円盤と位相同型な任意の三角形メッシュ間の合成法

金井 崇[†] 鈴木宏正[††] 木村文彦[††]

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新しい手法について提案する．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よく用いられる．これまでの手法では，対象とする形状の幾何学的性質が限定されてしまう，という問題点があった．本手法は，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．このことが，ユーザが三次元モーフィングを扱ううえで利点になることを，例題を通じて検証する．また，本手法の中の合成のためのアルゴリズムは頑健であり，また特に面の数が多い場合には高速なアルゴリズムであることも示している．

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

TAKASHI KANAI,[†] HIROMASA SUZUKI[††] and FUMIHIKO KIMURA[††]

This paper proposes a new method of combining two triangular meshes which are generally used in the Comuputer Graphics area. The operation of combining meshes are frequently used for the three-dimensional (3D) morphing. In the previous research, there is a problem that geometric properties of meshes are limited. Our method are based on expanding 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. We show that this characteristic can be merit for the 3D morphing. We also show that our algorithm for combining is numerically robust, and is fast especially for the case that the number of faces are 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2つのメッシュを合成するための新しい手法を提案する．ここでいう“合成”というのは，同じ位相を持つ2つのメッシュのグラフ構造をあわせ持つような，1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（*a combined mesh*）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．

2つのメッシュを合成するという操作は，主に2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[2)~4)]に比べ，三次元の形状を直接扱った三次元モーフィング[5)~8)]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持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がある．

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2つのメッシュ間に1対1対応を構築する必要がある☆☆．この問題を解決する1つの手法として，上記の合成メッシュを用い，合成メッシュの頂点に双方のメッシュ上の座標値を割り当てることで2つのメッシュ間の対応を構築できる．

合成メッシュの生成手法として，たとえばKentら

† 理化学研究所基礎科学特別研究員
Special Postdoctoral Researcher, The Institute of Physical and Chemical Research (RIKEN)
†† 東京大学大学院工学系研究科
Graduate School of Engineering, University of Tokyo

☆ ここでいう合成メッシュのことを，位相幾何学の分野では共通細分[1)]と呼んでいる．

☆☆ この問題は一般に対応問題（*correspondence problem*）と呼ばれる[5)]．

は，2つのメッシュを球に投影し合成する手法を示した[5]．しかし，対象形状が星型形状や回転体，スイープ立体というような，メッシュ全体の持つ幾何学的性質に制限を持つという問題があった．

本論文で述べる合成法[9]の基本となる考えは，Kentらの手法と同様，頂点数や面数の異なる2つの三角形メッシュを，ある1つの同じ形状に写像することである．本手法は，調和写像[10]を用いることで，2つのメッシュを二次元空間上の単位円上の同じ多角形内に写像する．写像によって得られた多角形どうしを合成することにより，合成メッシュを生成している．これにより，円盤と位相同型であるという位相的には限られた形状を対象にしているが，メッシュの凹凸などの幾何学的性質には制限がない．

本論文は，以前提案した合成法[9]よりも，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 $O(n)$ に近い計算量で行うことができる．また，一般に多角形どうしの演算を行う場合は，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．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．

2. 手法の概要

まず，本論文で使用するための用語について定義する．メッシュ $\mathcal{F}$ は，頂点 v_i $(i = 1 \ldots n)$，エッジ e_j $(j = 1 \ldots m)$，三角形面 f_k $(k = 1 \ldots l)$ のグラフ構造をなす．本論文で対象とするのは，円盤と位相同型のメッシュであるから，メッシュはただ1つの境界 $\partial\mathcal{F}$ を持つ．境界は隣接する面を1つしか持たないエッジ群からなる閉ループである．ここでは境界上にあるエッジ，頂点をそれぞれ境界エッジ（*boundary edge*），境界頂点（*boundary vertex*），それ以外のエッジ，頂点をそれぞれ内部エッジ（*internal edge*），内部頂点（*internal vertex*）と呼ぶことにする．また，それぞれの頂点 $v_i \in \mathcal{F}$ は，三次元空間上の座標値 $\mathbf{v}_i$ を持つ．

図1に，2つのメッシュ $\mathcal{F}^1$，$\mathcal{F}^2$ から合成メッシュ $\mathcal{F}^c$ を生成する手法の概要を示す．本手法は大きく分けて次の2つのステップからなる：

(1) まず，境界上の頂点の対応づけを行う．すなわち，2つのメッシュの境界から1つずつ頂点を選ぶ．これを対応頂点対（*correspondence vertex pair, CVP*），選ばれた頂点を対応頂点（*correspondence vertex, CV*）と呼ぶことにする．この対応頂点対の作成は，境界上の3つ以上の頂点に対して行う．次に，$\mathcal{F}^1$，$\mathcal{F}^2$ を同じ二次元平面上の凸多角形上に展開し，埋め込みメッシュ（*embedding mesh*）$\mathcal{H}^1$，$\mathcal{H}^2$ を作る．$\mathcal{H}^1$，$\mathcal{H}^2$ は，それぞれ $\mathcal{F}^1$，$\mathcal{F}^2$ と同じグラフ構造を持ち，頂点が持つ座標値のみが異なる．ここで，頂点 $v_i \in \mathcal{H}$ $(i = 1 \ldots n)$ が持つ二次元平面上の座標値を $\boldsymbol{v}_i$ とする．

(2) 2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（*combined embedding mesh*）$\mathcal{H}^c$ を生成する．$\mathcal{H}^c$ は $\mathcal{F}^c$ と同じグラフ構造を持ち，座標値のみが異なる．最後に $\mathcal{F}^c$ の頂点の座標値を求める．

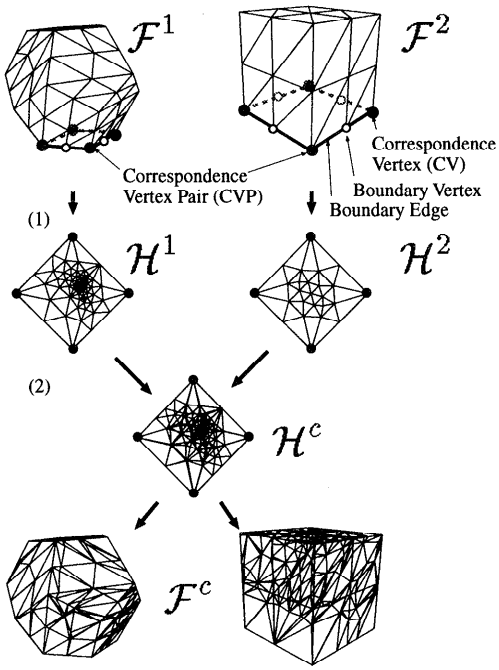

図1 合成手法の概要
Fig. 1 An overview of our combining method.

3. 写像に基づく埋め込みメッシュの生成

2つのメッシュ $\mathcal{F}^1$，$\mathcal{F}^2$ を，二次元空間の単位円上の凸多角形領域 $\mathcal{H} \subseteq \mathbf{R}^2$ に展開する方法について示す．本研究では，境界写像 g と調和写像 h の2つの写像によって埋め込みメッシュ $\mathcal{H}^1$，$\mathcal{H}^2$ を生成する．

3.1 境界写像 g による境界頂点の座標値の算出

$\mathcal{F}$ の境界 $\partial\mathcal{F}$ を $\mathcal{H}$ の境界 $\partial\mathcal{H}$ に写す境界写像 $g : \partial\mathcal{F} \mapsto \partial\mathcal{H}$ は，次のようにして実現する．$\mathcal{F}$ の境界上の頂点のうち n 個の対応頂点を，二次元空間上の原点を中心とする単位円上に置かれた正 n 角形の頂点に写像する．残りの境界頂点は正 n 角形の辺上

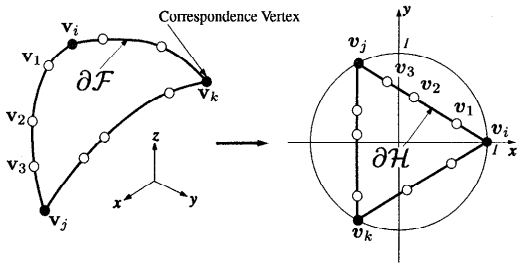

図 2 境界写像 g
Fig. 2 Boundary map g.

に写されることになるが，このとき頂点間のエッジの長さの比に等しくなるように配置する．

図 2 に，境界写像 g の決定方法について示す．この例では，$\mathcal{F}$ の境界セグメントを構成する頂点のうち 3 点 $\{v_i, v_j, v_k\} \in \mathcal{F}$（図の●印の頂点）がまず二次元空間上の原点を重心とする正三角形の頂点 $\{v_i, v_j, v_k\} \in \mathcal{H}$ にそれぞれ写される．次に残りの頂点（図の○印の頂点）は，正三角形上の辺上に写される．このとき，

$$|\mathbf{v}_i\mathbf{v}_1| : |\mathbf{v}_1\mathbf{v}_2| : |\mathbf{v}_2\mathbf{v}_3| : |\mathbf{v}_3\mathbf{v}_j| = |\boldsymbol{v}_i\boldsymbol{v}_1| : |\boldsymbol{v}_1\boldsymbol{v}_2| : |\boldsymbol{v}_2\boldsymbol{v}_3| : |\boldsymbol{v}_3\boldsymbol{v}_j|,$$

という関係が成り立つように，$\boldsymbol{v}_1$，$\boldsymbol{v}_2$，$\boldsymbol{v}_3$ を求める．ここで，$|\mathbf{v}_i\mathbf{v}_1|\ldots$ は 2 つの頂点間のユークリッド距離を表す．

3.2 調和写像 h による内部頂点の座標値の算出

次に $\mathcal{F}$ の内部 $\mathcal{F} \setminus \partial\mathcal{F}$ を $\mathcal{H}$ の内部 $\mathcal{H} \setminus \partial\mathcal{H}$ に写す調和写像（*harmonic map*）$h : \mathcal{F} \setminus \partial\mathcal{F} \mapsto \mathcal{H} \setminus \partial\mathcal{H}$ について説明する．調和写像 h は，位相的円盤 $\mathcal{F}$ の平面グラフへの埋め込みを実現する写像の 1 つで，計量分散☆を最小にするという性質を持つ[11]．

調和写像に基づく埋め込みメッシュの座標値の算出法であるが，Eck らの手法[10] をベースとして用いる．Eck らの手法は，厳密にいうと調和写像の線形近似手法である（Maillot らも同様の手法を提案している[12] が非線形問題を解く必要がある）が，座標値を頑健かつ高速に計算することができる．

$\mathcal{F}$ の内部の頂点の $\mathcal{H}$ 上での位置は，隣接するエッジにおいて定義される関数 E が最小になるように計算する．$\mathcal{H}$ 上のエッジの両端の頂点を $\boldsymbol{v}_i$ などとして表すと，E は，

$$E = 1/2 \sum_{\{v_i, v_j\} \in \mathcal{H}} \kappa_{i,j} |\boldsymbol{v}_i - \boldsymbol{v}_j|^2, \qquad (1)$$

となる．バネ定数 $\kappa_{i,j}$ は次のようにして計算される：

$$\kappa_{i,j} = ({l_{i,k_1}}^2 + {l_{j,k_1}}^2 - {l_{i,j}}^2)/A_{i,j,k_1} + ({l_{i,k_2}}^2 + {l_{j,k_2}}^2 - {l_{i,j}}^2)/A_{i,j,k_2}. \qquad (2)$$

ここで，$l_{i,j}\ldots$ はエッジ $e_{i,j}$ の端点 v_i，v_j 間のユークリッド距離，$A_{i,j,k_1},\ldots$ は頂点 $v_i, v_j, v_k \in \mathcal{F}$ で囲まれる面の面積である．境界エッジに関係するバネ定数の公式は 1 つの項のみを持つ．E は頂点 $\boldsymbol{v}$ に関する正の定符号の二次関数となり，唯一の最小解を持つ．最小値を与える $\boldsymbol{v}$ は，E の $\boldsymbol{v}$ に関する勾配から，方程式 $\nabla E = \mathbf{0}$ を解くことで得られる．E の勾配を求めるには，$\boldsymbol{v}$ の x，y 成分を並べて，

$$\mathbf{V} \equiv (v_1.x,\ v_1.y,\ \ldots,\ v_N.x,\ v_N.y), \qquad (3)$$

をつくる．ここで N は頂点数である．E は $\boldsymbol{v}$ の x，y 成分の二次形式であるから，$E = \mathbf{V}^T\mathbf{H}\mathbf{V}$ という形式で表現でき，その勾配は $\nabla E = \partial E/\partial \mathbf{V}$ となる．

さらに，境界上の頂点は固定するため，そのような頂点に関する変数は定数となる．したがって，このエネルギー関数の変数 $\mathbf{V}$ は，未決定変数部 $\mathbf{V}^h$ と固定変数部 $\mathbf{V}^g$ に分割することができる．定数行列 $\mathbf{H}$ もそれに合わせて分割してやれば，エネルギー関数 E は以下のように表せる：

$$E = \begin{bmatrix} \mathbf{V}^{h^T} & \mathbf{V}^{g^T} \end{bmatrix} \begin{bmatrix} \mathbf{H}^{hh} & \mathbf{H}^{hg} \\ \mathbf{H}^{gh} & \mathbf{H}^{gg} \end{bmatrix} \begin{bmatrix} \mathbf{V}^h \\ \mathbf{V}^g \end{bmatrix}. \qquad (4)$$

固定された点に関しては，エネルギー関数は定数となるので，結局，E を最小化するには，固定されていない点列 $\mathbf{V}^h$ に関する勾配 ∇E を求めて，次の連立一次方程式を解けばよい：

$$\nabla E = \frac{\partial E}{\partial \mathbf{V}^h} = 2\mathbf{H}^{hh}\mathbf{V}^h + 2\mathbf{H}^{hg}\mathbf{V}^g = \mathbf{0}. \qquad (5)$$

この連立一次方程式 (5) のうち，係数行列 $\mathbf{H}^{hh}$ の各行に注目すると，非零の項は，対応する頂点とその頂点に接続するエッジを挟んだ頂点のみであり，残りの項はすべてゼロの値となる．一般にこのような頂点の数は，すべての頂点数に比べて小さく，したがって式 (5) は疎の連立一次方程式となる．

4. 埋め込みメッシュの合成による合成メッシュの生成

本章では，前章で生成された埋め込みメッシュを合成し，得られる合成埋め込みメッシュをもとに合成メッシュを生成するためのアルゴリズムを示す．合成のためのアルゴリズムは，次の 5 つのステップから構成される：

(1) $\mathcal{H}^1$，$\mathcal{H}^2$ のエッジ間の交点を求める（4.1 節）．
(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（4.2 節）．

☆ $\mathcal{F}$ の小さい直径の領域が写像によってどれだけ伸びるかを計る尺度，すなわち計量的な歪みの尺度を意味する．

(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（4.3 節）．

(4) $\mathcal{H}^c$ の面を生成する（4.4 節）．

(5) $\mathcal{F}^c$ の頂点の座標値を算出する（4.5 節）．

ここで提案するアルゴリズムは，Kent らの手法[5]を拡張，改良したものである．ただ，Kent らの手法は，球に写像されたメッシュどうしの合成法であり，合成するうえで 2 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．このことより (1), (2) を拡張している．また (3) 以降は独自の考えに基づいている．

合成するうえで問題になることとして，Kent らも指摘しているように，合成するメッシュの面の数が多くなると，数値計算がもたらす不安定性によって，アルゴリズムが破綻したり，合成メッシュの位相がおかしくなったりする可能性があることである．本アルゴリズムの特徴は，このような場合に対処するために，エッジ間交点算出アルゴリズム（上記の (1)）によって得られる埋め込みメッシュのグラフ構造に関する情報のみを用いて，合成埋め込みメッシュのグラフ構造を構築していることである．

4.1 エッジ間の交点算出アルゴリズム

最初のステップとして，2 つの埋め込みメッシュ $\mathcal{H}^1$, $\mathcal{H}^2$ 間のエッジの交点を算出する．交点算出アルゴリズムは基本的には Kent らの手法[5]を用いる．**図 3** にその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グを持つ．フラグが 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 1 つの頂点が含まれる $\mathcal{H}^2(\mathcal{H}^1)$ の面のことをいう．このアルゴリズムは，以下の 3 つの利点を持つ：

- $\mathcal{H}^1$ のエッジに対し，交点があらかじめソートされた状態で得られるので，後述する交点のソートは $\mathcal{H}^2$ のエッジにのみ行えばよい．
- 交点と同時に $\mathcal{H}^1$, $\mathcal{H}^2$ の頂点の包含面を求めることができる．
- 高速である．

図 3 の交点算出アルゴリズムから，合成に必要な以下の情報を得ることができる：

```
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 のすべてのエッジのフラグに 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グに USED をたてる;
  }
}
```

図 3 エッジ間の交点算出アルゴリズム[5]
Fig. 3 An edge-edge interseciton algorithm[5].

(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

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（4.2 節），(3) からはエッジサイクル（ある 1 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（4.3 節），(4) からは $\mathcal{F}^c$ の頂点の座標値の算出（4.5 節）で必要な埋め込みメッシュのグラフ構造に関する情報が得られる．

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 4**）である．このアルゴリズムは，

CASE1 2 つのエッジともに内部エッジ

CASE2 一方のエッジが境界エッジ，もう一方が内部エッジ

CASE3 2 つのエッジともに境界エッジ

の 3 つの場合に分けて交差判定を行い，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 4 回の左側判定により交差を判定する．一方のエッジに対してもう一方のエッジの両端の頂点を判定し，両方とも同じ論理値であれ

```
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 (2 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 4 2 つのエッジの交差判定アルゴリズム
Fig. 4 A judgement algorithm of intersection between two edges.

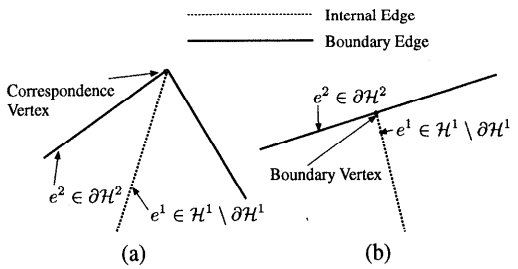

図 5 境界エッジと内部エッジの交差判定. (a) 対応頂点対を持つ場合. (b) 持たない場合
Fig. 5 Judgements of intersection between a boundary edge and an internal edge: (a) The case which edges have common CVs. (b) The case which edges don't have common CVs.

ば交差の可能性はないので FALSE を返す.

ここで左側判定とは, あるエッジを含む直線に対し, 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 v_s, v_e の座標をそれぞれ (x_s, y_s), (x_e, y_e), 判定する頂点 v_t の座標を (x_t, y_t) とすると,

$$(x_e - x_s)(y_t - y_s) - (x_t - x_s)(y_e - y_s) > 0, \tag{6}$$

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 5 のように, (a) 2 つのエッジが共通の対応頂点を持つ場合, (b) 持たない場合の 2 つの場合に分けて考える. (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.

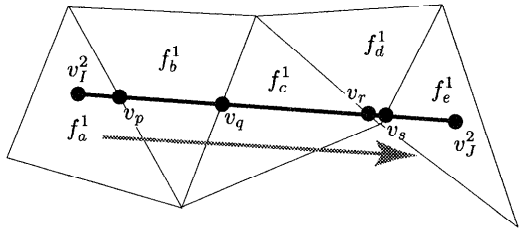

図 6 内部エッジにおける交点のソート
Fig. 6 A sort operation of intersection points in an internal edge.

CASE3 の場合は交差を調べる必要がないので, FALSE を返す.

4.2 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.

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. たとえば, 図 6 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 3) 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うときに, 2 つの交点どうしの面情報を比べ, 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えばよい. この操作は挿入ソート[13] 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

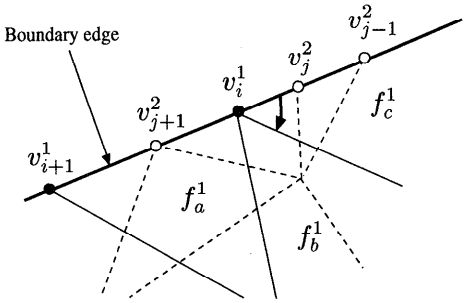

図 7 境界エッジにおける頂点のソート
Fig. 7 A sort operation of vertices in boundary edges.

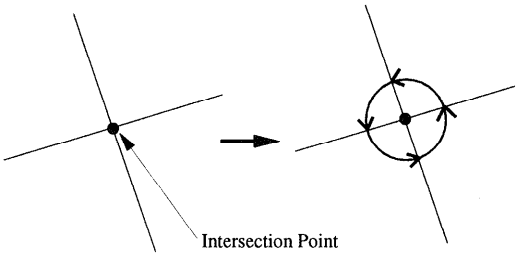

図 8 交点におけるエッジサイクル
Fig. 8 An edge cycle of an intersection point.

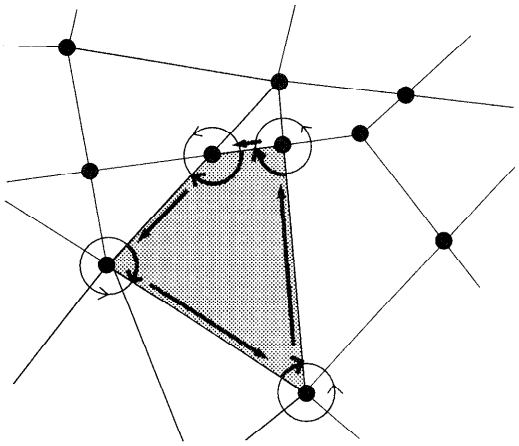

図 9 $\mathcal{H}^c$ の面の生成
Fig. 9 Creation of the faces of $\mathcal{H}^c$.

面は 4.1 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 1 つ用意する．まず，v^2 を順に探索していく．v^2_k と v^2_{k+1} に対して包含面を比較すると，

(1) v^2_k と v^2_{k+1} 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．

(2) v^2_k と v^2_{k+1} 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 1 つ以上の v^1 が存在する．

ということがいえる．(1) の場合は v^2_k 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^2_{k+1} の包含面のエッジをたどることで見つけることができ，これを v^1_l としリストに格納する．今度は v^1_l から順に v^1 を探索し，v^1_l 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．

図 7 を例にとって具体的に説明する．ここで，f^1_a，f^1_b，f^1_c は，実線で囲まれる $\mathcal{H}^1$ の面である．このとき，v^2_{j-1} と v^2_j の包含面はともに f^1_c 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^2_{j+1} の包含面は f^1_a であり，v^2_j，v^2_{j+1} 間のエッジには，必ず 1 つ以上の v^1 が存在することになる．この頂点は，v^2_j の包含面 f^1_c のエッジから見つける (v^1_i) ことができる．

4.3 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 2 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 8** 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 3) 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 4 本のエッジから構成されることが分かる．

実際には，交点が求まると同時に，4 つのエッジからなるエッジサイクルの情報をも記憶しておく．そして $\mathcal{H}^c$ のエッジが生成された時点で，随時，$\mathcal{H}^c$ の頂点が持つエッジサイクルの情報に対して生成されたエッジを入力していけばよい．

ただし，凸多角形上の頂点のエッジサイクルはこの時点では求まっていないが，これらの頂点以外のエッジサイクルは求まっている．$\mathcal{H}^c$ の面の生成には必要ないので，特に求める必要はない．

4.4 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 9**)：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 1 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たどり，1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たどりついたところで，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う．

4.5 合成メッシュの生成

合成埋め込みメッシュ $\mathcal{H}^c$ と合成メッシュ $\mathcal{F}^c$ のグラフ構造は同じであるから，あとは合成メッシュの頂点が持つ三次元空間上の座標値を求める必要がある．この座標値は，$\mathcal{F}^1$，$\mathcal{F}^2$ のそれぞれ面上の1点に対応した2通りの値を持つ．以下では，例として $\mathcal{F}^1$ に対応する座標値を求める．求めるべき座標値は：

- $\mathcal{H}^2$ ($\mathcal{F}^2$) から生成された頂点の $\mathcal{F}^1$ に対応する座標値
- $\mathcal{H}^1$，$\mathcal{H}^2$ のエッジ間の交点から生成された頂点の $\mathcal{F}^1$ に対応する座標値

である．これらの値は包含面の重心座標値から求める．v^c の包含面を $f^1=\{v^1_\alpha, v^1_\beta, v^1_\gamma\}$ とし，その重心座標を (α,β,γ)，$\alpha+\beta+\gamma=1$ とすると，$\mathbf{v}^1$ は次の式で計算される：

$$\mathbf{v}^1=\alpha\mathbf{v}^2_\alpha+\beta\mathbf{v}^2_\beta+\gamma\mathbf{v}^2_\gamma. \tag{7}$$

4.6 合成のための数値計算処理

以上の5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2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2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[14]を用いた．

整数帰着法は，位相構造の判定に用いる計算をすべて整数の演算に帰着させる方法で，矛盾の発生を防ごうとするものである．交差判定に必要な数値演算は左側判定に用いる式(6)のみであるから，式(6)に用いる合成メッシュ $\mathcal{H}$ の頂点の持つ座標値 $\boldsymbol{v}$ を整数化して演算を行う．整数演算に用いる座標値の最大値を L とすれば，$2L\times 2L+2L\times 2L\le 8L^2$ 以下の整数があふ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☆．

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いう）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として扱うというものである．実際には，式(6)の代わりに以下の(1)〜(8)を順に実行する[14]：

(1) v_s, v_e, v_t を添え字の小さい順に並べる．これを v_i, v_j, v_k $(i<j<k)$ とする．

(2) $\begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{vmatrix} \neq 0$ ならばその符号を z として(7)へ．

(3) $y_j \neq y_k$ ならば $z=\mathrm{sign}(y_j-y_k)$ として(7)へ．

(4) $y_i \neq y_k$ ならば $z=\mathrm{sign}(y_k-y_i)$ として(7)へ．

(5) $x_j \neq x_k$ ならば $z=\mathrm{sign}(x_k-x_j)$ として(7)へ．

(6) $z=-1$ として(7)へ．

(7) (s,e,t) を (i,j,k) と移す置換が偶置換ならば $z'=z$ とし，奇置換ならば $z'=-z$ とする．

(8) $z'>0$ のとき **TRUE**，そうでないときには **FALSE** を返す．

ここで $\mathrm{sign}(x)$ は x の符号を表す．

ここで述べた記号摂動法の代わりに，たとえば退化の状態が起こったときに，エッジの上に乗っている頂点を移動し，一致しているものとして扱う方法が考えられる．ただ，メッシュの位相を保存しなければならないといった移動に関する付加条件がつくので，アルゴリズムがやや複雑になるのと，後のソートやエッジサイクルの生成の際のアルゴリズムの修正が必要となる．

5. 計算量についての考察

本章では，3.4節で提案した合成メッシュ生成手法全体に必要な計算量について評価し，考察を行う．手法の中で計算量を考慮すべき操作は，(1) 調和写像による埋め込みメッシュの内部頂点の座標値の算出（3.2節），(2) 交点算出アルゴリズム（4.1節），(3) 交点のソート（4.2節），(4) 合成埋め込みメッシュの頂点，エッジの生成（4.3節），(5) 合成埋め込みメッシュの面の生成（4.4節），である．

(1) 3.2節で述べたように，式(5)は疎の連立一次方程式となる．我々はこの式を解くために双共役勾配法[15]を用いた．詳細な説明は文献15)に譲るが，この方法を用いると，主な計算は $\mathbf{H}^{hh}\mathbf{V}^h$ と

☆ 1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．

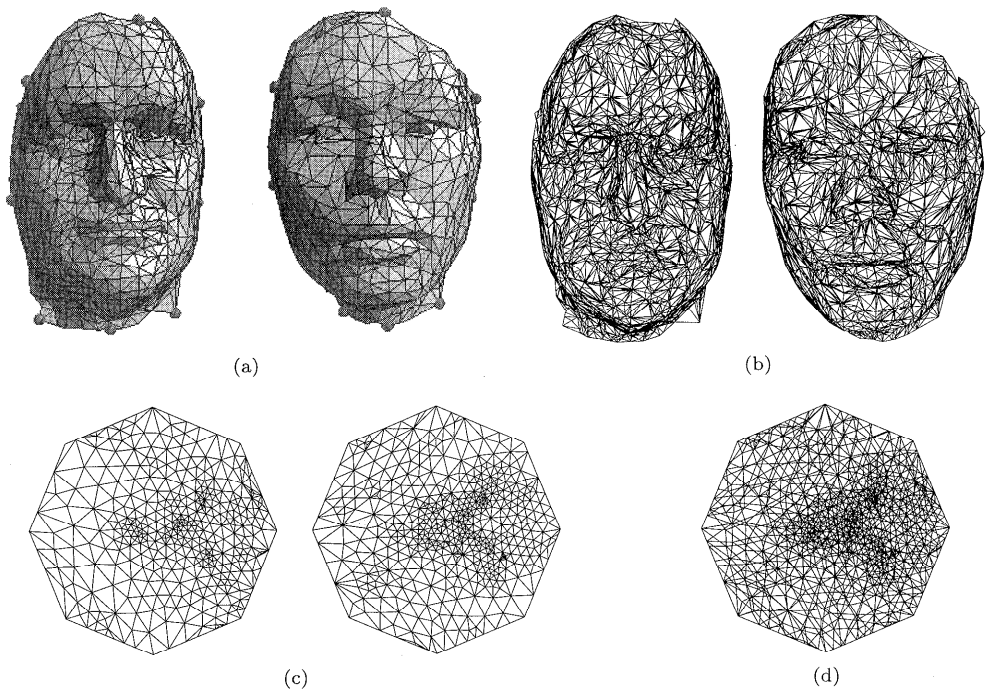

図 10 2つの顔モデル（Moser, Kanai）に対する合成メッシュ生成結果．(a) もとのメッシュと 8 個の対応頂点．(b) 合成メッシュ．(c) 埋め込みメッシュ．(d) 合成埋め込みメッシュ

Fig. 10 A result of combining two face models (Moser, Kanai). (a) Original meshes and eight CVs. (b) Combined meshcs. (c) Embedding meshes. (d) A combined embedding mesh.

${}^T\mathbf{H}^{hh}\mathbf{V}^h$ を繰り返し求める収束計算に帰着できる．これらは疎行列とベクトルのかけ算になる．よって，頂点の数を n_v，頂点の価数（1つの頂点に隣接するエッジの数）を n_{ve} とすると，1回の行列計算につき $O(n_v n_{ve})$ で計算できる．収束に必要な繰返し回数は，我々が行った例題のすべてが，10回を超えることはなかった．

(2) 交点算出アルゴリズムでは，$\mathcal{H}^1$ の1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程度であるから，$\mathcal{H}^1$ のエッジの数を n_e，1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．

(3) $\mathcal{H}^1$ 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1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，1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．

(4) $\mathcal{H}^c$ の頂点の生成には，$\mathcal{H}^1$，$\mathcal{H}^2$ の頂点の1回の探索と，エッジ間のすべての交点の1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1回の探索と，各エッジにつき交点の1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．

(5) こ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)は，頂点の数が多くなると頂点の数に対して価数は無視できる．(2)〜(4)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1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．

6. 適用結果

6.1 合成メッシュの生成

ここでは本手法を2つの例に適用し有効性を検討す

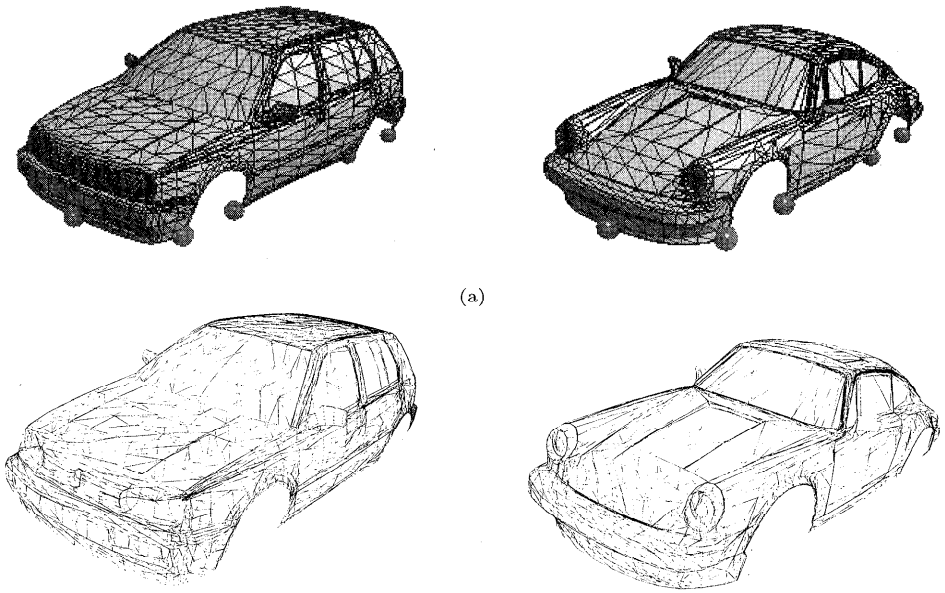

図 11 2つの車モデル（Golf, Porsche）に対する合成メッシュ生成結果．(a) もとのメッシュと 10 個の対応頂点．(b) 合成メッシュ
Fig. 11 A result of combining two car models (Golf, Porsche). (a) Original meshes and ten CVs. (b) Combined meshes.

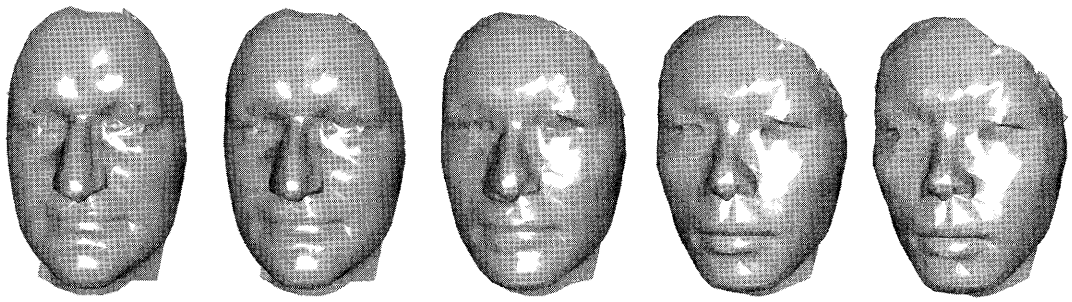

図 12 合成メッシュを用いた顔のモーフィング
Fig. 12 Face morphing using a combined mesh.

る．まず**図 10** に，2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 10 (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，たとえば顎の付近の頂点どうしというように，顔の特徴部分の頂点を対応づけしたものである．図 10 (b) に生成した2つの合成メッシュを示す．さらに，図 10 (c) に埋め込みメッシュを，図 10 (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．したがって，実装の段階で表示する必要はない．

また，より多い面の数を持つ 2 つの車のモデル（Golf, Porsche）に対し，**図 11** に合成メッシュを生成した結果を示す．比較のため，この例題は文献 9) と同じ例題を使用している．図 11 (a) にはもとのメッシュと境界上の特徴部分に対応づけられた対応頂点を，図 11 (b) には得られた 2 つの合成メッシュを示す．

表 1 に，2 つの例題についてのメッシュ要素数や計算時間などを示す．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の数（#v），エッジの数（#e），面の数（#f），対応頂点の数（#CV）のことを指す．また計算時間は，もとのメッシュから埋め込みメッシュの生成（$\mathcal{F} \mapsto \mathcal{H}$），埋め込みメッシュから合成埋め込みメッシュの生成（$\mathcal{H} \mapsto \mathcal{H}^c$）に費した時間を，MIPS R10000 CPU 175 MHz 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，

表 1 例題におけるメッシュの要素数と計算時間
Table 1 Number of mesh elements and calclation times in two examples.

	Moser	Kanai	Golf	Porsche
#v($\mathcal{F}$)	523	384	2,974	1,955
#e($\mathcal{F}$)	1,514	1,101	8,771	5,719
#f($\mathcal{F}$)	992	718	5,798	3,765
#v−#e+#f($\mathcal{F}$)	1	1	1	1
#CV	8		10	
#v($\mathcal{F}^c$)	3,612		15,174	
#e($\mathcal{F}^c$)	10,738		45,249	
#f($\mathcal{F}^c$)	7,127		30,076	
#v−#e+#f ($\mathcal{F}^c$)	1		1	
#time($\mathcal{F} \mapsto \mathcal{H}$) (Sec.)	0.15	0.10	1.48	1.17
#time($\mathcal{H} \mapsto \mathcal{H}^c$) (Sec.)	0.60		3.23	

おおよそ 6 秒ほどで合成のためのすべての計算を行うことができる．なお，車モデルの例は，文献 9) の手法を，本論文と同じ計算機環境で計測し直した結果，全体の処理に 22.93 秒かかった．これより，計算効率の良い手法として本手法の有効性を確認できた．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[16]（#v − #e + #f）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分かる．

6.2 モーフィングへの応用

次に，本手法により生成された合成メッシュをモーフィングに適用した例として，**図 12** に，図 10 (b) の，2 つの合成メッシュの頂点を線形補間することにより得られたモーフィング結果を示す．

モーフィングの際に考慮すべきこととして，いかにユーザが 2 つのメッシュの持つ特徴を正確に，しかも簡単に対応づけできるか，ということがあげられる[7]．本手法の中でユーザによる作業は，2 つのメッシュの境界上に対応頂点対を作ることだけである．メッシュ上の残りのすべての点は，本手法によって自動的に対応づけされる．さらに，文献 5) では，そのメッシュが対応づけ可能かどうかをチェックする必要があるが，本手法を適用することでそのような煩わしさからは開放される．

メッシュの境界には，対応点を増やすことで詳細な対応づけの制御を行うことができる．ただ，ある特定の領域に対応点を密集させることは，計算時間がかかるだけで，モーフィングの質の改良にはつながらない．また本手法では，境界頂点以外にユーザが対応づけをすることができない．これは特に複雑な形状を対象にする場合には問題である．これが本手法の限界であり，この問題を回避するためには，形状を分割したりするなどの別のスキームが必要になると思われる．

7. 結論と展望

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．また，例題を通じてこれらの利点を検証し，その結果実用的には有効な手法であることを確認した．さらに，本手法をモーフィングに適用した場合に，ユーザにとってより安易にモーフィングを生成できることも示した．

ただ，本手法は，円盤に位相同型という，位相的には限られた形状を対象としている．課題としては，本手法の球やトーラスなどの位相形状への拡張や，モーフィングのためのさらなる手法の改良などがあげられる．

謝辞 本論文の例題のうち，2 つの車のモデル（Golf, Porsche）は米 Viewpoint DataLab. 社により提供されるモデルである．また，2 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・金子研のご協力を得た．

なお，この研究の一部は（財）大川情報通信基金の支援を受けた．あわせてここに感謝の意を表する．

参 考 文 献

1) 松本幸夫：4 次元のトポロジー，日本評論社 (1991).
2) Beier, T. and Neely, S.: Feature-based image metamorphosis, *Computer Graphics* (*Proc. SIGGRAPH 92*), pp.35–42, ACM Press, New York (1992).
3) Lee, S., Chwa, K., Shin, S.Y. and Wolberg, G.: Image Metamorphosis Using Snakes and Free-Form Deformations, *Computer Graphics* (*Proc. SIGGRAPH 95*), pp.439–448, ACM Press, New York (1995).
4) Lee, S., Chwa, K., Hahn, J. and Shin, S.Y.: Image Morphing Using Deformation Techniques, *J. Visualization and Computer Animation*, Vol.7, No.1, pp.3–23 (1996).
5) Kent, J.R., Carlson, W.E. and Parent, R.E.: Shape transformation for polyhedral objects, *Computer Graphics* (*Proc. SIGGRAPH 92*), pp.47–54, ACM Press, New York (1992).
6) Chen, D.T., State, A. and Banks, D.: Interactive Shape Metamorphosis, *1995 ACM Sympo.*

on Interactive 3D Graphics, pp.43–44, ACM Press, New York (1995).
7) 湯本麻子，鈴木香緒里，佐々木繁：楕円球メッシュを用いた3次元モーフィング，グラフィックスとCAD研究会，78-5, pp.31–38, 情報処理学会 (1995).
8) DeCarlo, D. and Gallier, J.: Topological Evolution of Surfaces, *Proc. Graphics Interface '96*, pp.194–203, Morgan Kaufmann, San Francisco, CA (1996).
9) Kanai, T., Suzuki, H. and Kimura, F.: Three-dimensional Geometric Metamorphosis Based on Harmonic Maps, *The Visual Computer*, Vol.14, No.4, pp.166–176 (1998).
10) Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W.: Multiresolution Analysis of Arbitrary Meshes, *Computer Graphics* (*Proc. SIGGRAPH 95*), pp.173–182, ACM Press, New York (1995).
11) Eells, J. and Sampson, J.: Harmonic mappings of Riemannian manifolds, *American J. Mathematics*, Vol.86, pp.109–160 (1964).
12) Maillot, J., Yahia, H. and Verroust, A.: Interactive Texture Mapping, *Computer Graphics* (*Proc. SIGGRAPH 93*), pp.27–34, ACM Press, New York (1993).
13) 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).
14) 杉原厚吉：計算幾何工学，培風館 (1994).
15) Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P.: *Numerical recipes in C*, 2nd edition, Cambridge University Press, Cambridge, UK (1992).
16) Mäntylä, M.: *An Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland (1988).

(平成10年9月3日受付)
(平成11年9月2日採録)

金井　崇（正会員）

昭和44年生．平成10年東京大学大学院工学系研究科精密機械工学専攻博士課程修了・工学博士．同年理化学研究所に入所．現在基礎科学特別研究員として同研究所素形材工学研究室に所属．形状モデリングのCAD/CAMおよびCGへの応用に関する研究に従事．精密工学会，ACM，IEEE CS各会員．

鈴木　宏正（正会員）

昭和32年生．昭和61年東京大学大学院工学系研究科精密機械工学専門博士課程修了・工学博士．昭和62年東京大学助手（教養学部），昭和63年同講師，平成2年同助教授，平成6年同大工学系研究科助教授（精密機械工学専攻）．現在に至る．形状モデリング，CAD/CAMシステムの研究に従事．精密工学会，日本機械学会，日本図学会，ACM，IEEE各会員．

木村　文彦（正会員）

昭和20年生．昭和49年東京大学大学院工学系研究科航空学専攻博士課程修了．同年通産省電子技術総合研究所パターン情報部研究員．昭和54年東京大学工学部精密機械工学科助教授，昭和62年同教授．形状モデリング，CAD/CAM，生産システム等の研究に従事．工学博士．精密工学会，日本機械学会，ACM，IEEE等会員．