUSED のときはまだ格納されていない状態を，USED のときは現在格納されているか，すでに格納されたことのある状態を表している．また頂点の包含面とは，$\mathcal{H}^1(\mathcal{H}^2)$ の一つの頂点が含まれる $\mathcal{H}^2(\mathcal{H}^1)$ の面のことをいう．

図 2 のアルゴリズムから，合成に必要な以下の情報: 1. エッジ間の交点の座標値， 2. 交点を境にした面の接続情報， 3. 交点のまわりのエッジサイクル， 4. $\mathcal{H}^1, \mathcal{H}^2$ の頂点の包含面，を得ることができる．このうち 1. は，後述する合成埋め込みメッシュのグラフ構造の構築には関与しない． 2. からは交点のソート (2.2 節)， 3. からはエッジサイクル (ある一つの頂点に接続するエッジを反時計まわりに並べたもの) の生成 (2.3 節)， 4. からは合成メッシュの頂点の座標値の算出，で必要な情報が得られる．

さて，交点算出アルゴリズムにおいて拡張したのは，交差判定のアルゴリズム (図 3) である．このアルゴリズムは，**CASE1** 二つのエッジともに内部エッジ (*internal edge*)，**CASE2** 一方のエッジが境界エッジ (*boundary edge*) でもう一方が内部エッジ，**CASE3** 二つのエッジともに境界エッジ，の三つの場合に分けて交差判定を行ない，TRUE (交差している)，FALSE (交差していない) のいずれかの論理値を返す．

CASE1 の場合は，最大四回の左側判定により交差を判定する．ここで左側判定とは，あるエッジを含

```
Boolean 交差判定( エッジ e^1, エッジ e^2 )
{
  v_s^1 ← e^1 の始点; v_e^1 ← e^1 の終点;
  v_s^2 ← e^2 の始点; v_e^2 ← e^2 の終点;
  switch ( e^1, e^2 ) {
    case ともに内部エッジ: /* CASE1 */
      if ( 左側判定( e^1, v_s^2 ) = 左側判定( e^1, v_e^2 ) )
        return FALSE;
      if ( 左側判定( e^2, v_s^1 ) = 左側判定( e^2, v_e^1 ) )
        return FALSE;
      return TRUE;
    case 一方が境界エッジ, 他方が内部エッジ: /* CASE2 */
      if (二つのエッジが対応頂点対を持つ) return TRUE;
      else {
        e ← e^1, e^2 のうち内部エッジの方;
        v_s, v_e ← 境界エッジの頂点;
        if ( 左側判定( e, v_s ) = 左側判定( e, v_e ) )
          return FALSE;
        return TRUE;
      }
    case ともに境界エッジ: /* CASE3 */
      return FALSE;
  }
}
```

図 3: 二つのエッジの交差判定アルゴリズム

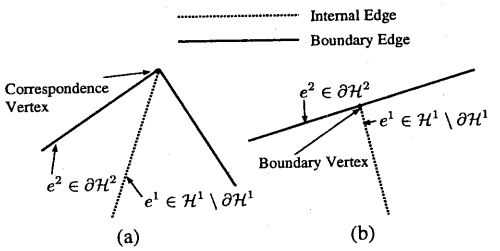

図 4: 境界エッジと内部エッジの交差判定 (a) 対応頂点対を持つ場合 (b) 持たない場合

む直線に対し，頂点が直線の左側にあるかどうかを判定する関数である．エッジ両端の頂点の座標をそれぞれ $(x_s, y_s), (x_e, y_e)$，判定する頂点の座標を (x, y) とすると，

$$(x_e - x_s)(y - y_s) - (x - x_s)(y_e - y_s) > 0, \quad (1)$$

のとき TRUE，そうでないときには FALSE を返す．**CASE2** の場合は，さらに図 4 のように，(a) 二つのエッジが共通の対応頂点 (*correspondence vertex*) を持つ場合，(b) 持たない場合の二つの場合に分けて考える．ここで対応頂点とは，凸多角形の頂点にあたる点のことを呼ぶ．(a) の場合は，必ず交差しているので TRUE を返す．(b) の場合，交差している可能性があるのは，内部エッジのどちらかの頂点が境界頂点になる場合である．この場合，内部エッジに対して，境界エッジの両端の頂点の左側判定を判定するだけでよい．**CASE3** の場合は，交差を調べる必要がないので FALSE を返す．

2.2 交点のソートとエッジの分割

前述の交点算出アルゴリズムから求まった交点を用いた，エッジの分割方法について述べる．分割を行なうには，各エッジの持つ交点を順番に並べるソートが必要になる．ソート操作は，$\mathcal{H}$ の内部エッジと境界エッジとで算出方法が異なる．

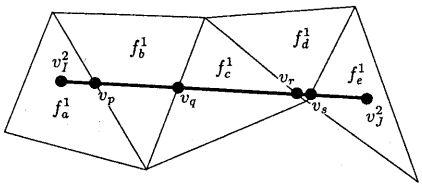

図 5: 内部エッジにおける交点のソート

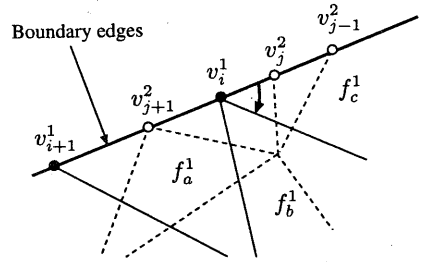

図 6: 境界エッジにおける頂点のソート

内部エッジの場合，交点のソートは次のように考えることができる: 交点が存在するということはすなわち，交点を生成するエッジを挟んで面の情報が変化することを意味する．例えば図 5 のエッジ $\{v_I^2, v_J^2\}$ において，v_I^2 から v_J^2 の方向にエッジ上の点を進んでみると，最初はその点が面 f_a^1 上にあるが，交点 v_p を境に点は f_b^1 上に移る．したがって v_I^2 から v_J^2 に進む過程の中で，面は $f_a^1, f_b^1, f_c^1, f_d^1, f_e^1$ と変化する．このように面の変化をチェックすることで，接続性の情報のみを用いてソートを実現することができる．

実際にはまず，エッジの交点算出アルゴリズム (図 2) において，交点が求まった時点でエッジの両側の面を同時に記憶しておく．各エッジにおける交点のソートを行なうときに，二つの交点同士の面情報を比べ，同じ面を持つ交点を隣に置く，という操作をすべての交点について行なえばよい．この操作は挿入ソート [1] を用いて実現することができる．

境界エッジの場合は，交点のソートは必要ないものの，重なる境界エッジ上の頂点をソートする必要がある．境界上にある頂点のうち，$\mathcal{H}^1$ の頂点を v^1，$\mathcal{H}^2$ の頂点を v^2 とし，$\mathcal{H}$ ごとのそれぞれの頂点群はあらかじめソートされているものとする．また頂点の包含面は 2.1節よりすでに求められている．

ソートの操作は次のようになる．なお，ソートされた頂点を格納するためのリストを一つ用意する．まず，v^2 を順に探索していく．v_k^2 と v_{k+1}^2 に対して包含面を比較すると，

1. v_k^2 と v_{k+1}^2 の包含面が等しければ，両頂点間のエッジ上には v^1 は存在しない

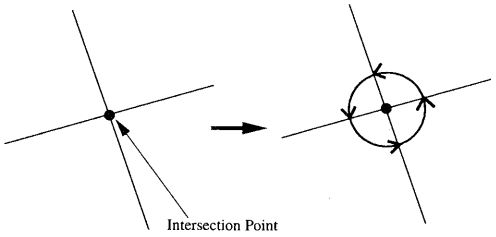

図 7: 交点におけるエッジサイクル

2. v_k^2 と v_{k+1}^2 の包含面が異なるならば，両頂点間のエッジ上には少なくとも一つ以上の v^1 が存在する

ということが言える．1. の場合は v_k^2 をリストに格納し，v^2 の探索を続ける．2. の場合は，エッジ間に存在する v^1 を探す必要があるが，これは v_{k+1}^2 の包含面のエッジを辿ることで見つけることができ，これを v_l^1 としリストに格納する．今度は v_l^1 から順に v^1 を探索し，v_l^1 と同じ包含面を持つならば，その頂点をリストに格納する．なければ v^1 の探索をやめ，v^2 の探索に戻る．この操作を v^2 の探索が終了するまで続ける．

ここで，図 6 を例にとり，上記の手順について具体的に説明する．f_a^1, f_b^1, f_c^1 は実線で囲まれる $\mathcal{H}^1$ の面である．このとき，v_{j-1}^2 と v_j^2 の包含面はともに f_c^1 であるから，両頂点間のエッジには v^1 は存在しない．これに対し，v_{j+1}^2 の包含面は f_a^1 であり，v_j^2, v_{j+1}^2 間のエッジには必ず一つ以上の v^1 が存在することになる．この頂点は，v_j^2 の包含面 f_c^1 のエッジから見つける (v_i^1) ことができる．

2.3 合成埋め込みメッシュの頂点とエッジの生成

$\mathcal{H}^c$ の頂点は，$\mathcal{H}^1, \mathcal{H}^2$ の頂点と，交点算出アルゴリズムにより求めた交点から構成される．また，$\mathcal{H}^c$ のエッジは，ソートされた交点を用いて分割された $\mathcal{H}^1, \mathcal{H}^2$ のエッジから構成される．

$\mathcal{H}^c$ の面の生成に必要なエッジサイクルは，頂点を次の二つの場合に分けて考えることで求めることができる：$\mathcal{H}^c$ の頂点のうち，$\mathcal{H}^1$, $\mathcal{H}^2$ の頂点から生成されるものは，もとの $\mathcal{H}^1$, $\mathcal{H}^2$ の頂点まわりのエッジサイクルを継承すればよい．また，交点から生成されるものに関しては図 7 のように処理することができる．すなわち，エッジの交点算出アルゴリズム (図 2) において，交点が求まった時点でその交点のまわりのエッジサイクルは四本のエッジから構成されることがわかる．

対応頂点におけるエッジサイクルはこの時点では求まっていないが，$\mathcal{H}^c$ の面の生成には必要ないので，特に求める必要はない．

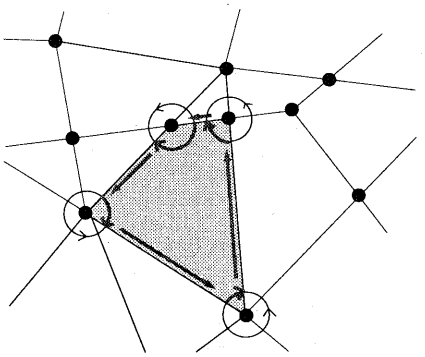
図 8: $\mathcal{H}^c$ の面の生成

2.4 合成埋め込みメッシュの面の生成

$\mathcal{H}^c$ の頂点とエッジ，それと頂点の持つエッジサイクルから，$\mathcal{H}^c$ の面は以下のように生成できる (図 8) : まず，ある頂点から出発して頂点のまわりのエッジのうちの一つに進む．そしてそのエッジのもう一方の頂点にきたら，頂点のまわりのエッジサイクルを逆方向に辿り，一つ前のエッジに進む．その操作を繰り返し，最初に出発した頂点に辿りついたところで，面を構成する頂点サイクルを生成する．生成された $\mathcal{H}^c$ の面には，三角形にならない面もあるので，最後に，このような面について 三角形分割を行なう．

3 合成のための数値計算処理

四つのステップを通じて，埋め込みメッシュからの合成埋め込みメッシュの生成，特にグラフ構造の構築のための数値計算は，交点算出アルゴリズムの中の二つのエッジの交差判定と，アルゴリズムの最初の e^1 の始点に対する包含面 f^2 を求める処理の二つである．このうち，問題なのは合成埋め込みメッシュのグラフ構造の構築にかかわる前者の処理の方である．そして，この処理に対しては整数帰着法と記号摂動法 [9] を用いた．

整数帰着法において，交差判定に必要な数値演算は左側判定に用いる式 (1) のみであるから，式 (1) に用いる埋め込みメッシュ $\mathcal{H}$ の頂点のもつ座標値 $\boldsymbol{v}$ を整数化して演算を行なう．整数演算に用いる座標値の最大値を L とすれば，$2L \times 2L + 2L \times 2L \leq 8L^2$ 以下の整数が溢れないだけの座標値を確保すればよい．これは $\boldsymbol{v}$ がすべて単位円の中に収まっていることを考えると容易なことである[2]．

記号摂動法は，エッジの交差において退化 (一方のエッジ上にもう一方のエッジの両端にある頂点がのっている場合を言う) が起こったときに対する矛盾を回避するための手法で，退化が検知されたときに座標値に摂動 (ゆらぎ) を与え，退化が起こっていないもの

[2]一つの座標値につき 64 ビット確保できれば十分である．これは，通常のワークステーションでは long 型に相当する．

として扱うというものである．この方法に関しては文献[9]と同一の手法を用いているので，文献の方を参照されたい．

4 計算量についての考察

本手法の中で計算量を考慮すべき操作は，1. 交点算出アルゴリズム，2. 交点のソート，3. 合成埋め込みメッシュの頂点，エッジの生成，それに4. 合成埋め込みメッシュの面の生成である．

1. の交点算出アルゴリズムでは，$\mathcal{H}^1$ の一つのエッジの交点の探索につき，探索する $\mathcal{H}^2$ のエッジの数は，最悪で (各エッジにつき求まった交点の数) × 3 程度である．$\mathcal{H}^1$ のエッジの数を n_e，一つのエッジが持つ交点の数を n_{int} とすると，すべての $\mathcal{H}^1$ のエッジに対する交点の探索は $O(n_e n_{int})$ 回かかる．2. の $\mathcal{H}^1$ のエッジの交点は，アルゴリズムの性質上，すでにソートされた状態で出力されているので，上記のソートの操作は $\mathcal{H}^2$ のエッジを一回探索すればよい．各エッジの交点のソートにかかる時間は，$\mathcal{H}^2$ のエッジの数を n_e，一つのエッジが持つ交点の数を n_{int} とすると，交点の挿入ソートの操作が $O((n_{int})^2)$ だけかかるので，このステップ全体の計算は $O(n_e(n_{int})^2)$ となる．3. の $\mathcal{H}^c$ の頂点の生成には，$\mathcal{H}^1, \mathcal{H}^2$ の頂点の一回の探索と，エッジ間のすべての交点の一回の探索のみでよい．よって，$O(n_v^1 + n_v^2 + n_v^1 n_{int}^1)$ となる．また，$\mathcal{H}^c$ のエッジの生成は，$\mathcal{H}^1$, $\mathcal{H}^2$ のエッジの一回の探索と，各エッジにつき交点の一回の探索でよいので，$O(n_{int}^1 n_e^1 + n_{int}^2 n_e^2)$ となる．4. の操作はすべての頂点を一度探索すればよいので，$\mathcal{H}^c$ の頂点の数を n^c とすれば，$O(n^c)$ の時間で実行できる．

1.-3. は，計算量はエッジ間の交点の数に依存する．しかし実用的には，エッジの数が多くなるほど，エッジの総数に対して一つのエッジの持つ交点の数は無視できるほど小さくなる，と考えられる．総じて本手法は，頂点，エッジの数が多くなるほど，全体の計算量は $O(n)$ に近くなる，ということが言える．

5 適用結果

本手法をいくつかの例題に適用した結果を示す．図9 に，二つの顔のモデル (Moser, Kanai) に対して合成メッシュを生成した結果を示す．図9(a) のうち球で表される頂点が対応頂点である．これらの頂点は，例えば顎の付近の頂点同士というように，顔の特徴部分の頂点を対応づけしたものである．図9(b) に生成した二つの合成メッシュを示す．さらに，図9(c) に埋め込みメッシュを，図9(d) に合成埋め込みメッシュを示す．これらの形状を用いて合成のための計算を行なってはいるものの，あくまで内部の構造を表すためのものである．従って実装の段階で表示する必要はない．

	Moser	Kanai	Golf	Porsche
頂点	523	384	2,974	1,955
エッジ	1,514	1,101	8,771	5,719
面	992	718	5,798	3,765
オイラー数	1	1	1	1
CV	8		10	
頂点 (合成)	3,612		15,174	
エッジ (合成)	10,738		45,249	
面 (合成)	7,127		30,076	
オイラー数	1		1	
調和写像 (Sec.)	0.15	0.10	1.48	1.17
合成 (Sec.)	0.60		3.23	

表 1: 例題におけるメッシュの要素数と計算時間

また表1に，上記の例題と，より多い面の数を持つ二つの車のメッシュ (Golf, Porsche) を合成した場合のメッシュ要素数や計算時間等を示す．ここでいうメッシュ要素数とは，もとのメッシュと合成メッシュの頂点数，エッジ数，面数，対応頂点 (CV) 数のことを指す．また，もとのメッシュから埋め込みメッシュの生成 (調和写像) ，埋め込みメッシュから合成埋め込みメッシュの合成に費した時間を，MIPS R10000 CPU 175MHz で計測し，秒単位で示している．特に車のモデルの例を見ると，おおよそ6秒程で合成のためのすべての計算を行なうことができ，実用的には計算効率のよい手法であることがわかる．さらに，計算結果の位相的な妥当性を示すために，表にはオイラー数[8] を載せている．円盤と位相同型な形状では，オイラー数が必ず1になる．表から合成後の形状もそのオイラー数が保存されているのがわかる．

図10に，得られた合成メッシュをモーフィングに適用した結果を示す．モーフィングにおいては，ユーザによるインタラクティブな対応づけの制御ができることが望ましいとされている[7]．本手法は，計算速度の点から考えれば，十分にインタラクティブな処理の可能な方法であるということが言える．

6 おわりに

本論文では，円盤と位相同型である任意の三角形メッシュを合成するための新しい手法を提案した．本手法の中での，合成のためのアルゴリズムは頑健であり，面の多いときには $O(n)$ に近い計算量で処理することが可能であることを示した．

本手法は，円盤に位相同型という位相的には限られたメッシュを対象としており，境界のみの対応点の指定だけでは，モーフィングにおける制御には (主に質の面から考えると) 十分ではない，と思われる．ただ，[5] では，同位相の任意のメッシュに対し，ユーザによって与えれた頂点間対応づけを満たすよう，メッシュを分割して処理する方法を提案している．この方法に，本手法を直接適用することが可能であることを確認している．

なお，以下のホームページに三次元形状モーフィングに関する情報を集めているので，是非参照されたい:

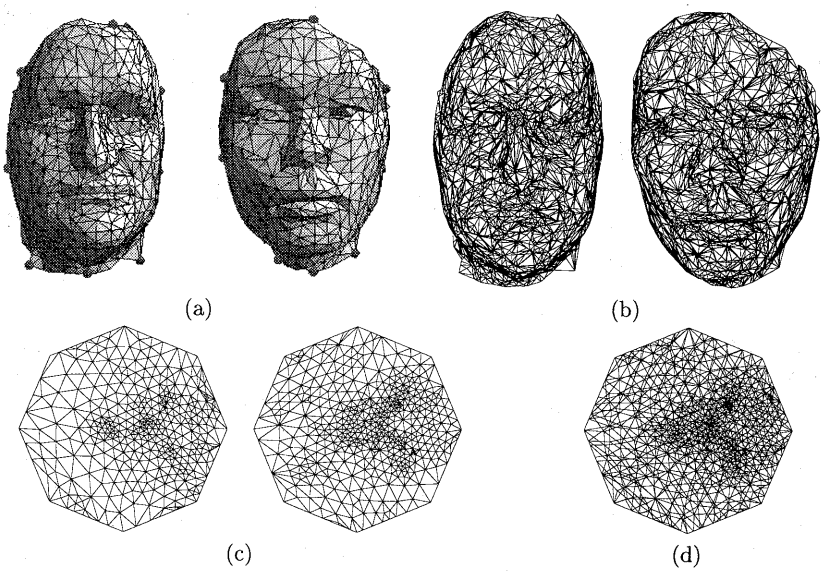

図 9: 二つの顔モデル (Moser, Kanai) に対する合成メッシュ生成結果. (a) もとのメッシュと 8 個の対応頂点. (b) 合成メッシュ. (c) 埋め込みメッシュ. (d) 合成埋め込みメッシュ.

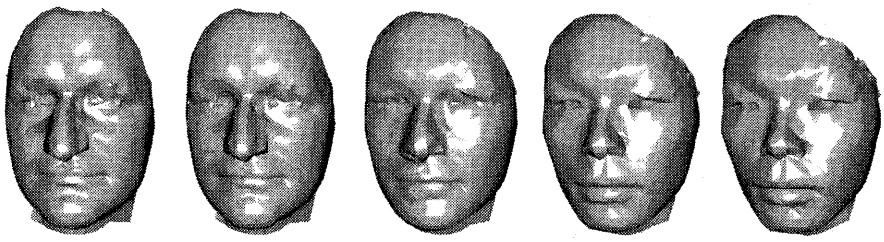

図 10: 合成メッシュを用いた顔のモーフィング

http://www.riken.go.jp/lab-www/mat-fab/personal/kanai/Gmorph.html

謝辞

例題に使用した二つの顔のモデル (Moser, Kanai) は, 米 Cyberware 社の 3D デジタイザにより計測した点群データから構築したモデルである. 装置を使用するにあたり, 東大工学部電気電子工学科原島・金子研の御協力を得た. また, この研究の一部は (財) 大川情報通信基金の支援を受けた. 併せてここに感謝の意を表する.

参考文献

[1] AHO, A. V., HOPCROFT, J. E. and ULLMAN, J. D. データ構造とアルゴリズム (邦訳), 培風館 (1987).

[2] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M. and STUETZLE, W. Multiresolution Analysis of Arbitrary Meshes, Computer Graphics (Proc. SIGGRAPH 95), ACM Press, New York (1995).

[3] GOMES, J., DARSA, L., COSTA, B. and VELHO, L. *Warping and Morphing of Graphical Objects*, Morgan Kaufmann, San Francisco, Calif. (1998).

[4] KANAI, T., SUZUKI, H. and KIMURA, F. Three-dimensional Geometric Metamorphosis Based on Harmonic Maps, *The Visual Computer*, **14**, 4 (1998), 166–176.

[5] KANAI, T., SUZUKI, H. and KIMURA, F. Metamorphosis of Arbitrary Triangular Meshes with User-specified Correspondence, *IEEE Computer Graphics and Applications* (to appear).

[6] KENT, J. R., CARLSON, W. E. and PARENT, R. E. Shape transformation for polyhedral objects, Computer Graphics (Proc. SIGGRAPH 92), ACM Press, New York (1992).

[7] LAZARUS, F. and VERROUST, A. Three-dimensional metamorphosis: a survey, *The Visual Computer*, **14**, 8/9 (1998), 373–389.

[8] MÄNTYLÄ, M. *An Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland (1988).

[9] 杉原厚吉 計算幾何工学, 培風館 (1994).