 Vol.2018-CVIM-212 No.9
2018/5/10

特徴的なテンプレート表情の転写による
アバターブレンドシェイプの自動生成

鬼塚 駿仁[1,a)] ディィエゴ トマ[2,b)] 内山 英昭[2,c)] 島田 敬士[2,d)] 谷口 倫一郎[2,e)]

概要：人間の顔の表情をキャラクターなどのアバターに転写 (リターゲティング) する際，ブレンドシェイプモデルが多く使用されている．しかし，ブレンドシェイプによるリターゲティングを行うには，特徴的な表情を持つ 3 次元のアバター (キーシェイプ) を事前に入手する必要がある．本研究では，ブレンドシェイプで用いられるアバターのキーシェイプを自動的に生成する手法を提案する．具体的には入力となる表情を持たない 3 次元のアバターに対して，あらかじめ用意したキーシェイプの表情の変化のみを転写することで，表情を持ったアバターを生成する．また，表情変化の転写が難しいテンプレートと形状が大きく異なるアバターに対しても，目や口のサイズ，向きを考慮して転写することで，正しい変形を行う．実験では提案手法の利点を示すため，従来手法との比較による出力結果の定性的な評価を行った．

Automatic generation of avatar blendshapes by transferring template facial expressions

Hayato Onizuka[1,a)] Diego Thomas[2,b)] Hideaki Uchiyama[2,c)] Atsushi Shimada[2,d)]
Rin-ichiro Taniguchi[2,e)]

Abstract: We introduce a semi-automatic method to create a set of pre-defined key facial expressions (called blendshapes) for any avatar mesh. Fitting a set of template blendshapes to data captured by a sensor with video and depth input is a successful technique for marker-less facial motion capture. However, the captured motion is represented in the space formed by the template blendshapes. Therefore, for accurate retargeting the avatar mesh must have its own set of blendshapes and a known mapping between the template and avatar blendshape spaces. This is difficult because manually creating blendshapes of any avatar mesh requires professional skills, and finding the mapping between the template and the avatar mesh can be challenging. In this work we automatically generate the blendshape of any avatar that map directly to the template blendshapes (the user is only required to select a few facial landmarks on the avatar mesh). By doing so, captured facial motion can be easily retargeted to any avatar, even when the avatar has largely different shape and topology compared with the template mesh. Our experimental results show the accuracy of our proposed method compared with the a state-of-the-art method for mesh deformation transfer.

1 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
2 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
a) onizuka@limu.ait.kyushu-u.ac.jp
b) thomas@ait.kyushu-u.ac.jp
c) uchiyama@limu.ait.kyushu-u.ac.jp
d) atsushi@ait.kyushu-u.ac.jp
e) rin@kyudai.jp

1. はじめに

一般層におけるインターネットの普及により，不特定多数のユーザー同士でコミュニケーションを取り合う機会が増加している．このような環境において，匿名性を守りつつ，自分自身を表現する手段として，自分の代役となるアバターを設定する場合がある．オンラインゲームや SNS など，多くのサービスにおいてアバターは使用されているが，特に近年ではアバターを設定するだけではなく，それ

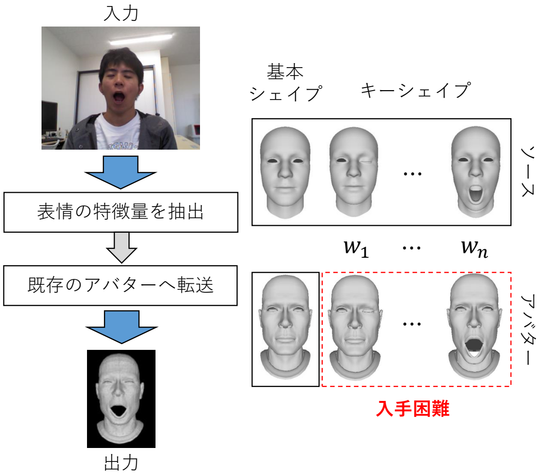

図 1 ブレンドシェイプによるリターゲティングの流れ

らにユーザーの顔や体の動きを付与する場合もある．このように，ユーザーの様々な表情，体の動きをアバターに転写するという技術，すなわちリターゲティング技術に対する世間の関心は高まりを見せている．

表情をアバターに転写する手法には，映画産業などにおいて多用されている，顔に多数のマーカーを付けてトラッキングを行い，マーカーの動きをアバターに伝える手法 [1]，入力となる表情の変形をダイレクトにアバターに適用する手法 [2] など，様々なものがある．しかし，ユーザーの顔に多数のマーカーを付ける必要があるものや，計算コストがかかり過ぎるものは，一般ユーザーが利用する上では好ましくない．そこで，今回はこれらのリターゲティング手法の中でも，マーカーレスで計算コストが軽い，ブレンドシェイプと呼ばれる手法 [3] について考察を行う．図 1 に大まかな流れを示す通り，ブレンドシェイプは入力となる表情をキーシェイプと呼ばれる表情モデルの重み付き線形和で近似することで各表情の特徴量を抽出し，その特徴量を同じ表情を持つアバターの重み付き線形和に使用することで，表情の転送を行う．しかし，ブレンドシェイプを行うにはアバターのキーシェイプをあらかじめ数十種類用意しておかなければならないが，それらの入手・作成難易度は非常に高い．また，特徴量は抽出に使用されたモデル (ソース) の表情における値が推定されるため，アバターのキーシェイプの数，表情が視覚的にすべてソースと一致しなければ，表情の転送が行えないという制約もある．

そこで，本研究では表情を持たないアバターメッシュから，アバターのキーシェイプ群を自動生成するシステムの構築を行う．このシステム内には，ソースとなる既知のキーシェイプ群がテンプレートとしてあらかじめ与えられており，表情の転写の際に目や口などのパーツごとの大きさ，向きを考慮した新たな転送手法を用いることで，ソースと形状が大きく異なるアバターに対しても正しいキーシェイプの生成を可能とした．このシステムにより，特別な技術が無くとも，誰でも比較的容易にあらゆるブレンドシェイプモデルを得ることが可能となる．また，ソースと同じ数，同じ表情のアバターのキーシェイプが生成されるため，ソースの表情におけるブレンドシェイプ係数の値をそのままアバターに適用することができる．

2. 関連研究

ある顔の表情のトラッキング，およびその表情を他の顔に転写する方法 (リターゲティング) については過去に様々な研究がなされている．以下では，それらに関する研究，及び一部のリターゲティング手法で用いられるメッシュ自体を変形する手法についての研究を紹介する．

Thies らは RGB 入力のみを用いてブレンドシェイプによる 3 次元の顔モデルの生成及びトラッキングを行った [4]．この研究では，入力となる RGB 映像から得られる顔領域の見た目と，ソースのキーシェイプの線形和で表された 3 次元モデルを 2 次元に投影したものの見た目の差を最小化することで，2 次元の RGB 映像から 3 次元の顔のモデルを復元している．入力が RGB のみであるため，システム構成が容易であるという利点があるが，手や髪などで隠れた領域 (オクルージョン) が発生した場合に正確な表情の推定が行われないという欠点がある．

Hsieh らは RGB-D カメラを用いて，RGB 入力のみでは対応できなかったオクルージョンに頑健なブレンドシェイプによる表情のトラッキング，リターゲティングを行った [5]．顔領域の各ピクセルごとに，デプスデータによる実測値とソースのキーシェイプの線形和によって推定された顔の表面との差を用い，オクルージョン領域の判別を行うことで，間違った表情の推定を抑制している．しかし，太陽光などの強い光源や急激な光源の変化が起こる環境においてはオクルージョン領域の取得が難しく，表情の推定が正しく行われない場合がある．

Sifakis らは，ユーザーの顔に付けた多数のマーカー及びアバター上の対応するマーカーの位置のずれを最小化することによるリターゲティングを行った [1]．この手法ではアバター内に筋肉や骨格を仕込んでおり，実際の人間の顔の動きに準拠した高精細なリターゲティングを行っている．結果としてはかなり高精度なリターゲティングを行うことができるが，ユーザーの顔にマーカーを付けなければならない点，アバターの設計が少し複雑すぎる点により，一般的な人々が利用するには制約が大きい．

リターゲティングにおいて，ブレンドシェイプでは既知のメッシュの重み付き線形和で新たな表情を表現することができるが，マーカーの移動を検知する手法などでは 1 つの基本となるアバターメッシュを変形する必要がある．ここでは，そのような手法で必要となるメッシュの変形を行う手法について紹介する．

Nor&Neumann らは，ソースのメッシュを構成する頂点の動きをアバター上の頂点に転送する手法を用いてメッッ

シュの変形を行った [6]．ユーザーは基本となる無表情のアバターメッシュ上の 10 個未満の点を選択し，その点とソースメッシュ上にあらかじめ与えられた対応する点を基に，ソース上の各頂点の動きを対応するアバター上の各頂点に適用することで表情の転写を行う．この手法は，表情の変化の転送には特化しているが，全体的なメッシュ形状の変化の転送には対応していない．

Sumner らは，頂点の動きではなく，代わりにポリゴンの変形の転送によるメッシュの変形を行った [2]．ソースとアバターの 2 つのメッシュ間で対応する位置に数十個設定したランドマーク点を補助として，無表情のアバターのメッシュを無表情のソースのメッシュの形に変形させたうえでソースとアバターのポリゴンの対応情報を取得し，ソースの各ポリゴンの変形を対応するアバターの各ポリゴンに適用することで表情の転写を行う．この手法では，表情部分に限らずメッシュ全体の動きを転送することができるが，細かい動きに関しては，アバターの形状がソースと大きく異なるほど正確な結果が得られなくなる．

3. 定義

本章では，まず計算式中で用いられる文字等の表記法の定義を行う．その後，3 次元モデルの変形の際に用いられる式について解説する．

3.1 表記法の定義

提案手法において，ベクトルは $\mathbf{a}$ のように太字，行列は $\mathbf{A}$ のように大文字の太字による表記を行う．また，リストは L のように大文字による表記，または一部 l のように小文字による表記を行い，その他例外についてはその都度説明を行う．

本研究で用いる 3D メッシュは $\{F, V\}$ のように，頂点のリスト $V = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$，及びポリゴンのリスト $F = \{f_1, ...f_m\}$ から構成され，頂点 $\mathbf{v}$ は 座標値 $(x\ y\ z)^\top$，ポリゴン f は頂点のリスト V の中から 3 点のインデックスを持つ．本手法では，ポリゴンのリスト F には変更を加えず，頂点のリスト V の各頂点が持つ座標値 $(x\ y\ z)^\top$ を変更した新たな頂点のリスト $\tilde{V} = \{\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n\}$ を計算することで，ポリゴンの数，及び隣接関係を保った状態で，新たなメッシュを生成する．

ブレンドシェイプでは，基本となる無表情のメッシュ (基本シェイプ)B_0，及び特徴的な表情を持ったメッシュ (キーシェイプ) 群 $B_i \quad (i = 0...n)$ が与えられたとき，

$$Mesh = B_0 + \sum_{i=1}^{n} \{w_i(B_i - B_0)\} \tag{1}$$

のような重み付き線形和を取ることで，様々な表情を持ったメッシュ作り出すことが出来る．また，本研究において使用する基本シェイプとキーシェイプにはソースとアバターの 2 種類がある．図 1 に示されるように，ソースは基本となる無表情のメッシュ (基本シェイプ)，及び表情のサンプルとなる特徴的な表情を持ったメッシュ (キーシェイプ) 群から構成される．また，アバターは基本シェイプのみを持ち，キーシェイプは持たない．ここで，今回の研究では，アバターの基本シェイプを入力とし，アバターのキーシェイプを生成することで，アバターにおいてもブレンドシェイプを行えるようにすることが目標である．

3.2 ポリゴンの変形の定義

提案手法では，あるポリゴンの変形を，3 次元におけるアフィン変換を用いて表現する．アフィン変換は，ポリゴンの回転，拡大縮小，剪断を表す 3×3 の行列 $\mathbf{T}$，及びポリゴンの平行移動を表す 3×1 のベクトル $\mathbf{d}$ から構成される．

ポリゴンのアフィン変換を表す際，ポリゴンの向きを考慮するため，$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ に加え，これらの頂点から以下のようにして計算された頂点 $\mathbf{v}_4$ を新たに追加する．

$$\mathbf{v}_4 = \mathbf{v}_1 + \frac{(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)}{|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)|} \tag{2}$$

この $\mathbf{v}_4$ は，$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ から構成されるポリゴンの正規化された法線が $\mathbf{v}_1$ から出た場合の先端の座標を示す．

提案手法の肝となる，ソースの変形をアバターに伝えるという手法を考える場合，単純には以下のような条件付きの評価関数を最小化することで実現が可能である．

$$\min_{(\mathbf{T}_1, \mathbf{d}_1), ..., (\mathbf{T}_{|T|}, \mathbf{d}_{|T|})} \sum_{i=1}^{|F|} ||\mathbf{S}_{s_i} - \mathbf{T}_{t_i}||_F^2$$
$$subject\ to \quad \mathbf{T}_j \mathbf{v}_i + \mathbf{d}_j = \mathbf{T}_k \mathbf{v}_i + \mathbf{d}_k, \quad \forall i, \forall j, k \in p(\mathbf{v}_i) \tag{3}$$

ここで，$\mathbf{S}$，$\mathbf{T}$ はそれぞれソース，アバターの基本シェイプからキーシェイプへのアフィン変換の線形変換成分であり，$\mathbf{S}$ は既知の値，$\mathbf{T}$ は未知の変数である．また，s_i, t_i はソースとアバターのポリゴンの既知の対応組であり，この対応組のポリゴン同士の線形変換成分の差の最小化を行うことで，ソースのポリゴンの変形をアバターに伝える．また，$||\cdot||_F$ はフロベニウスノルム，$|F|$ はアバターのポリゴンの総数である．しかし，メッシュを構成するポリゴン全てに対してこの条件を成り立たせることは難しく，計算が非効率的であるため，式 (3) を効率的な形に変形する．

式の効率化に際して，ポリゴンの変形を以下のように表す．

$$\mathbf{T}\mathbf{V} = \tilde{\mathbf{V}} \tag{4}$$

ここで，式中の $\mathbf{V}, \tilde{\mathbf{V}}$ はそれぞれ $\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_1, \mathbf{v}_4 - \mathbf{v}_1$ および $\tilde{\mathbf{v}}_2 - \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_3 - \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_4 - \tilde{\mathbf{v}}_1$ を水平方向に結合した 3×3 の行列である．

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_2 - \mathbf{v}_1 & \mathbf{v}_3 - \mathbf{v}_1 & \mathbf{v}_4 - \mathbf{v}_1 \end{bmatrix} \tag{5}$$

$$\tilde{\mathbf{V}} = \begin{bmatrix} \tilde{\mathbf{v}}_2 - \tilde{\mathbf{v}}_1 & \tilde{\mathbf{v}}_3 - \tilde{\mathbf{v}}_1 & \tilde{\mathbf{v}}_4 - \tilde{\mathbf{v}}_1 \end{bmatrix} \tag{6}$$

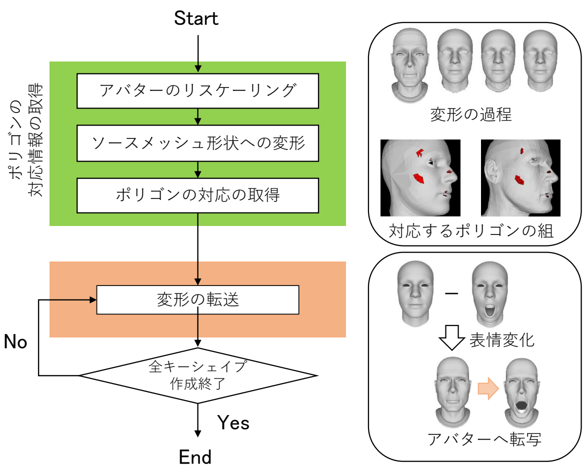

図 2　提案手法の流れ

また，変形後の頂点 $\tilde{\mathbf{v}}_i$ は，$\tilde{\mathbf{v}}_i = \mathbf{T}\mathbf{v}_i + \mathbf{d}$ のように表される．これらの式より，ポリゴンの線形変換成分 $\mathbf{T}$ は次のような頂点座標を変数とした関数で表される．

$$\mathbf{T} = \tilde{\mathbf{V}}\mathbf{V}^{-1} \tag{7}$$

これにより，式 (3) は頂点 $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$ を新たに変数として次のように書くことができる．

$$\min_{\tilde{\mathbf{v}}_1,...,\tilde{\mathbf{v}}_n} \sum_{i=1}^{|F|} ||\mathbf{S}_{s_i} - \mathbf{T}_{t_i}||_F^2 \tag{8}$$

このように，線形変換成分及び平行移動がソースメッシュのポリゴンと近くなるようにアバターメッシュの頂点の位置自体を動かすことで，隣接関係および全体の頂点数を保ったまま，アバターの変形を行う．以降，メッシュの変形に関する評価関数を定義する際は，この頂点ベースによる手法を用いる．

4.　提案手法

本研究では表情を持たないアバターメッシュから，アバターのキーシェイプ群を自動生成するシステムの構築を行う．提案するシステムでは，Sumner[2] らの手法に対して新たに顔のランドマークの移動を評価する関数を導入することで，ポリゴンの変形の転送のみでは難しい，大きく形状が異なるアバターへの表情の転写を可能とする．

4.1　概要

提案手法では，ユーザーが各自用意した表情を持たないアバターのメッシュ (基本シェイプ) を入力とし，あらかじめ用意されたソースのキーシェイプの持つ表情を転写することで，アバターのキーシェイプを新たに作成する．今回提案するシステムは，図 2 に示す通り，大きく分けて 2 つのステップで構成される．ポリゴンの対応情報の取得のステップでは，アバターの基本シェイプのサイズ調整を行った後，アバターの基本シェイプの形状をソースの基本シェイプの形状に変形させ，最後にそれらを重ね合わせて最も距離と方向が近いポリゴン同士を対応する組として登録する．次のソースの変形の転送のステップでは，前のステップで得られた対応組リストを基にした，アフィン変換の線形変換成分の転送，及びソースの顔のパーツを構成する頂点の動きに合わせてアバターの対応する頂点を動かすことによる変形の転送の 2 つの手法を組み合わせて用い，ソースのキーシェイプの表情をアバターに転写する．

4.2　ポリゴンの対応情報の取得

本節では，図 2 に緑色で示される，ポリゴンの変形の転送元・転送先となるソース・アバターのポリゴンの対応付けを取るステップについて述べる．また，後の節においても重要となる，頂点座標を用いたアフィン変換の線形変換成分の表現についても述べる．

4.2.1　アバターのリスケーリング

入力となるアバターには大小様々なスケールのものがある．そこで，正しいポリゴンの対応の取得のため，以下の式を用いてアバターのリスケーリングを行い，ソースメッシュとアバターメッシュのスケールを視覚的にほぼ同程度にする．

$$\mathbf{v}^{'} = \mathbf{v}\frac{x^s + y^s + z^s}{x^t + y^t + z^t} \tag{9}$$

ここで，$\mathbf{v}$ はリスケーリング前のアバターの各頂点，$\mathbf{v}^{'}$ はリスケーリング後のアバターの各頂点である．また，x^s, y^s, z^s および x^t, y^t, z^t はそれぞれソースメッシュ，アバターメッシュの各座標軸における頂点座標の最大値と最小値の差を表しており，これらの和の比率によってソース，アバター間のメッシュのスケール比を算出する．

4.2.2　ソースメッシュ形状への変形

ここでは，ポリゴンの対応を取るための下準備として，アバターの形状をソースの形状に変形させる．具体的には，以下の 3 つの制約付き評価関数の重み付き線形和を取り，この評価関数の最小化を重みを変化させながら数回繰り返し行うことで変形を行う．

1 つ目の評価関数は以下の式で表される．

$$E_S(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{|F|} \sum_{j \in adj(i)} ||\mathbf{T}_i - \mathbf{T}_j||_F^2 \tag{10}$$

ここで，$\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$ は変形後のアバターメッシュの頂点，$adj(i)$ は i 番目のポリゴンに隣接するポリゴンの集合である．また，$\mathbf{T}_i$ はアバターメッシュのポリゴンの線形変換成分であり，3.2 項で紹介したような，頂点を変数とした関数である．

2 つ目の評価関数は以下の式で表される．

$$E_I(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{|F|} ||\mathbf{T}_i - \mathbf{I}||_F^2 \tag{11}$$

ここで，$\mathbf{I}$ は 3×3 の単位行列を表している．

3 つ目の評価関数は以下の式で表される．

$$E_C(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{n} ||\mathbf{v}_i - \mathbf{c}_i||^2 \qquad (12)$$

ここで，$\mathbf{c}_i$ は最近傍有効点と呼ばれるソースメッシュ上の頂点であり，アバターメッシュ上の頂点 $\mathbf{v}_i$ に対して，$\mathbf{v}_i$ と $\mathbf{c}_i$ の頂点法線のなす角度が 90° 以内で座標が最も近い頂点である．図 3 に示される例のように頂点法線の角度を考慮することで，唇の上下などを区別した頂点の対応を取ることができる．

以上の 3 つの評価関数の重み付き線形和を取った評価関数を最小化することで，変形後のアバターの頂点座標 $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$ を得る．また，最小化に際しては，計算の高速化のため，[7] に示される正規方程式を用いた最適化手法を用いる．

$$\begin{aligned} \min_{\tilde{\mathbf{v}}_1,...,\tilde{\mathbf{v}}_n} \ & E(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = w_s E_S + w_i E_I + w_c E_C \\ & \textit{subject to} \quad \tilde{\mathbf{v}}_{corr(i)} = [\mathbf{v}_i]_{s,neut}, \quad i \in L \end{aligned} \qquad (13)$$

ここで，$[\mathbf{v}]_{s.neut}$ は，頂点 $\mathbf{v}$ がソースの基本シェイプ上の頂点であることを表す．また，以下で紹介するが，L はソースメッシュ上のランドマークのインデックスのリスト，$corr(i)$ はソースの i 番目のランドマークに対応するアバターの i 番目のランドマークのインデックスである．

この評価関数を最小化する上で必要となるのが，ユーザーによる顔のランドマークの設定である．ランドマークとは，眉，目，鼻，口などのパーツの周りに設定された，アバターメッシュの変形の補助を行う頂点である．ソースには既にこちらからランドマークの頂点のインデックスのリスト L が与えられている．ユーザーはこのソースのランドマークを参考に，アバターの対応する箇所にランドマークを手動で指定する．ここで指定されたアバターのランドマークは，アバターメッシュをソースメッシュ形状へ変形する際，必ずソースメッシュ上の対応するランドマークと同じ座標に移動しなくてはならないという強い制約を設けることで，アバターの形状をソースの形状へ適切に収束させることが可能となる．

最適な結果を得るため，線形結合された評価関数の最小化をそれぞれの評価関数にかかる重みを変化させながら数回繰り返す．本研究では反復回数を 4 回とし，1 回目の反復では重みを $w_s = 1, w_i = 0.1, w_c = 0$，2 回目以降の反復については $w_s = 1, w_i = 0.001, w_c = 1$ とした．これにより得られた結果を図 4 に示す．

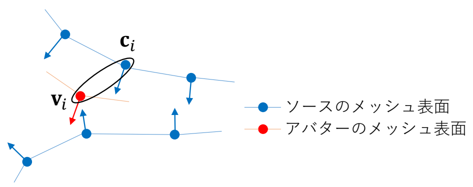

図 3　最近傍有効点の取得例

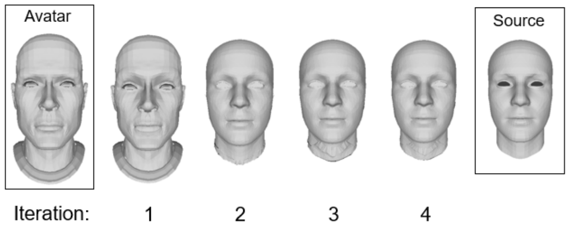

図 4　アバターメッシュの変形の過程

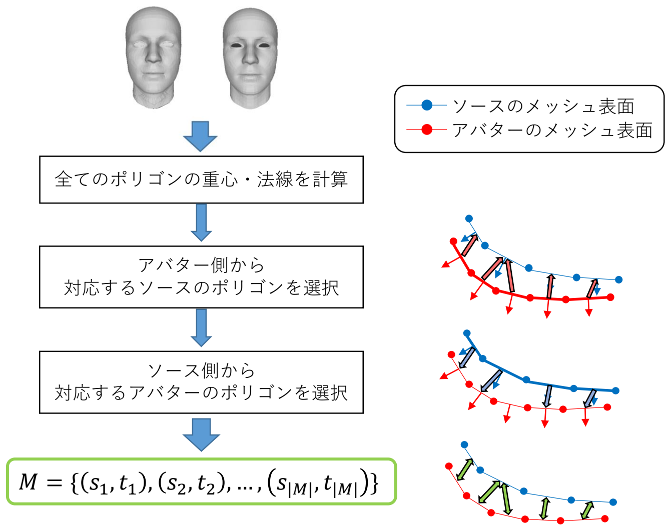

図 5　ポリゴンの対応取得の流れ

4.2.3　ポリゴンの対応の取得

このようにして得られたソースの形状に変形したアバターを用い，ポリゴンの対応情報の取得を行う．手法としては，ソースの基本シェイプと得られた変形後のアバターの基本シェイプとの比較を行い，対応するポリゴンの組を取得する．図 5 にポリゴンの対応取得の流れを示す．

取得に際しては，まず，変形後のアバターメッシュ及びソースメッシュの全てのポリゴンの法線及び重心を計算する．その後，アバターメッシュ上のあるポリゴンに対して，そのポリゴンと重心の座標が最も近く，法線同士のなす角度が 90° 以内のソースメッシュ上のポリゴンを探索し，それらのポリゴンを対応組とする．この処理を全てのアバターメッシュのポリゴンに対して行った後，逆の処理を全てのソースメッシュのポリゴンに対して行う．このように，対応の取得の処理を双方向から行うことで，1 対多，多対 1 の対応を許容し，ポリゴンの総数が異なるメッシュ同士においても正しい対応を取ることができる．ここで得られたポリゴンの対応は以下のように表す．

$$M = \{(s_1, t_1), (s_2, t_2), ..., (s_{|M|}, t_{|M|})\} \qquad (14)$$

ここで，s_i はソースメッシュのポリゴンのインデックス，

t_i はアバターメッシュのポリゴンのインデックスである．

4.3 変形の転送

本節ではソースのキーシェイプと同じ表情を持ったアバターメッシュの生成を行う．ここでは前節で得られたポリゴンの対応を用いたソースメッシュのポリゴンの変形の転送，およびソースのランドマークの動きに合わせてアバターのランドマークを動かすことによる表情の転送を行う．

4.3.1 ポリゴンの変形の転送

この評価関数では，前節で得られたポリゴンの対応関係を用いたソースのポリゴンの変形の転送を行う．評価関数は以下の式で表される．

$$E_d(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{|F|} ||\mathbf{S}_{s_i} - \mathbf{T}_{t_i}||_F^2 \tag{15}$$

この式は，式 (8) に対して，ポリゴンの既知の対応組として前節で得られた $M = \{(s_1, t_1), (s_2, t_2), ..., (s_{|M|}, t_{|M|})\}$ の要素を用いたものである．

Sumner[2] らの手法では，この評価関数のみを用いてソースの変形をアバターに転送している．4 章において，提案手法とこの評価関数のみを用いた Sumner らの手法による表情の転写の結果を比較し，考察を行う．

4.3.2 ランドマークの動きの転送

対応するポリゴン間の線形変換成分の転送だけでは対応できないメッシュに対する表情の転送を保証するため，本研究ではユーザーによって与えられたランドマークの動きをソースの表情変化によるランドマークの動きと対応させる以下の評価関数を導入した．

$$E_l(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{k=1}^{6} \sum_{i \in L_k} ||\mathbf{A}_k \mathbf{R}_k \mathbf{u}_i - \tilde{\mathbf{u}}_{corr(i)}||^2$$
$$where \quad \begin{cases} \mathbf{u}_i = [\mathbf{v}_i]_{s.exp} - [\mathbf{v}_i]_{s.neut} \\ \tilde{\mathbf{u}}_{corr(i)} = \tilde{\mathbf{v}}_{corr(i)} - [\mathbf{v}_{corr(i)}]_{t.neut} \end{cases} \tag{16}$$

ここで，L_k は後述するが，ソースメッシュ上の顔のパーツごとのランドマークのインデックスリストである．また，$corr(i)$ はソースメッシュ上のランドマークに対応するアバターメッシュ上のランドマークのインデックスである．$\mathbf{u}_i$，$\tilde{\mathbf{u}}_{corr(i)}$ は，ソースメッシュ，アバターメッシュそれぞれにおける，表情の変化による各ランドマークの 3 次元の移動量を示している．ここで $[\mathbf{v}]_{s.exp}, [\mathbf{v}]_{s.neut}, [\mathbf{v}]_{t.neut}$ は，それぞれ $\mathbf{v}$ が表情を持ったソースメッシュ，無表情のソースメッシュ，無表情のアバターメッシュ上の点であることを表す．また，$\mathbf{A}_k, \mathbf{R}_k$ についても後述するが，それぞれソースメッシュのランドマークの移動量，及び移動の方向をアバターに合わせる役割を持つ．

アバターによって顔のパーツの大きさ・向きは異なるため，本研究ではソース，アバターの顔の各パーツを右眉，左眉，鼻，右目，左目，口の 6 つに分け，パーツの向き，サイズを測定し，ランドマークの移動量及び方向を調整した．

はじめに各パーツの向きの測定を行う．それぞれのパーツを構成するランドマークを $L_k (k \in 1...6)$ とし，この中から，さらにパーツの方向を示すのにふさわしいと思われるランドマークを選び，そのリストを L'_k とする．その後，L'_k 内の任意の 2 点間に引かれる直線すべてに対しておよそ垂直になるようなベクトル $\mathbf{n}_k$ を以下の評価関数を最小化することにより算出する．

$$E(\mathbf{n}_k) = \sum_{i=0}^{|L'_k|} \sum_{j=i}^{|L'_k|} \mathbf{n}_k \cdot (\mathbf{v}_{L'_k[i]} - \mathbf{v}_{L'_k[j]}), \quad k \in 1...6 \tag{17}$$

ここで得られたパーツの向き $\mathbf{n}_k$ より，z 軸正方向を向いたベクトルを，$\mathbf{n}_k$ と同じ方向に向ける回転行列 $\mathbf{R}_{k,s}, \mathbf{R}_{k,t}$ を求める．また，ソース，アバターの 6 種類の顔のパーツそれぞれに対して個別に $\mathbf{R}_{k,s}, \mathbf{R}_{k,t}$ の値のセットを保持する．

顔の各パーツのサイズの測定方法については，まず，それぞれのパーツを前のステップで得られたローテーション $\mathbf{R}_{k,s}, \mathbf{R}_{k,t}$ の転置行列を用いて z 軸正方向に向ける．その後，パーツを構成するランドマークの中での x, y, z 軸それぞれにおける座標の最大値，最小値の差を取り，アバターの各パーツの各軸方向におけるサイズを $a_{x,s}$ $a_{y,s}$ $a_{z,s}$，ソースの各パーツのサイズを $a_{x,t}$ $a_{y,t}$ $a_{z,t}$ とする．これらの要素より，各パーツのサイズ比 $\mathbf{A}_k$ を以下のようにして得る．

$$\mathbf{A}_k = \begin{bmatrix} \frac{a_{x,t}}{a_{x,s}} & 0 & 0 \\ 0 & \frac{a_{y,t}}{a_{y,s}} & 0 \\ 0 & 0 & \frac{a_{z,t}}{a_{z,s}} \end{bmatrix} \tag{18}$$

ここで例外として，無表情なアバターメッシュのみでは口の可動域が測定不能なため，口の y 軸方向のサイズ比の算出に鼻のパーツの下端，および下唇の中央上端間の距離を用い，z 軸方向のサイズ比については 1 に固定した．

このようにして得られた各パーツのサイズ比 $\mathbf{A}$，ソースメッシュの各パーツの向きをアバターメッシュの各パーツの向きに合わせるローテーション $\mathbf{R}_k = \mathbf{R}_{k,t} \cdot \mathbf{R}_{k,s}^{\top}$ を用いて，ソース上での各ランドマークの動きをアバター上での動きに変換する．以上の操作で得られたアバターメッシュ上での動きに変換されたソースメッシュ上の各ランドマークの移動ベクトル $\mathbf{A}_k \mathbf{R}_k \mathbf{u}_i$ とアバターの各ランドマークの移動ベクトル $\tilde{\mathbf{u}}_{corr(i)}$ の差を最小化することで，ソースメッシュにおけるランドマークの移動からアバターのランドマークの移動を導く．

4.3.3 最適化

これら 2 つの評価関数，及び前節において紹介したメッシュ表面の平滑化を行う評価関数 E_S, E_I の重み付き線形和を取った評価関数を最小化することで，変形後のアバターの頂点座標 $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$ を得る．

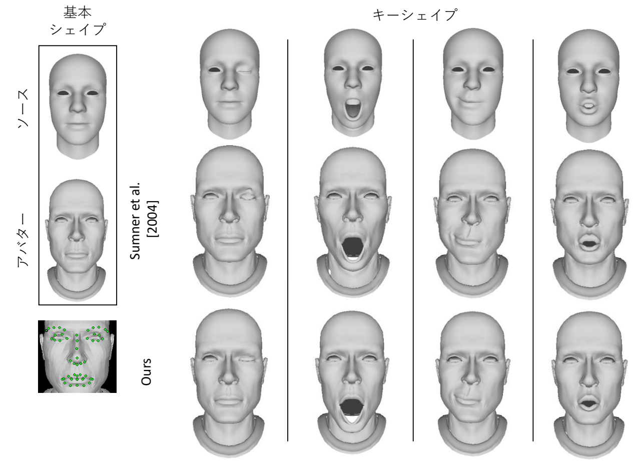

図 6　人間の顔のアバターの出力結果

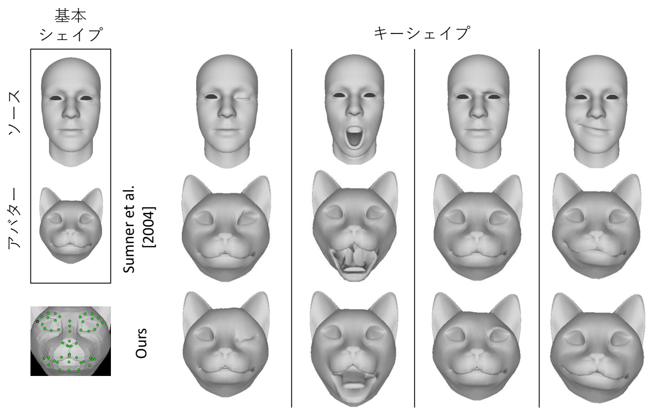

図 7　猫のアバターの出力結果

$$\min_{\tilde{\mathbf{v}}_1,...,\tilde{\mathbf{v}}_n} E(\tilde{\mathbf{v}}_1,...,\tilde{\mathbf{v}}_n) = w_d E_d + w_l E_l + w_i E_I \qquad (19)$$

ここで，重みはそれぞれ $w_d = 1, w_l = 100, w_s = 10, w_i = 1$ とした．この評価関数の最小化の処理を，与えられているソースのすべてのキーシェイプに対して繰り返し行い，ソースメッシュのキーシェイプと同じ表情を持つアバターメッシュのキーシェイプを生成する．

5.　実験

入力とするアバターとして，ソースメッシュと形状の類似度が高い人間の顔のメッシュ，及び形状の類似度が低い馬や猫等のメッシュを用意した．ここでは，Sumner[2] らの手法と本研究における手法を比較しながら考察を行う．図 6，図 7，図 8，図 9 に出力結果を示す．

各アバターに対して，選択したランドマークを各図の左下に示す．人間のアバターに対しては，アバターとソースの顔のパーツ形状の類似度がかなり高いため，ある程度正確な頂点を選択することができるが，顔のパーツ形状の類似度が低い猫や馬などのアバターに関しては，アバターの見た目を基にある程度対応する頂点を選択した．

人間のアバター (図 6) に関しては，入力となるアバターが人間の顔であることから正しいポリゴンの対応が得られたこと，及び表情の変化によるポリゴンの変形がソースと類似していたことにより，双方の手法において高精度な表情の転写結果となっている．しかし，猫や馬のアバター (図 7，図 8) に関しては Sumner らの手法では主に目の周りを中心に正しく表情の転写が行えていない．これはメッシュの全体的な構造がソースとは異なるため，正しいポリゴンの対応が取れなかったこと，及び表情変化によるポリゴンの変形の仕方がソースとは異なっていたことが原因であると考えられる．一方，本研究による手法ではポリゴンの変形に加えてランドマークの移動による頂点ベースでの変形も考慮しており，正しい表情の転写を行えている．

キャラクターのアバター (図 9) に関しては，提案手法では目を完全に閉じることが出来ているが，その際額の部分が不自然な動きをしている．これは単純にアバターメッシュのポリゴンの密度が低いことが原因であると考えられる．これに対しては，密度の低い部分のポリゴンを分割するなどの解決策が考えられる．また，提案手法の一部のメッシュにおいて口の内側にある歯のパーツが唇付近から若干突き出ている．今回の手法では構成されるパーツ間の関係を考慮していないため，本来起こり得ない動きに対する制約がないことが原因であると思われる．このような不自然な動きはメッシュの形状に凹凸が多い場合や表情

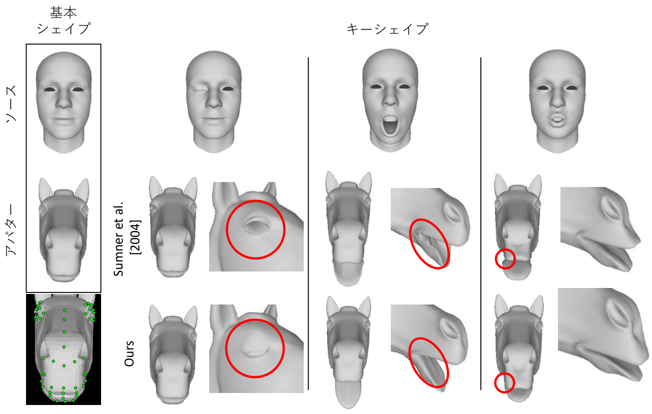

図 8　馬のアバターの出力結果

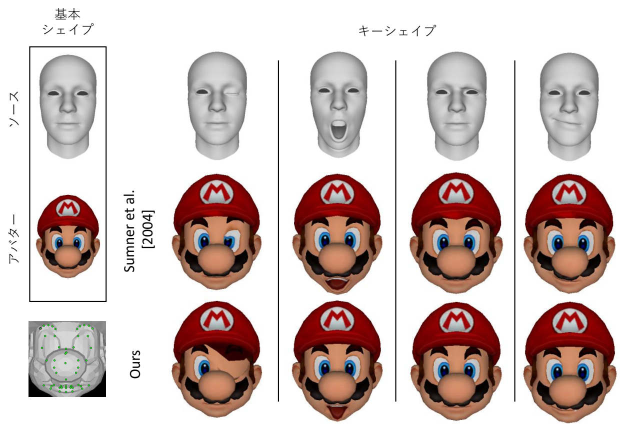

図 9　キャラクターのアバターの出力結果

の変化の大きさによっては，パーツ毎に分かれていないアバターに対しても起こる可能性がある．このような複数のメッシュから構成されるアバター，及び凹凸の多いアバターに対応するには，メッシュを構成するパーツや凹凸ごとに，それぞれ周りとの位置関係を記憶し，周りのパーツの動きに付随して動くような条件付けをするなどの対策をとる必要がある．

また，アバターによっては図 8 のように口が開きすぎるもの，図 9 のように若干開き加減が足りないものが確認できる．これは，ランドマークの動きを転送する関数において使用される顔の各パーツのサイズ比 $\mathbf{A}$ に，無表情なアバターメッシュのみでは，口の開き加減の限界値を知ることが出来ず，口のパーツのサイズ比のみ推定値を用いたことが原因であると考えられる．解決策としては，出力結果をユーザーが判断し，スケールの調整を行うなどの手法が考えられるが，キーシェイプの自動生成という目的に反するため，慎重に折衝点を見つける必要がある．

提案手法では，Sumner[2] らの手法の表情に対する忠実度を高めるため，Deformation Transfer の評価関数に加え，新たにランドマークの動きを模倣するような評価関数を導入した．また，今回の実験から，

- 単一のメッシュからなるアバターにおいては，様々な形状のアバターに対して正しく表情の転写を行うことが可能である
- 複数のメッシュから構成されるアバター，ポリゴンの密度が少ないアバターに対しては不自然な転写結果となる場合がある
- アバターの口の開き具合に関しては改善が必要である

という結果が得られた．幾つかの改善点はあるものの，全体的に見ると様々なアバターに対してソースメッシュと同じ表情を持ったキーシェイプが生成されており，特に形状が大きく異なるアバターメッシュに対しても適切にリターゲティングが行えたことは，新たに導入した評価関数の有効性を示している．

6. まとめ

本研究では，ブレンドシェイプで用いられるアバターのキーシェイプを自動的に生成する手法を示した．具体的には Sumner[2] らが用いた手法をベースにし，Deformation transfer の評価関数に対して，新たにユーザーによって与えられた顔のランドマークの動きを模倣するコストを導入することで，形状が大きく異なるアバターに対しても正しい動きを保証した．

サンプルとして数種類のアバターを用いて行った実験においては，アバターのメッシュが複数のパーツで構成されている場合や，メッシュ毎の個別の形状の特性により一部不自然な点が現れている場合があるが，殆どのアバターのキーシェイプにおいて自然な表情のメッシュが生成されており，様々な形状のアバターに対してキーシェイプを生成するという本研究の目的に貢献することができた．

今回の手法によって，ユーザーが各自好みのアバターからブレンドシェイプに用いられるキーシェイプの作成を行うことが可能となった．今後の研究では，RGB-D カメラ等を用い，これらのアバターに対して顔の皺や身体的特徴などを含む忠実度の高いリターゲティングを行う手法について考察を行う．

参考文献

[1] Sifakis, E., Neverov, I. and Fedkiw, R.: Automatic Determination of Facial Muscle Activations from Sparse Motion Capture Marker Data, *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, New York, NY, USA, ACM, pp. 417–425 (online), DOI: 10.1145/1186822.1073208 (2005).

[2] Sumner, R. W. and Popović, J.: Deformation Transfer for Triangle Meshes, *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, New York, NY, USA, ACM, pp. 399–405 (online), DOI: 10.114-5/1186562.1015736 (2004).

[3] Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F. and Deng, Z.: Practice and Theory of Blendshape Facial Models, *Eurographics 2014 - State of the Art Reports* (Lefebvre, S. and Spagnuolo, M., eds.), The Eurographics Association, (online), DOI: 10.2312/egst.20141042 (2014).

[4] Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C. and Nießner, M.: Face2Face: Real-time Face Capture and Reenactment of RGB Videos, *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2016).

[5] Hsieh, P. L., Ma, C., Yu, J. and Li, H.: Unconstrained realtime facial performance capture, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1675–1683 (online), DOI: 10.1109/CVPR.2015.7298776 (2015).

[6] Noh, J.-y. and Neumann, U.: Expression Cloning, *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, ACM, (online), DOI: 10.1145/1185657.1185862 (2006).

[7] 株式会社タイコ牛山善太：最適化とは、最小二乗法について，株式会社オプティカルソリューションズ 光学設計ノーツ 53 (ver.1.0) (2013).