修士論文　　平成 20 年度 (2008)

テクスチャの特徴箇所を考慮した 3DCG モーフィング

東京工科大学大学院メディア学研究科

飯田 英輔

修士論文　　平成 20 年度 (2008)

テクスチャの特徴箇所を考慮した 3DCG モーフィング

指導教員　渡辺 大地

東京工科大学大学院メディア学研究科

飯田 英輔

論 文 の 要 旨

論文題目	テクスチャの特徴箇所を考慮した 3DCG モーフィング
執筆者氏名	飯田 英輔
指導教員	渡辺 大地
キーワード	3DCG テクスチャ モーフィング

[**要旨**]

3DCG の技術にモーフィングというものがある。モーフィングとはある 3DCG モデルから別の 3DCG モデルへ時間の経過とともに形状変形を行うものである。3DCG のモーフィングは自由な視点からモデルを見ることができ、その見た目の面白さから映画やアニメーション、ゲームなどで利用されている。3DCG モデルを映像作品に用いる利点は、作業効率の上昇やデータの使いまわしによる利便性の上昇などが挙げられる。モーフィング技術は形状についての研究は盛んであるが、3DCG モデルの表面画像も含めたモーフィングの研究はあまりされていない。3DCG モデルの形状と表面情報を同時にモーフィングを行うには様々な問題がある。その問題とは 2 つの 3DCG モデルの表面画像をどのように変化させていくか、その表面画像を形状の正しい位置に表示させなくてはならないことなどである。これらの問題を解決することにより、3DCG モデルを使用した映像作品の制作現場でより広い範囲でモーフィングを活用することができるようになり、表現の幅が広がる。

本研究では、表面画像の特徴箇所を維持したまま、表面画像も含めた 3DCG モーフィングの手法を提案する。モーフィングに使用する 2 つのモデルからモーフィング実行前に UV マップの情報を抜き出し、それに変換を加えることでテクスチャ付きのモーフィングを行う。

本手法を用いて作成した実行例により、その有用性を示す。

A b s t r a c t

Title	3DCG morphing that considers feature part of texture
Author	Eisuke Iida
Advisor	Taichi Watanabe
Key Words	3DCG Texture Morphing

[**summary**]

Morphing is the technique of 3DCG. That is the one to transform shape from 3DCG model to another 3DCG model with the time passage. 3DCG morphing is used from the movie, animation, and game for interest of the externals, and it can see the model from a free viewpoint. The reseach of Shape is active in morphing Technique but I think that it necessary the one including the surface image of 3DCG model to use it more easily on the site of the image production. There are various problems in doing morphing at the same time shape and surface information on 3DCG model. The problems are how should change the surface image of two models of 3DCG, and to make the surface image displayed at the position where shape is correct. It comes to be able to use morphing by solving these problems within the wider range on the production site of the image work that uses 3DCG model, and the user expression extends. The purpose of this study is the technique of 3DCG morphing including the texture image with the feature part of the texture image maintained. Morphing with the texture is done by pulling out information in the UVmap before morphing is executed from two models used for morphing, and adding conversion to it. I show the utility according to the execution example to make by using this technique.

目 次

第 1 章

はじめに

1.1 研究背景と目的

近年、アニメーションやビデオゲームでは、3 次元コンピュータグラフィクス (以下 3DCG) が利用されることが多くなっている。3DCG の技術の 1 つにモーフィングがある。モーフィングとは、ある物体から別の物体へと時間の経過とともに滑らかな変形を行うものである。モーフィングには 2 次元で画像処理の技術を使用して行う方法と、3DCG モデルを使用したものがある。2 次元で処理を行ったものには、映画「ターミネーター 2」やマイケル・ジャクソンのプロモーションビデオ「black and white」[1] などがあり、話題にもなった。2004 年に発売された携帯ゲーム機「プレイステーション・ポータブル」には、3DCG モデルを使用したモーフィングをサポートする機能が組み込まれている [2]。映画やアニメーションの現場では 3DCG モデルを用いた作品も多く、3DCG モデルを利用したモーフィングを使用する場面は数多く考えられる。また、映像作品以外にも寺沢ら [3] のインターネットを利用して、直感的に物体の情報を提供するためにモーフィングを応用した研究や、水田ら [4] の医学教育を目的としてヒト胎児の成長表現をモーフィングで行ったもの、別宮ら [5] の 2 つの入力画像から光源位置の補間を行い、光源の位置を自由に置き換えることができる研究がある。

映画やアニメーションなどの映像作品でモーフィングを利用する場合は、3DCG モデルを使用していてもレンダリングを行い 2 次元で画像処理を行う場合が多い。2 次元で処理を行う理由は、3DCG モデルのままモーフィングを行うよりも処理が簡単なことが挙げられる。3DCG モデルを使用してモーフィングを行う利点は、モーフィング中でも様々なカメラアングルから見ることが可能な点である。これによりカメラアングルを動かしながら、モーフィングシーンを表現できる。本研究では 3DCG モデルを使用したモーフィングを扱う。

コンピュータでの 3DCG モデルの記述方法は、CSG 表現、ボクセルモデル、境界表現などがある [6]。CSG 表現は、プリミティブな基本図形を組み合わせることで複雑な立体を表すものである。プリミティブな形状を和、差、積などを用いて

立体を表現しているため、細かく複雑な形状を表現するのが難しく、徐々に形状が変化していくモーフィングには不向きである。ボクセルモデルとは、3 次元の格子状に微小な立方体を並べることで立体を表現する。複雑な形状を表現するのに向いているが、データ量が膨大であり変換作業に時間がかかってしまう。境界表現とは、形状を頂点、稜線、面で構成する平面の多角形によって表すものである。そのため曲面は多角形を使用して、近似することで表現する。幾何学的な計算に向いており、頂点を移動することにより形状の変形が行える。三角形は必ず平面になるため、多角形には三角形を用いて表現することが多い。本研究では、この境界表現で全ての面を三角形で構成した三角形メッシュモデルを扱う。

メッシュモデルを対象としたモーフィングの研究で中心になっているのは、対応問題、補間問題というものである。対応問題とは、3DCG モデルを構成するメッシュの各頂点をソースモデルとターゲットモデルの間で、1 対 1 の対応関係を構築することである。モーフィングを行うことを前提にせずに作成したソースモデルとターゲットモデルを使用するには、この問題が大きく関わってくる。モデル間の頂点数や面数が異なる場合やメッシュの接続性が異なる場合に、共通の頂点数や面数、もしくは接続性を持ったメッシュを構築することで、モーフィングを行えるようにする。この問題に、湯本ら [7] の楕円球を用いる手法や、Gregory[8] らのパラメータ化を行うことによって補間用のメッシュを構築する手法がある。補間問題とはソースモデルとターゲットモデルの間で構築した対応関係の頂点をどのように補間するかというものである。モデルの形状によって補間の途中で形状の歪みや自己交差が現れる場合があり、補間方法によりそれを解決するものである。また、補間方法によって、変形途中の形状に差がでたり、形状の一部分だけの変形を行ったりすることもできる。この問題に対して、川井ら [9] のスペクトル分解を用いて、形状の一部のみの変形を可能にした手法や、舩富ら [10] らの自己交差を回避する手法がある。道川ら [11] の研究では、近似剛体手法を用いて高品質なモーフィングを実現した。これらの 2 つの問題は主に形状に関わる問題である。形状に関わる研究以外では、白石ら [12] の画像を絵画風にレンダリングを行

い、なおかつ絵画風の特色を失わないモーフィング手法の研究や、白濱ら [13] の複数枚の画像からキャラクターの中間画像で表情を自動に生成する研究がある。

3DCG モデルを使用したモーフィングの利用方法として、大きく 2 つに分類できる。1 つ目は大きく形状の異なった 3DCG モデルを使用して行うものである。例を挙げると人間と犬の 3DCG モデルを用意して、人間から犬へと変化するというものである。2 つ目は同一の対象を表すものであるが、異なった状態を表す 3DCG モデルを使用し、その中間の形状を得るものである。例を挙げると直立状態と椅子に座った状態の人間の 3DCG モデルを使用して、椅子に座る動作を生成するものである。このような中間の動作を生成するモーフィング技術は、Maya[14] や 3ds Max[15] などの 3DCG モデルを扱うソフトウェアに機能の 1 つとして組み込まれている事もある。この 2 つの利用方法の大きな差は、表面画像の変化である。2 つ目の利用方法は、同一の対象を表した 3DCG モデルなのでソースとなるモデルとターゲットになるモデルの間に表面画像の違いがない。一方、1 つ目の利用方法はまったく異なった対象を表した 3DCG モデルを使用するので、表面画像も異なり形状の変化と共に表面画像を変化させなくてはならない。中間の動作を生成するモーフィングは、3DCG を扱うソフトウェアに機能の 1 つとして組み込まれている。表面画像を変化させるモーフィングは変形に伴って、その形状に適した表面画像を作成しなければならなくなり、それを作成するには時間や手間が掛かる。また、3DCG モデルの形状の表面画像が適した位置に表れるようにしなければならない。1 つ目の利用方法で表面画像をモーフィング途中でも適した位置に表示させるには、表面画像も含めてモーフィングを実行することを前提に、ソースモデルとターゲットモデルを作成する必要がある。このことを前提に作成していない 2 つの 3DCG モデル間で、表面画像も含めてモーフィングを実行するには、表面画像に関わる部分を作成し直さなければならない。これには非常に手間が掛かる。

先に挙げたように形状については、モーフィングを行うことを前提としてない頂点数や面数が異なるソースモデルとターゲットモデルに変換を加えることでモーフィングを可能にするが研究ある。本研究では、表面画像に焦点を当て、表面画

像も含めてモーフィングを行うことを前提としていないソースモデルとターゲットモデルに変換を加えることで、形状だけではなく表面画像も同時にモーフィングを行うことを目的とした。その際には、ソースモデルとターゲットモデルの形状の特徴箇所のみならず、表面画像の特徴箇所も対応が取れ、実行することが重要である。本研究により、モーフィングを行うことを前提としていないモデル間で表面画像も含めてモーフィングが行えるようになる。これによりゲームやアニメーションなどの映像作品で 3DCG モデルを用いた表現の幅が広がる。

1.2 論文構成

本論文は、本章を含めて全 5 章からなる。第 2 章では、三角メッシュモデルの構成と表面画像をモーフィングする際の問題点を挙げ、第 3 章で 3DCG モデルのテクスチャ付きモーフィングの手法について述べる。第 4 章では、本論文の手法を実装した結果の評価を行い、第 5 章で研究のまとめを述べる。

第 2 章

研究に用いる3DCGモデルと問題点について

この章では、本研究に用いる 3DCG モデルと、その 3DCG モデルの構成から起こるモーフィング実行時における問題点を述べる。

2.1 三角メッシュモデル

本研究で扱う三角メッシュモデルについて述べる。三角メッシュモデルとは、3 次元座標上に配置した頂点から三角形の面を構成し、その三角形の面が集合することで 3 次元形状を表現するものである。この 3 次元座標はベクトルで表現することにより、ベクトルや行列を使った計算を行うことができる [16]。モーフィングを行う際には、この三角形を構成する頂点の座標値を移動することで、形状の変形を表現する。図 2.1 は立方体を表現した三角メッシュモデル、図 2.2 は猫の頭部を表現した三角メッシュモデルである。

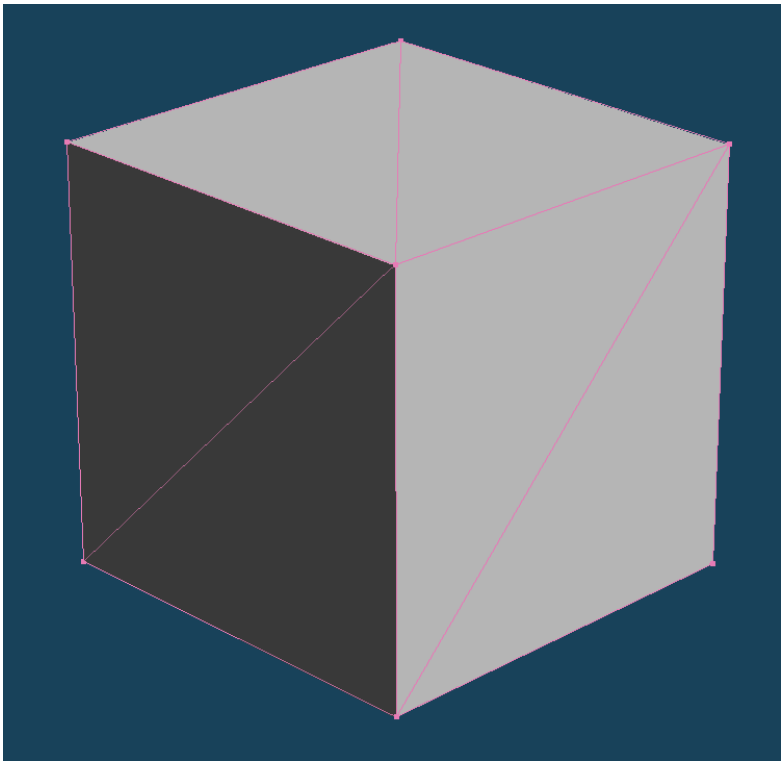

図 2.1: 立方体の三角メッシュモデル

モデルの表面画像はテクスチャ画像というものを利用することにより、表現し

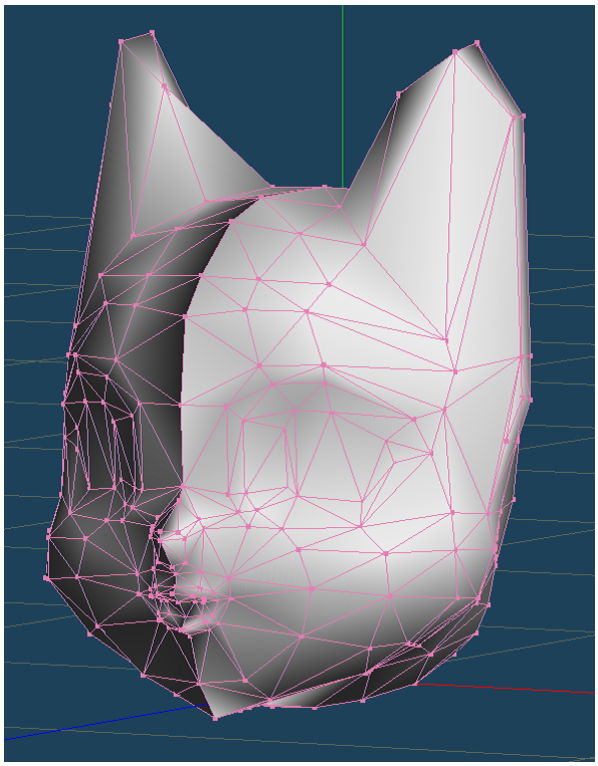

図 2.2: 猫の三角メッシュモデル

ている。このテクスチャ画像をモデルの表面に貼り付けることをテクスチャマッピングという。岩肌やレンガの表面を表したテクスチャ画像を用いて 3DCG モデルの質感を表現したり、目や口などが描かれたテクスチャ画像を貼り付けて 3DCG モデルの目や口を表現することができる。画像にはビットマップ画像とベクター画像の 2 種類があるが、3DCG モデルにはビットマップ画像を使用する。ビットマップ画像はピクセルという小さな点が集合して、1 つの画像となる。ピクセルは色情報を数値で持っていて、それにより色を表現する。本研究では色情報を RGB 法で扱う。RGB 法とは Red、Green、Blue の 3 原色のそれぞれの強さを数値で指定するものである。それぞれの Red(R 値)、Green(G 値)、Blue(B 値) を 256 段階の数値で指定する。この場合表現できる色は $256^3 = 16,777,216$ 色である。図 2.3 は市松模様のテクスチャ画像、図 2.4 は立方体にそのテクスチャ画像を貼り付けたものである。

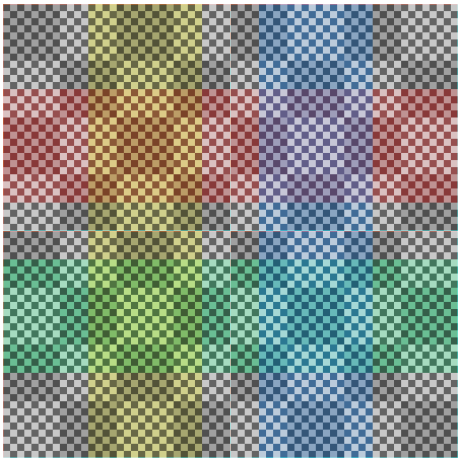

図 2.3: 市松模様のテクスチャ画像

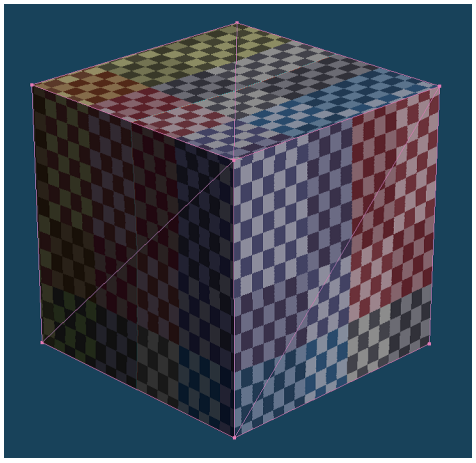

図 2.4: テクスチャ画像を貼り付けた立方体のモデル

図 2.5 は猫の目とひげを描いたテクスチャ画像、図 2.6 は猫の頭部のモデルにそのテクスチャ画像を貼り付けたものである。

テクスチャマッピングは、形状の各頂点とテクスチャ画像上の位置をどのように対応をとるか指定していくことによって出来上がる。このときに指定したテクスチャ画像上の位置を 2 次元座標で表現したものを UV 座標という。この UV 座標

図 2.5: 猫のモデルに使用するテクスチャ画像

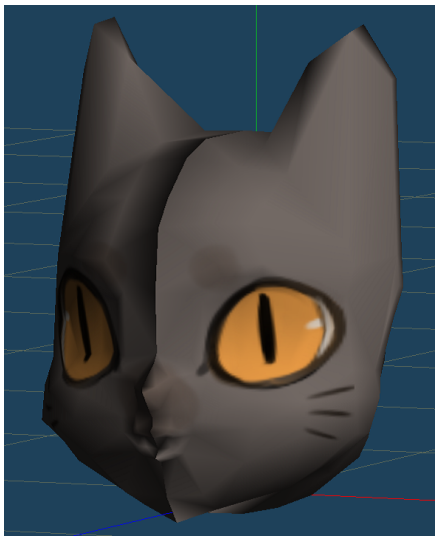

図 2.6: テクスチャ画像を貼り付けた猫のモデル

により、形状が変形しても形状の頂点とテクスチャ画像上の位置が対応を保つことができる。形状の全ての頂点が持つ UV 座標を 2 次元上に反映し、三角メッシュと同じ構成で線を結んだものを UV マップという。UV マップの作成は、制作者によって様々な形になる。メッシュの接続をある領域ごとに分割した後に展開し並べる方法や、形状をある箇所で切り開いて平面にした展開図を用いる方法などがある。図 2.7 はテクスチャ画像上に立方体の面を分割して並べた UV マップを黒い線で表示したもの、図 2.8 は立方体を切り開いて展開した UV マップである。本研究では図 2.8 のように、形状を 1 枚に切り開いて展開した UV マップを対象とする。

2.2 テクスチャ付き 3DCG モデルモーフィングの問題点

テクスチャ画像の特徴箇所の対応を考えてモーフィングを行う場合は、前提条件としてソースモデルとターゲットモデルの間で同一の UV マップを使用してモデル作成をすることが必要になる。同一の UV マップを使用していないモデル間でテクスチャ画像も含めてモーフィングを行おうとしたときは、UV マップとテク

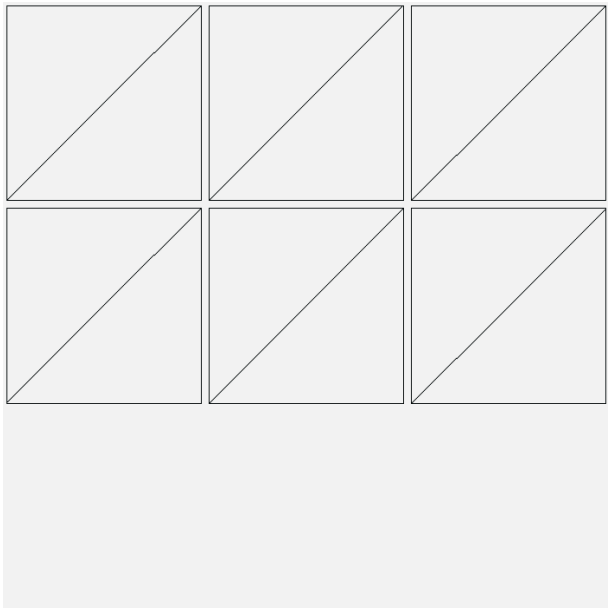

図 2.7: 分割した UV マップ

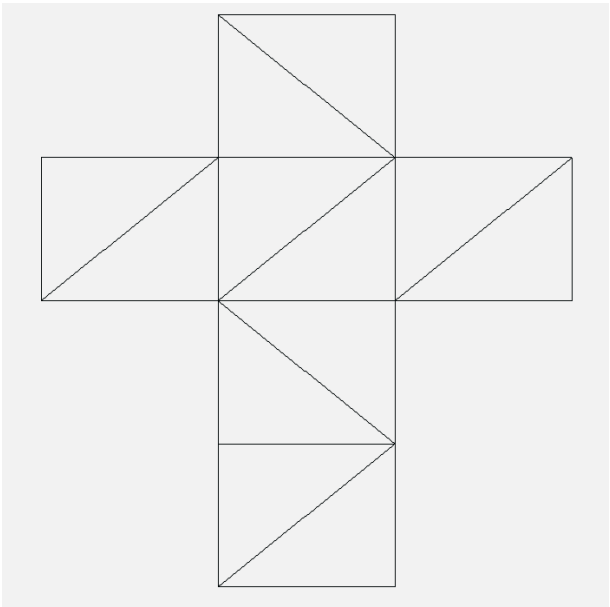

図 2.8: 切り開いた UV マップ

スチャ画像を作成し直さなくてはならない。ソースモデルとターゲットモデルのUV マップが同一でなく、作り直さずに実行するには 2.1 節で述べた、形状の頂点座標値、テクスチャ画像の RGB 値、UV マップの UV 座標値をモーフィング実行時に操作することが考えられる。しかし、これらの 3 種類の値をそれぞれ線形的に補間を行うと不都合が起こる。不都合とはテクスチャ画像が形状の表面の適した位置に表れないことである。この問題は、形状とテクスチャ画像の関係を表す UV 座標値に関わりがある。テクスチャ画像の補間を行うと、テクスチャ画像上に表される特徴箇所の位置は徐々にずれていく。図 2.9 は猫と象の 3DCG モデルに図 2.10 のテクスチャ画像を使用し、形状の頂点座標値、テクスチャ画像の RGB 値、UV マップの UV 座標値をそれぞれ直線的な線形に補間したものである。この 2 つのモデルはテクスチャを含めてモーフィング実行することを前提として作成していないので、UV マップは同一のものではない。中央の図で形状に表れるテクスチャ画像の目の位置がずれて表示されているのがわかる。

図 2.9: テクスチャのずれ

これはモデル間で UV マップが異なるためにモーフィング途中で算出される UV 座標値が、テクスチャ画像の適した位置を示さないことが原因である。

他の補間方法として、テクスチャ画像に対して、2 次元の画像処理の技術のモーフィングを応用することも考えることができる。画像の特徴箇所を考慮したモーフィングでは、ソース画像とターゲット画像で対応させる箇所を指定する。指定した箇所を基に三角形分割し、三角形ごとに形と色の補間を行う。この方法を 3DCG

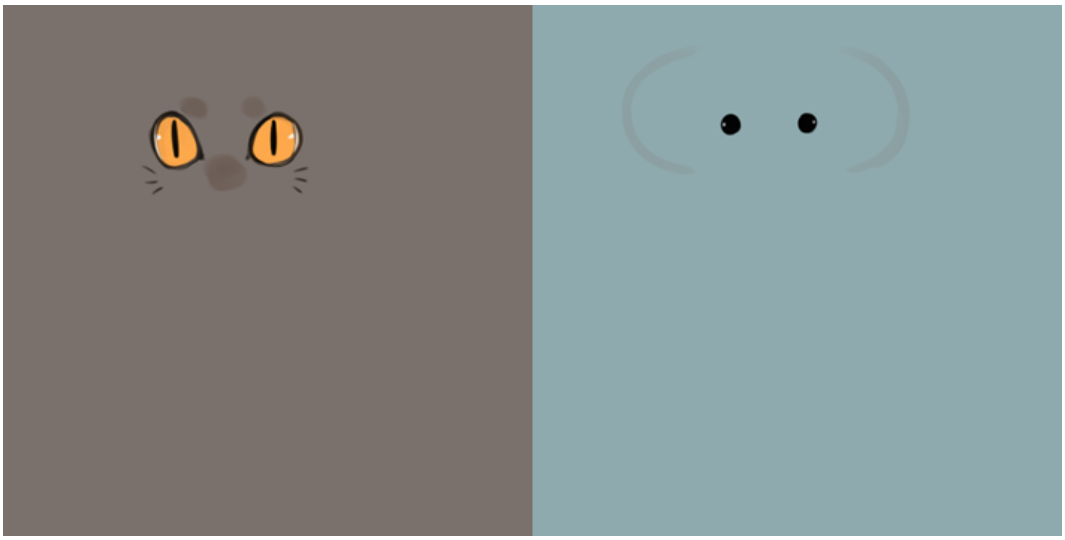

図 2.10: モデルに使用したテクスチャ画像

モデルに応用すると、テクスチャ画像上に特徴箇所を指定しなければならなくなり、モーフィングを利用する際に手間が増えてしまう。また、補間途中のテクスチャ画像に対して、形状の各頂点が適切な UV 座標値を得る保証がないため、テクスチャがずれて表示されることがある。

第 3 章

提案手法

この章では、本研究の提案手法を述べる。提案手法は大きく 2 つの段階に分けることができる。第 1 段階は、モーフィング実行前にソースモデルとターゲットモデルから補間用に使用する UV マップの作成、それに伴いそれぞれのモデルのテクスチャ画像の変換を行う。第 2 段階で作成した補間用 UV マップと変換したテクスチャ画像を使用して、ソースモデルとターゲットモデル間でモーフィングを実行する。

3.1 モーフィング実行前の UV マップ変換

本研究では、モーフィング途中のテクスチャ画像の表示のずれを防ぐ解決策として、ソースモデルとターゲットモデルに自動的な変換を加えてソースモデルとターゲットモデルで同一の UV マップを使用できる手法を考えた。本研究では、ソースモデルとターゲットモデルのそれぞれの UV マップから新たにモーフィング実行時に使用する補間用 UV マップを作成することにした。補間用 UV マップの作成に伴い、ソースモデルとターゲットモデルのそれぞれのテクスチャ画像も補間用 UV マップに適したものへ変換する。この補間用 UV マップ作成時に本来の UV マップから大幅な移動や変形があるとテクスチャ画像の特徴が損なわれてしまうことがあるため、補間用 UV マップはソースモデルとターゲットモデルのそれぞれの UV マップの特徴をなるべく維持したものが望ましい。UV マップは 2.1 節で述べたように、三角形メッシュとなっている。2 つの異なる三角形メッシュから理想的な三角形メッシュを作成することは、2 次元モーフィングの分野で研究が行われている。Han-Bing ら [17] の研究でソースとターゲットとなる画像からモーフィングに使用しやすい補間用の共通のメッシュを構成するものがある。また、Alexa ら [18] の近似剛体補間というものがある。この手法ではソースとターゲットとなる 2 次元の画像を三角形の集合に分割し、それぞれの対応する三角形に対して理想的な補間形状を求める。実際の補間形状は、全ての三角形が相互に重ならないようにするため、それぞれの理想的な三角形の形状の計算結果に近似させたものを算出する。そうすることによって歪みの少ない補間形状を作ることができ、視

覚的に違和感の少ないモーフィングが行える。これらの研究は 2 次元モーフィングのため、補間形状を時間の経過と共にいくつも生成するものであり、計算も複雑になっている。UV マップの頂点数は 3DCG モデルの形状の頂点数と同数になるため、モデルにより UV マップの頂点数が多くなることがある。そこで本研究では、より簡易な計算で 1 つの補間形状である補間用 UV マップを作成する。

補間用 UV マップの作成の際に、ソースモデルとターゲットモデルの UV マップに含まれる各頂点座標値を線形補間しただけでは不都合な場合がある。それはソースモデルとターゲットモデルの UV マップの状態によって、三角形が裏返る、あるいは潰れてしまうことである。図 3.1 の (a) はソースモデルの UV マップ、(b) はターゲットモデルの UV マップ、(c) がそれぞれ対応する UV 座標の頂点同士を線形補間したものである。(c) で三角形の裏返りや潰れが起きているのがわかる。

以下に、このような不都合が起こらない補間用 UV マップの作成方法を述べる。頂点座標値の単純な線形補間だけでなく、回転をある程度補正することにより三角形の裏返りや潰れを防ぐことができる。まずは回転についてソースモデルとターゲットモデルの UV マップ全体がどの程度傾いているのかを計算する。ソースモデルの UV マップに含まれる三角形の頂点を $\mathbf{S}_{ij}(i = \{0, 1, \ldots, n-1\}, j = \{0, 1, 2\})$、ターゲットモデルの UV マップに含まれる三角形の頂点を $\mathbf{T}_{ij}(i = \{0, 1, \ldots, n-1\}, j = \{0, 1, 2\})$ とする。この対応関係にある 1 対の三角形に注目する。この三角形同士がどの程度回転しているかを調べるために、各三角形の重心を中心とした対応する頂点の角度差を計算する。ソースモデルの三角形の重心座標を $\mathbf{G}_i(i = 0, 1, \ldots, n-1)$、ターゲットモデルの三角形の重心座標を $\mathbf{H}_i(i = 0, 1, \ldots, n-1)$ とすると $\mathbf{G}_i$ と $\mathbf{H}_i$ はそれぞれ式 3.1 で求める。

$$\begin{aligned} \mathbf{G}_i &= \frac{(\mathbf{S}_{i0} + \mathbf{S}_{i1} + \mathbf{S}_{i2})}{3} \\ \mathbf{H}_i &= \frac{(\mathbf{T}_{i0} + \mathbf{T}_{i1} + \mathbf{T}_{i2})}{3} \end{aligned} \tag{3.1}$$

対応関係にある重心座標 $\mathbf{G}_i, \mathbf{H}_i$ と対応関係にある各頂点座標 $\mathbf{S}_{ij}, \mathbf{T}_{ij}$ を用いて、重心を中心としたときの対応関係にある頂点の角度差 $\theta_{ij}(i = \{0, 1, \ldots, n-1\}, j =$

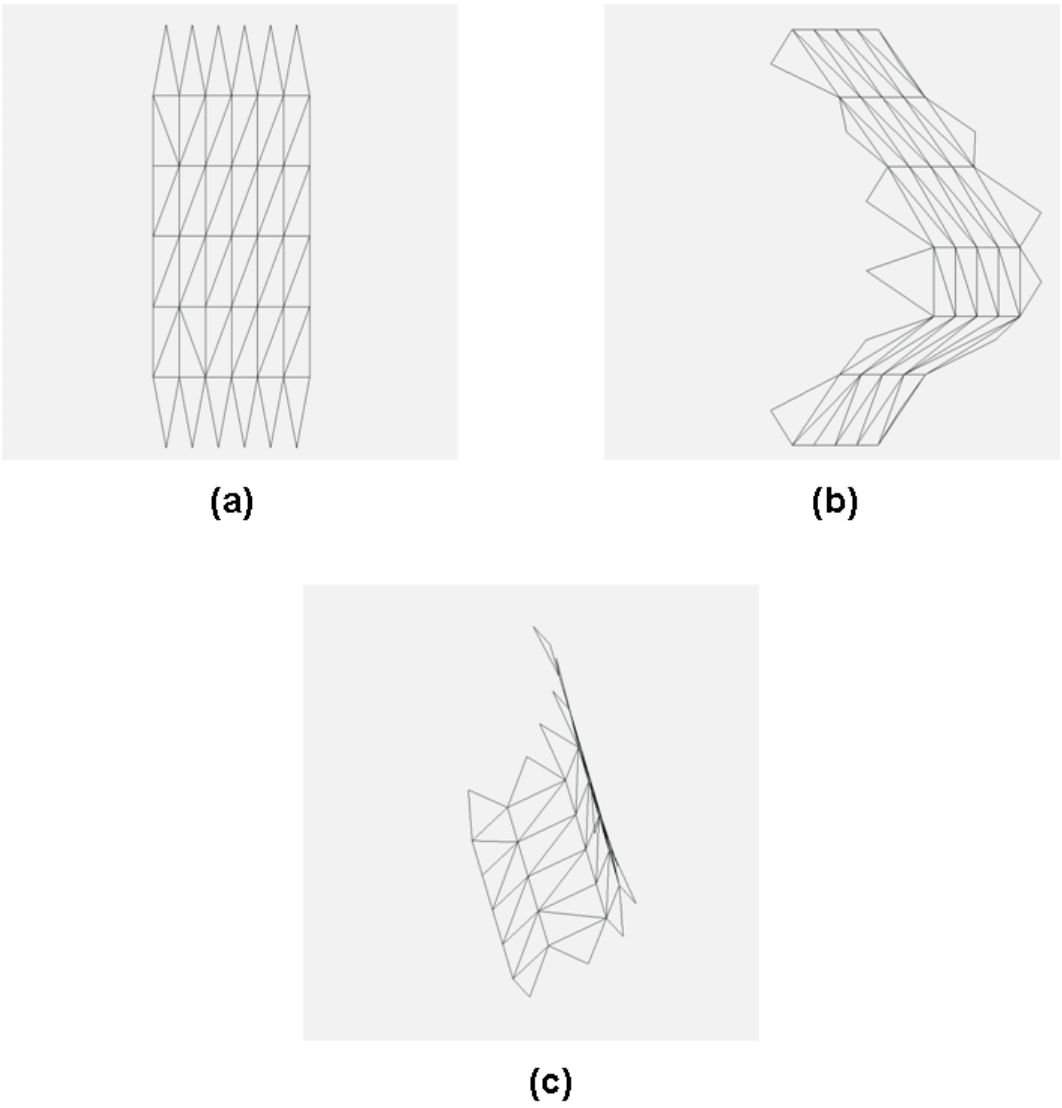

図 3.1: 線形補間で不都合な場合

$\{0,1,2\}$) を式 3.2 で求める。式中の $\overrightarrow{\mathbf{G}_i\mathbf{S}_{ij}} \cdot \overrightarrow{\mathbf{H}_i\mathbf{T}_{ij}}$ は $\overrightarrow{\mathbf{G}_i\mathbf{S}_{ij}}$ と $\overrightarrow{\mathbf{H}_i\mathbf{T}_{ij}}$ の内積を表している。

$$\cos\theta_{ij} = \frac{\overrightarrow{\mathbf{G}_i\mathbf{S}_{ij}} \cdot \overrightarrow{\mathbf{H}_i\mathbf{T}_{ij}}}{|\ \overrightarrow{\mathbf{G}_i\mathbf{S}_{ij}}\ ||\ \overrightarrow{\mathbf{H}_i\mathbf{T}_{ij}}\ |} \tag{3.2}$$

対応関係にある三角形内の 3 つの角度差 θ_{ij} から式 3.3 を用いて、三角形 $\mathbf{S}_{ij}$ と三角形 $\mathbf{T}_{ij}$ の重心を中心としたおおよその角度差 $\theta_i (i = 0, 1, \ldots, n-1)$ がわかる。

$$\theta_i = \frac{1}{3}\sum_{j=0}^{2}\theta_{ij} \tag{3.3}$$

これを全てのソースモデルとターゲットモデルの対応関係にある三角形で行い、式 3.4 で UV マップ全体のおおよその角度差 θ がわかる。

$$\theta = \frac{1}{n}\sum_{i=0}^{n-1}\theta_i \tag{3.4}$$

次にソースモデルの UV 座標上の全ての頂点を UV 座標 $(\frac{1}{2}, \frac{1}{2})$ を中心に θ 回転する。この回転後のソースモデルの三角形の頂点座標を $\mathbf{S}'_{ij}$ とすると、$\mathbf{S}'_{ij}$ は式 3.5 で求める。

$$\mathbf{S}'_{ij} = \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -\frac{1}{2} \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} \mathbf{S}_{ij} \tag{3.5}$$

また回転した後、ソースモデルの三角形の重心座標を再計算し $\mathbf{G}'_i$ とする。その重心座標 $\mathbf{G}'_i$ とターゲットモデルの対応する三角形の重心座標 $\mathbf{H}_i$ で線形補間を行う。この値が補間用 UV マップに作成される三角形の重心座標 $\mathbf{I}_i$ となる。補間用 UV マップの重心座標 $\mathbf{I}_i$ は次の式 3.6 で求める。式中の t はソースモデルとターゲットモデルの混合率である。

$$\mathbf{I}_i = (1-t)\mathbf{G}'_i + t\mathbf{H}_i \tag{3.6}$$

これを UV マップに含まれる全ての重心座標に対して行う。補間用 UV マップの各三角形の重心座標 $\mathbf{I}_i$ が決定した後は、ソースモデルとターゲットモデルの UV

マップに含まれる対応する三角形ごとに頂点座標の補間を行う。1 つの三角形について重心座標 $\mathbf{I}_i$ に重心 $\mathbf{G}'_i$ と重心 $\mathbf{H}_i$ を移動する。このとき重心 $\mathbf{G}'_i$ と三角形の各頂点 $\mathbf{S}'_{ij}$、重心 $\mathbf{H}_i$ と三角形の各頂点 $\mathbf{T}_{ij}$ の位置関係を保ったまま $\mathbf{S}'_{ij}, \mathbf{T}_{ij}$ も移動を行う。行列 $\mathbf{M}_s, \mathbf{M}_t, \mathbf{M}_r$ を式 3.7 のようにすると移動後の各頂点座標 $\mathbf{S}''_{ij}, \mathbf{T}'_{ij}$ は式 3.8 で求める。式 3.8 中の $\mathbf{G}'_{ix}, \mathbf{G}'_{iy}, \mathbf{H}_{ix}, \mathbf{H}_{iy}, \mathbf{I}_{ix}, \mathbf{I}_{iy}$ は、それぞれ $\mathbf{G}'_i, \mathbf{H}_i, \mathbf{I}_i$ の x 座標成分、y 座標成分を表す。

$$\begin{aligned}
\mathbf{M}_s &= \begin{pmatrix} 1 & 0 & -\mathbf{G}'_{ix} \\ 0 & 1 & -\mathbf{G}'_{iy} \\ 0 & 0 & 1 \end{pmatrix} \\
\mathbf{M}_t &= \begin{pmatrix} 1 & 0 & -\mathbf{H}_{ix} \\ 0 & 1 & -\mathbf{H}_{iy} \\ 0 & 0 & 1 \end{pmatrix} \\
\mathbf{M}_r &= \begin{pmatrix} 1 & 0 & \mathbf{I}_{ix} \\ 0 & 1 & \mathbf{I}_{iy} \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
\mathbf{S}''_{ij} &= \mathbf{M}_r \mathbf{M}_s \mathbf{S}'_{ij} \\
\mathbf{T}'_{ij} &= \mathbf{M}_r \mathbf{M}_t \mathbf{T}_{ij}
\end{aligned} \tag{3.8}$$

最後に移動を加えた新たな頂点座標 $\mathbf{S}''_{ij}, \mathbf{T}'_{ij}$ から補間用 UV マップの三角形の頂点 $\mathbf{U}_{ij}$ を式 3.9 を用いて求める。

$$\mathbf{U}_{ij} = (1-t)\mathbf{S}''_{ij} + \mathbf{T}'_{ij} \tag{3.9}$$

ここまでの計算をソースモデルとターゲットモデルで対応関係にある全ての三角形に対して行い、補間用 UV マップの全ての頂点座標を決定する。しかし、ここまでの処理は対応する三角形ごとに計算しているため、変換前の UV マップで同じ UV 座標値にあり、複数の三角形で共有されているはずの頂点の座標が異なっていることがある。これを解決するために、本来共有している頂点同士で線形補間を行い頂点座標値を統合する。

これらの計算により三角形メッシュに含まれる三角形の特徴をなるべく維持し

た補間用 UV マップが作成できる。図 3.2 は図 3.1 の (a),(b) をソースモデルとターゲットモデルにし、本手法を用いて作成した UV マップである。図 3.1 の結果に比べて三角形の裏返りや潰れが起きていないのがわかる。

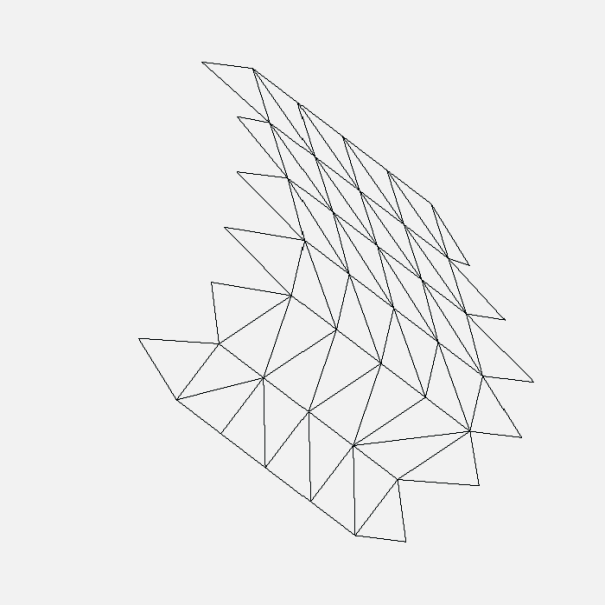

図 3.2: 提案手法を用いた補間用 UV マップ

3.2 モーフィング実行前の画像変換

UV マップの変換に伴って、テクスチャ画像も変換しなくてはならない。UV マップの UV 座標値に変換を加えて値を書き換えるということは、3DCG モデルの形状とテクスチャ画像の関係性が変更されているということである。テクスチャ画像にも変換を加えて、変換前の 3DCG モデルと同様にテクスチャ画像が表示するようにする。まず変換前の UV マップの UV 座標値に従って、変換前のテクスチャ画像を三角形に分割する。同様の方法で変換後の仮のテクスチャ画像も補間用 UV マップの UV 座標に従って三角形に分割する。変換前と変換後の対応する 1 つのテクスチャ画像の分割した三角形を考える。変換前の三角形内の 1 つのピクセルの色情報を、変換後のそれぞれの三角形を構成する 3 頂点から相対的に同じ位置にコピーす

る。画像変換のアルゴリズムは秋元の三角メッシュに分割した画像の合成の手法 [19] を基にしたものを利用する。変換前の三角形の 3 頂点を $(\mathbf{J}_{ix}, \mathbf{J}_{iy})(i = 0, 1, 2)$、変換後の三角形の 3 頂点を $(\mathbf{K}_{ix}, \mathbf{K}_{iy})(i = 0, 1, 2)$、変換前の三角形内のある点を $(\mathbf{J}_x, \mathbf{J}_y)$、変換後の三角形内の対応する点を $(\mathbf{K}_x, \mathbf{K}_y)$ とすると点 $(\mathbf{K}_x, \mathbf{K}_y)$ は次の式 3.10 で表すことができる。

$$\begin{pmatrix} \mathbf{K}_{0x} - \mathbf{K}_{2x} & \mathbf{K}_{1x} - \mathbf{K}_{2x} & \mathbf{K}_{2x} \\ \mathbf{K}_{0y} - \mathbf{K}_{2y} & \mathbf{K}_{1y} - \mathbf{K}_{2y} & \mathbf{K}_{2y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{K}_x \\ \mathbf{K}_y \\ 1 \end{pmatrix} \tag{3.10}$$

この α, β を解き、式 3.11 を用いることによって変換後の点 $(\mathbf{K}_x, \mathbf{K}_y)$ に対応する変換前の点 $(\mathbf{J}_x, \mathbf{J}_y)$ を求めることができる。

$$\begin{aligned} \mathbf{J}_x &= \alpha\mathbf{J}_{0x} + \beta\mathbf{J}_{1x} + (1 - \alpha - \beta)\mathbf{J}_{2x} \\ \mathbf{J}_y &= \alpha\mathbf{J}_{0y} + \beta\mathbf{J}_{1y} + (1 - \alpha - \beta)\mathbf{J}_{2y} \end{aligned} \tag{3.11}$$

対応する点 $(\mathbf{J}_x, \mathbf{J}_y), (\mathbf{K}_x, \mathbf{K}_y)$ の UV 座標値が求まれば、点 $(\mathbf{K}_x, \mathbf{K}_y)$ のピクセルに点 $(\mathbf{J}_x, \mathbf{J}_y)$ のピクセルの色情報をコピーすることによってテクスチャ画像の変換を行う。図 3.3 は補間用 UV マップ作成後のソースモデルの UV マップとテクスチャ画像、図 3.4 は補間用 UV マップ作成後のターゲットモデルの UV マップとテクスチャ画像である。それぞれ (a) が画像変換処理を加える前、(b) が画像処理変換を加えた後のものである。テクスチャ画像上の赤い丸と青い丸が対応する三角メッシュ内に移動しているのがわかる。

3.3 形状のモーフィング手法

3.1 節, 3.2 節によって変換処理を行った後はモーフィングの実行を行う。

形状のモーフィングについて述べる。形状のモーフィングの単純な方法は、対応関係にある頂点同士で直線的な線形補間を行うものである。モーフィング途中の形状を構成する頂点はこれらの対応関係を築いた頂点間に存在することになる。

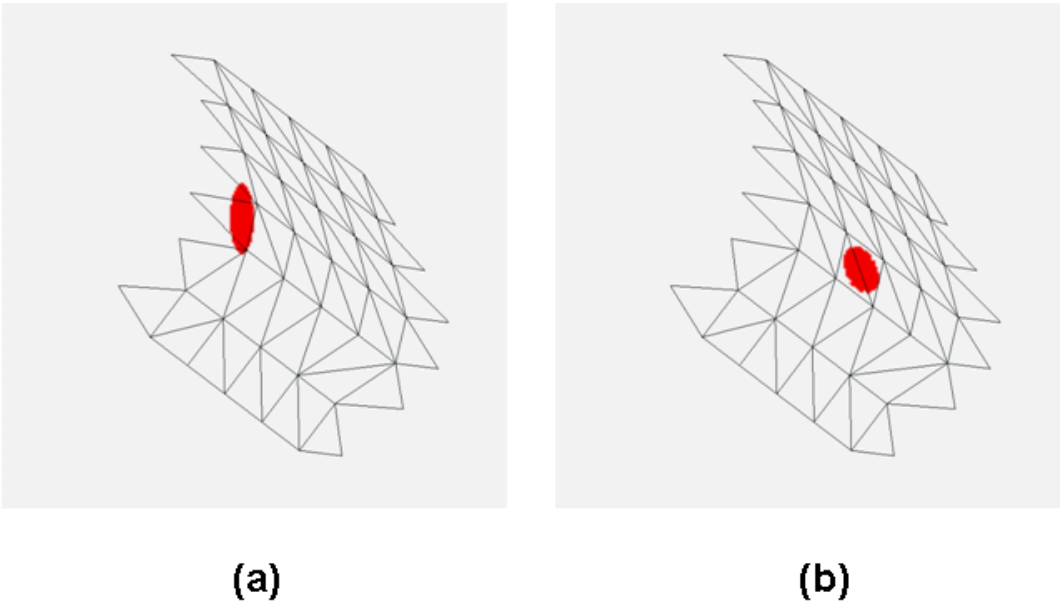

図 3.3: 画像変換前後の比較

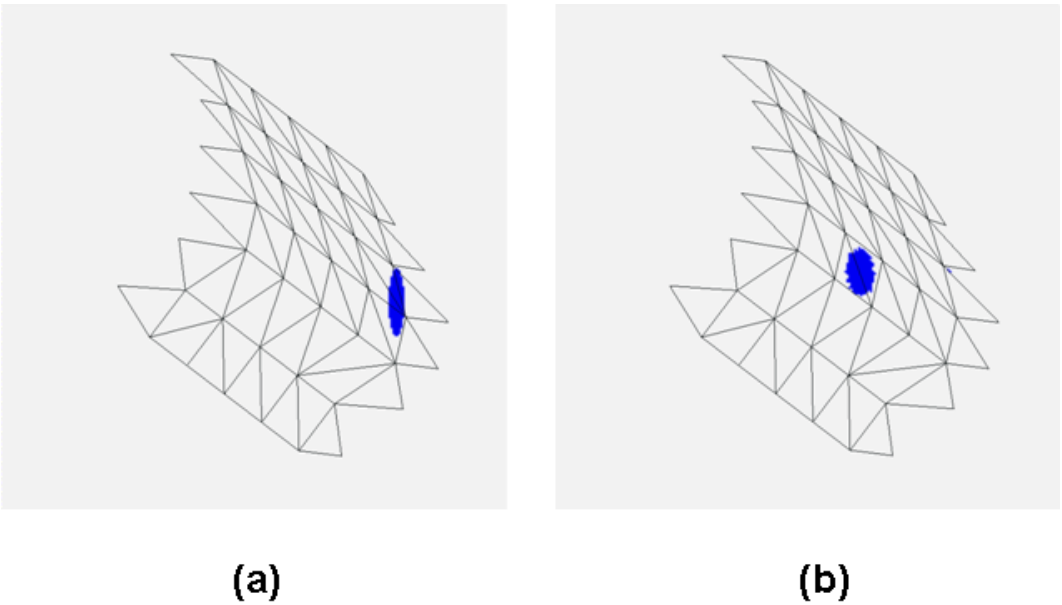

図 3.4: 画像変換前後の比較

経過時間 t の状態の頂点 $\mathbf{V}$ を座標値 $\mathbf{V}(t)$ とする。$\mathbf{V}(t)$ はソースモデルの座標値 $\mathbf{V}_0$ とターゲットモデルの座標値 $\mathbf{V}_1$ から式 3.12 を使って求めることができる。

$$\mathbf{V}(t) = (1-t)\mathbf{V}_0 + t\mathbf{V}_1 \tag{3.12}$$

図 3.5 は立方体、図 3.6 はその立方体の 1 つの頂点だけを移動して変形させたものである。図 3.5 をソースモデル、図 3.6 をターゲットモデルとして形状のモーフィングを行う。この例では赤い点で示された頂点同士が対応関係を結ぶ。この 2 つの形状をモーフィングすると図 3.7 のようになる。仮にソースモデルの赤く示した頂点座標を $(0,0,0)$ とし、ターゲットモデルの赤く示した頂点座標を $(0,10,0)$ とする。これに式 3.12 を用いると $t=0$ のときの、$\mathbf{V}(0)$ は $(0,0,0)$、$t=1$ のときの、$\mathbf{V}(1)$ は $(0,10,0)$ となり、$t=0$、$t=1$ はそれぞれソースモデル、ターゲットモデルの頂点座標値を示すことになる。$t=\frac{1}{2}$ のときの、$\mathbf{V}(\frac{1}{2})$ は $(0,5,0)$ になる。図 3.7 はその様子を示したものである。

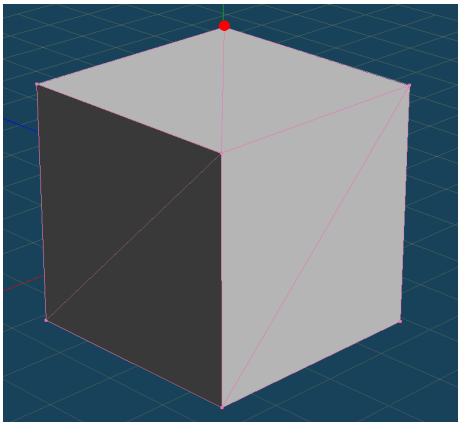

図 3.5: ソースモデル

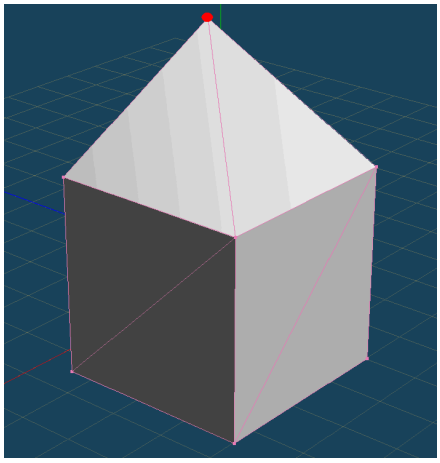

図 3.6: ターゲットモデル

形状全体にモーフィングを行う場合は、これ計算を全ての頂点に対して、同時に行うことにより形状の変形を行う。図 3.8 は左にソースモデルの猫、右にターゲットモデルの象を用いて中間形状を表したものである。中央は経過時間 $t=\frac{1}{2}$ のときの中間形状である。

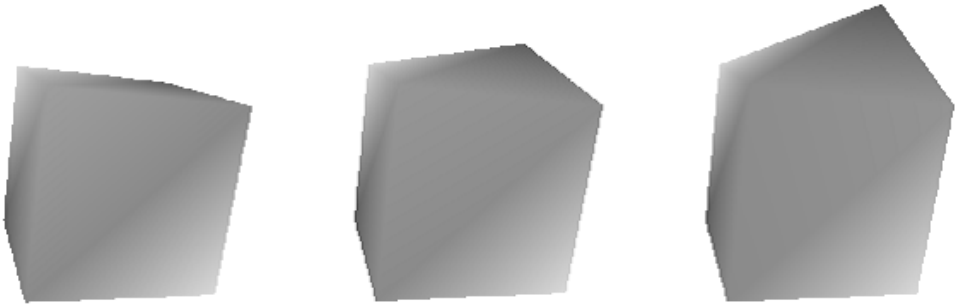

図 3.7: モーフィング例

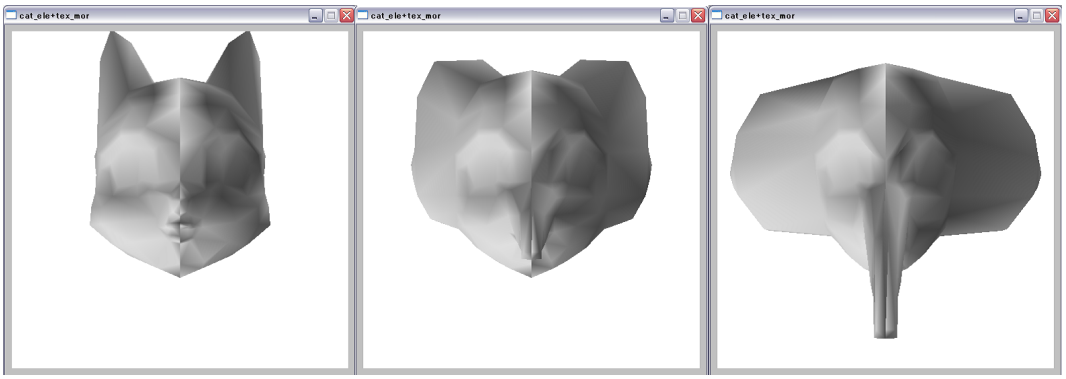

図 3.8: 形状モーフィング

3.4　テクスチャ画像のモーフィング手法

形状モーフィングと同時にテクスチャ画像のモーフィングも行う。テクスチャ画像はモーフィング実行前に UV マップとともに補間用に変換しているので、ソースモデルとターゲットモデルのそれぞれのテクスチャ画像で同じ UV 座標値にあるピクセルの色情報を使用する。ソースモデルのあるピクセルの R 値、G 値、B 値をそれぞれ R_s, G_s, B_s として、それに対応するターゲットモデルのあるピクセルの R 値、G 値、B 値をそれぞれ R_t, G_t, B_t とする。時間 t のときのテクスチャ画像のあるピクセルの R 値、G 値、B 値を R_r, G_r, B_r とすると、R_r, G_r, B_r は次の式 3.13 で求める。

$$\begin{aligned} R_r &= (1-t)R_s + tR_t \\ G_r &= (1-t)G_s + tG_t \\ B_r &= (1-t)B_s + tB_t \end{aligned} \tag{3.13}$$

図 3.9 は黒と黄色の縦縞のテクスチャ画像と横縞のテクスチャ画像をモーフィングさせたものである。左が縦縞、右が横縞のテクスチャ画像。中央は $t = \frac{1}{2}$ のときのテクスチャ画像である。$t = \frac{1}{2}$ のときは縦縞、横縞が合わさって、格子状になっているのがわかる。

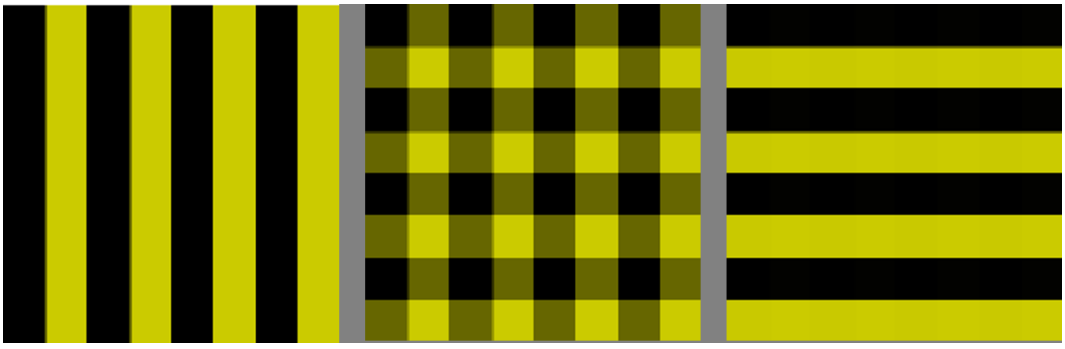

図 3.9: テクスチャ画像のモーフィング

このテクスチャ画像のモーフィングを形状のモーフィングと同時に行うことによって、テクスチャ付きの 3DCG モデルモーフィングを実行する。

3.5　本章のまとめ

本章では、モーフィング実行前に UV マップ、テクスチャ画像を変換することでテクスチャ付きの 3DCG モデルモーフィングを実行する手法を提案した。図 3.10 は UV マップ、テクスチャ画像の変換の流れを示したものである。形状はソースモデルターゲットモデル共に変換作業を行わない。UV マップはソースモデルとターゲットモデルの情報から、補間用 UV マップを新たに作成する。テクスチャ画像は新たに作成した補間用 UV マップの情報に従い、補間用のテクスチャ画像に変換を行う。

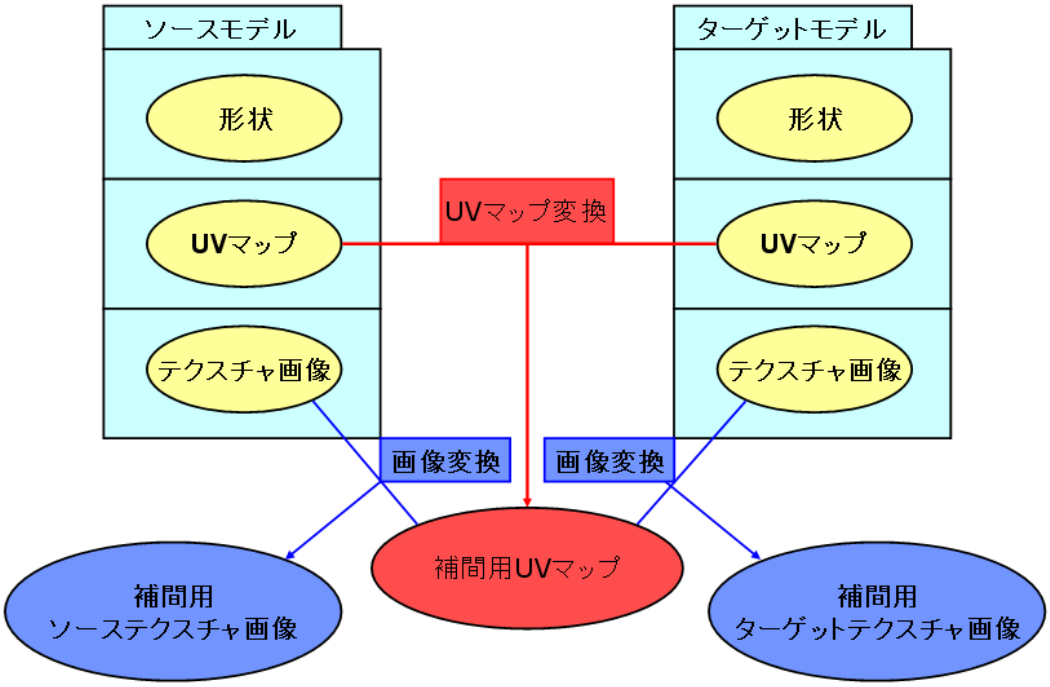

図 3.10: UV マップ、テクスチャ画像の変換の流れ

図 3.11 はテクスチャ付きの 3DCG モデルモーフィング実行時のソースモデルとターゲットモデル間の関係を示したものである。形状はソースモデルとターゲットモデル共に元々保持しているデータを使用し、ソースモデルとターゲットモデル間で時間経過に伴って形状の補間を行う。UV マップはソースモデルとターゲッ

トモデル共に、モーフィング実行前に行った処理で作成した補間用 UV マップを使用する。そのためソースモデルとターゲットモデルの UV マップは同一のため、補間を行う必要はない。テクスチャ画像はモーフィング実行前に行った変換で作成した補間用のテクスチャ画像を使用する。補間用のテクスチャ画像はソースモデルとターゲットモデルが元々保持している画像を補間用 UV マップに適するように変換を加えたものである。そのため画像はソースモデルとターゲットモデルでは異なるので、時間経過に伴って色情報の補間を行う。

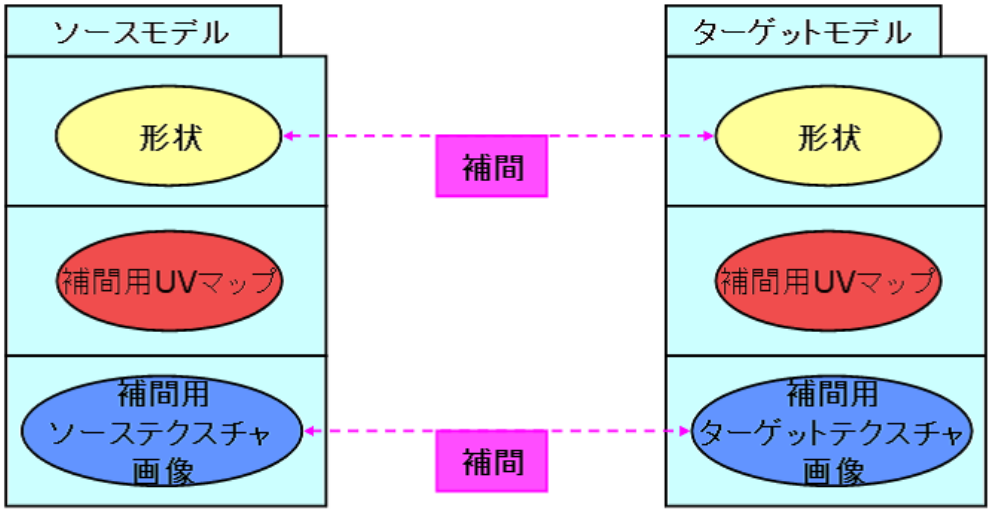

図 3.11: 変換後のモデル間の関係

第 4 章

実装と評価

この章では本研究の提案手法の実装結果とその検証、考察を行う。

4.1 実装方法

本研究の提案手法を実装した。実装にはC++言語を用い、グラフィックスライブラリにはOpenGLの上位ライブリである渡辺らのFKシステム[20]を用いて作成した。三角メッシュモデルのデータ構造はインデックスフェースセットを用いた。インデックスフェースセットとは、頂点座標値と面を構成する頂点の番号によりメッシュを表現する。これを三角形だけで構成することにより三角メッシュモデルを表現する。本研究は、モーフィングの際のテクスチャ画像について焦点を当てた。そのため用意する3DCGモデルにあらかじめ制限を設け、形状に関する処理を簡易にした。1つ目はソースモデルとターゲットモデルの頂点数、位相はは同じものを用意した。これにより形状モーフィングにおける対応問題を避けた。2つ目はなるべく簡易な形状のモデルを用意した。これにより形状モーフィングの最中に起こる自己干渉や裏返りなどの補間問題を避けた。また、UVマップの作成方法はソースモデル、ターゲットモデル共に、同じように切り開いた展開図を用いた。

3DCGモデルの作成にはMetasequoia[21]を使用した。Metasequoiaとは、3DCGデータのモデリングソフトである。一般的な3次元ソフトのファイル形式を扱えるほか、Metasequoia専用のファイル形式がある。Metasequoia専用ファイル形式はMQOファイルという。実装に使用した3DCGモデルのデータはこのMQOファイルである。モーフィングに使用したのは直方体と球体の三角メッシュモデルで、共に頂点数32、面数60である。テクスチャ画像は、縦256ピクセル、横256ピクセルの画像を使用した。図4.1はテクスチャ画像を貼り付けた状態の直方体のモデル、図4.2は直方体のモデルに使用しているテクスチャ画像とUVマップである。図4.3はテクスチャ画像を貼り付けた状態の球体のモデル、図4.4は球体のモデルに使用しているテクスチャ画像とUVマップである。

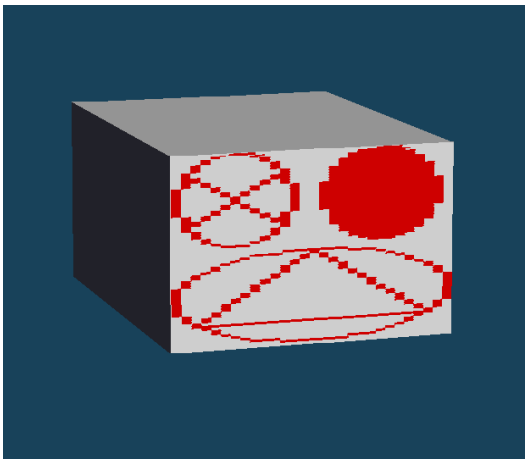

図 4.1: 直方体のモデル

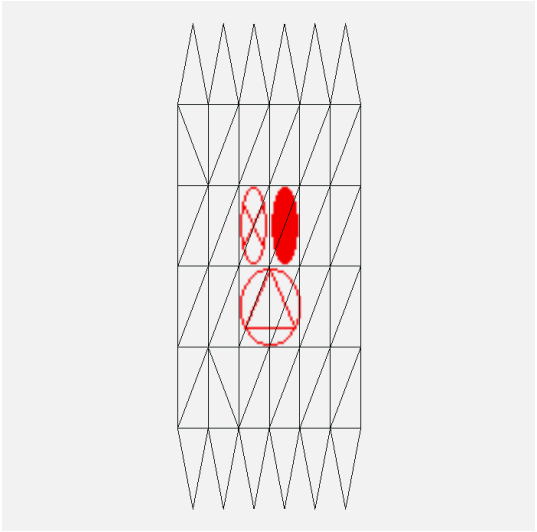

図 4.2: 直方体のモデルのテクスチャ画像

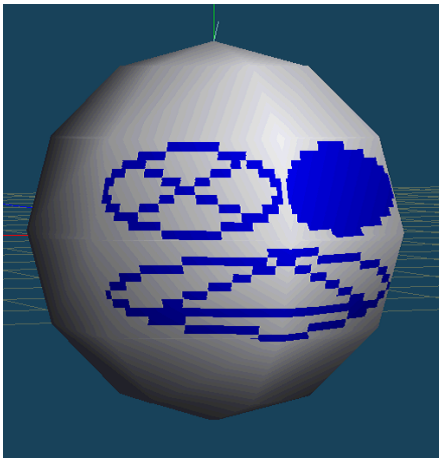

図 4.3: 球体のモデル

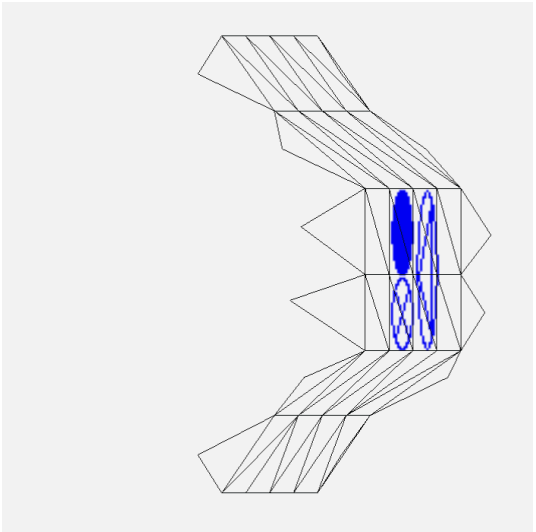

図 4.4: 球体のモデルのテクスチャ画像

4.1.1 UV マップ、テクスチャ画像変換部分

モーフィング実行前に補間用の UV マップとテクスチャ画像を作成するための実装について述べる。変換部分は、ソースモデルとターゲットモデルの形状、テクスチャ画像、UV マップをそれぞれ読み込む。読み込んだ後は、ソースモデルとターゲットモデルの UV マップから本研究の手法を用いて補間用 UV マップを作成する。その後作成した補間用 UV マップとソースモデルの UV マップを比較し、その結果からソースモデルのテクスチャ画像を変換する。これを同様にターゲットモデルのテクスチャ画像にも行う。補間用 UV マップとテクスチャ画像の変換が終わった後は、ソースモデル、ターゲットモデルそれぞれの UV マップとテクスチャ画像を変換したものに置き換える。

4.1.2 モーフィング実行部分

モーフィング実行の実装について述べる。用意したソースモデルとターゲットモデルはあらかじめ、自己干渉や裏返りが起きないように作成したため、形状のモーフィングは直線的な線形補間を行った。実装の結果、時間が t のときの形状のモーフィングは図 4.5 のようになった。

4.2 実行結果

直方体と球体のモデルのそれぞれ UV マップから補間用 UV マップを作成した結果は図 4.6 のようになった。三角形の裏返りや潰れがなく作成することができた。

補間用 UV マップに適したように画像変換をした結果は直方体のモデルは図 4.7、球体のモデルは図 4.8 のようになった。同じメッシュの位置に対応した印が描かれているのがわかる。

これらの処理を行った後に実行したモーフィング結果は図 4.9 のようになった。時間 $t = 0$ のときは直方体モデル、$t = 1$ のときは球体モデルになっている。また、

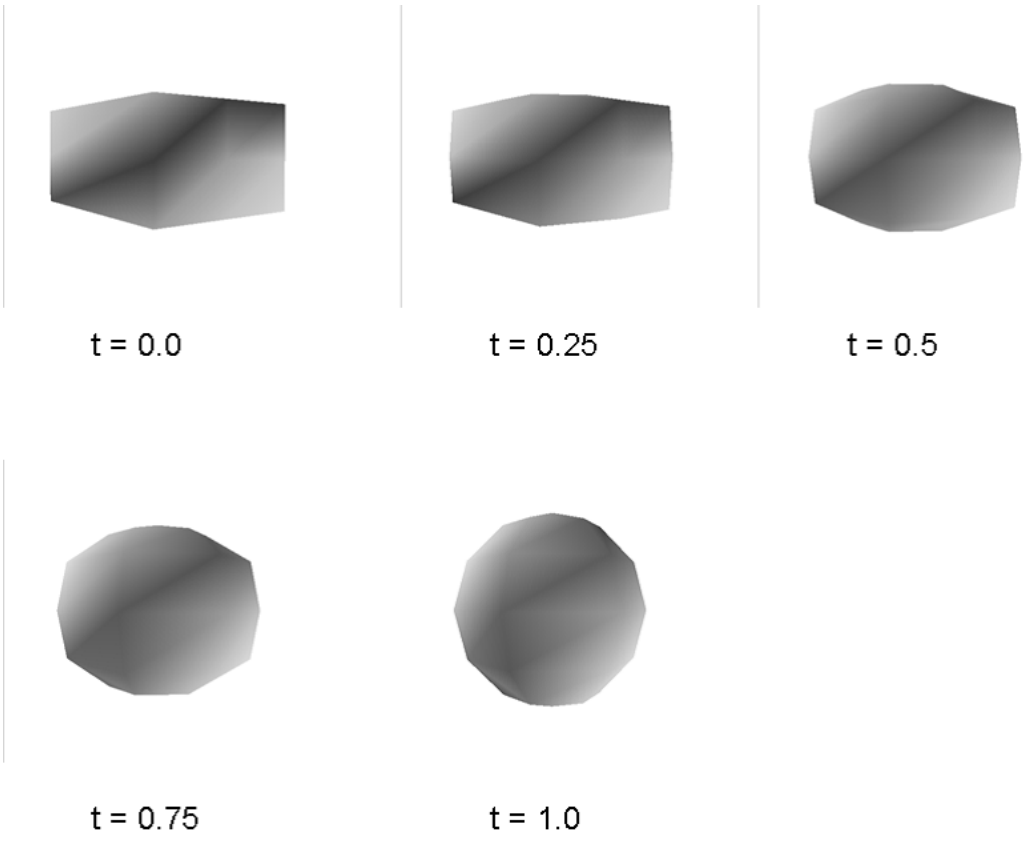

図 4.5: 形状モーフィングの結果

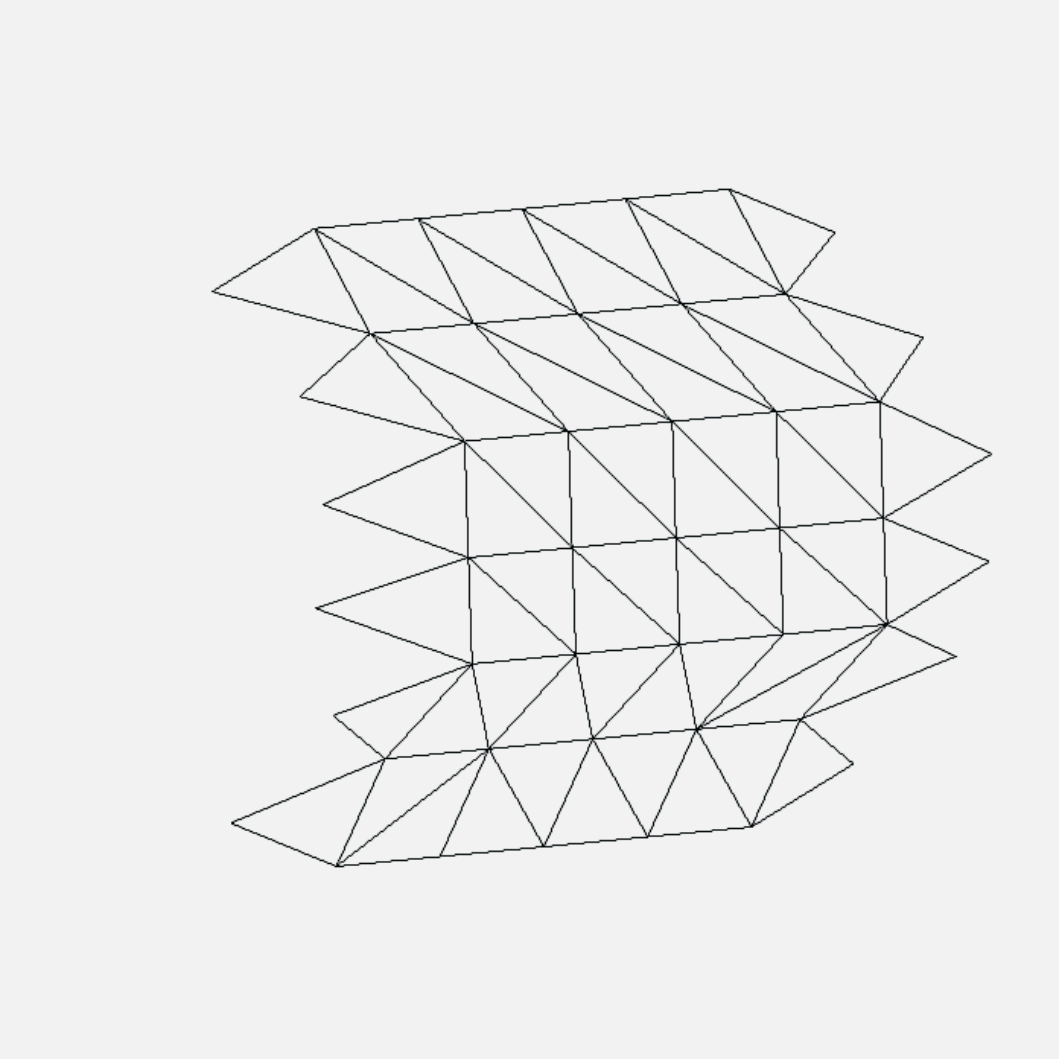

図 4.6: 作成した補間用 UV マップ

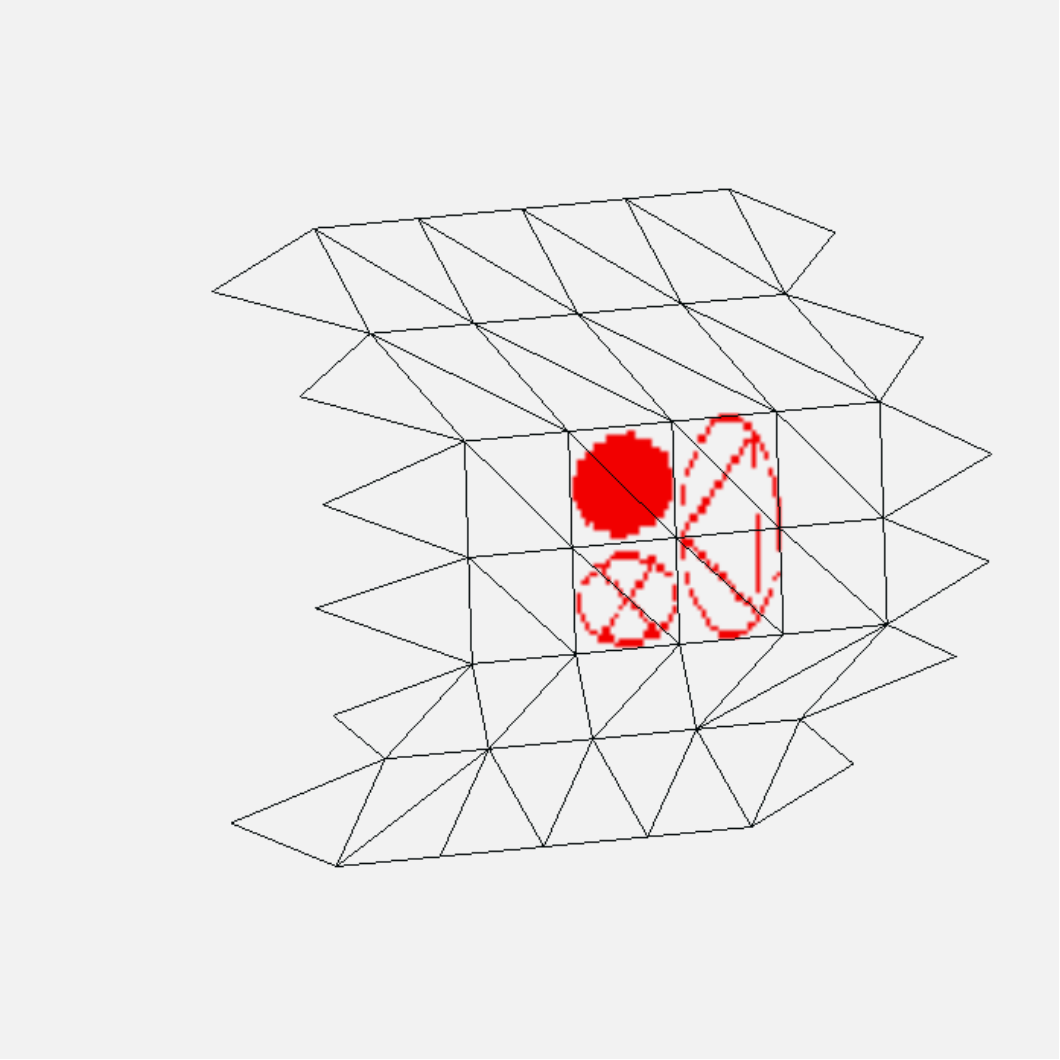

図 4.7: 画像変換後の直方体モデルのテクスチャ画像

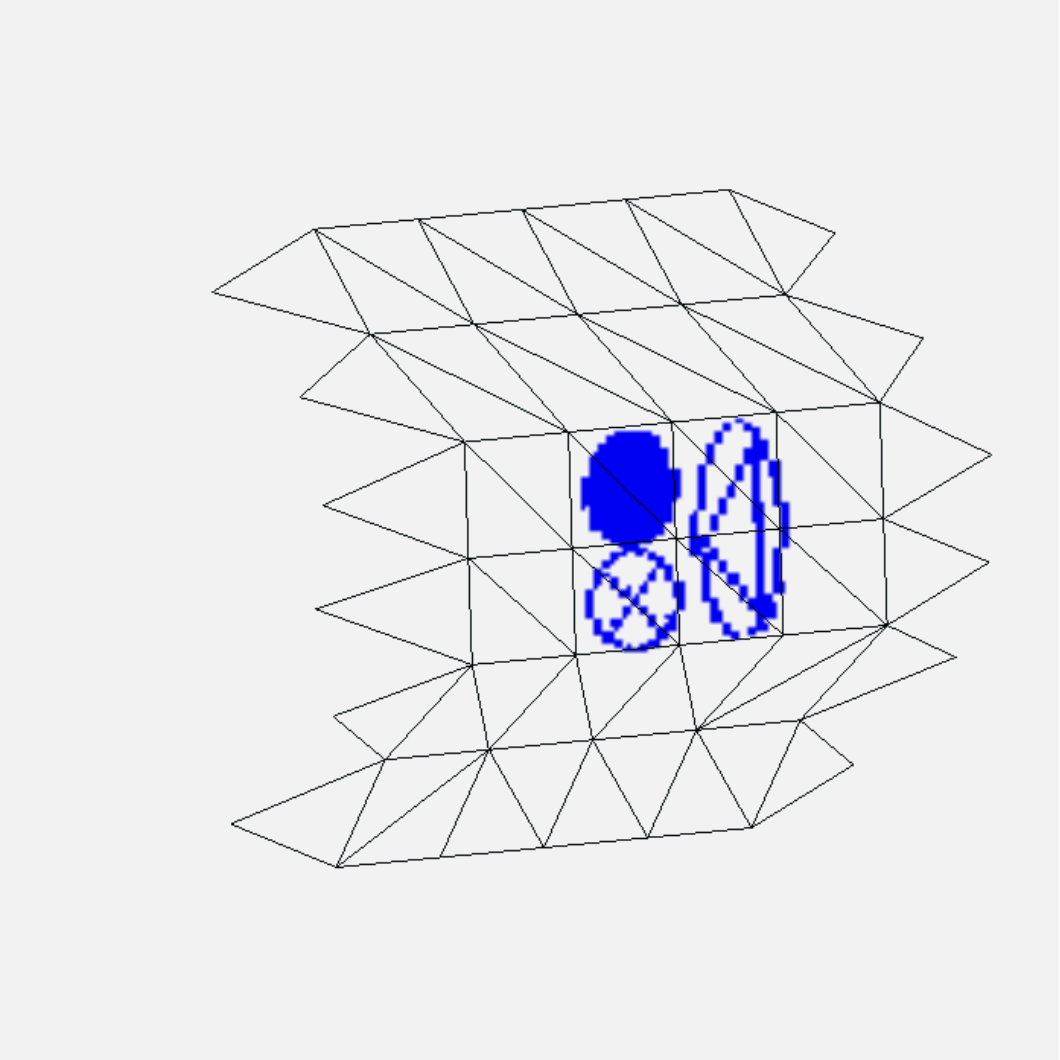

図 4.8: 画像変換後の球体モデルのテクスチャ画像

経過途中のモデルはテクスチャの特徴箇所がおおよそ同じ位置に表れていて、モーフィング実行前の処理が上手くいっているのがわかる。

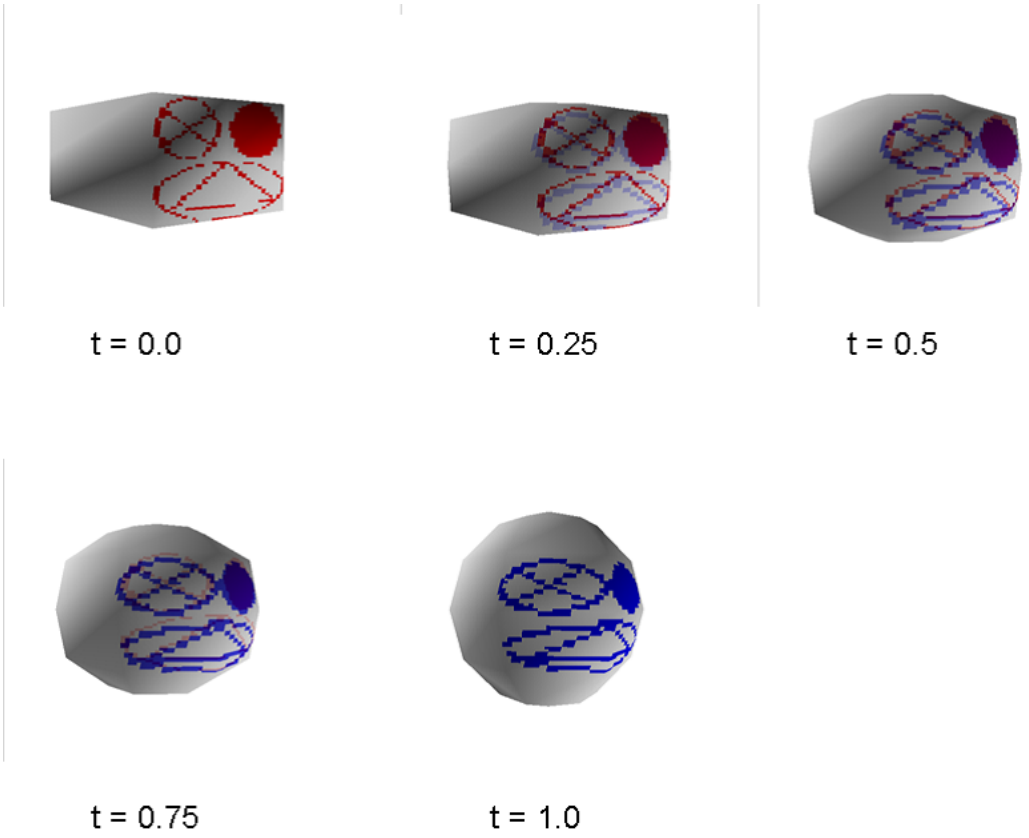

図 4.9: モーフィング実行結果

頂点数、面数、テクスチャ画像サイズ、形状を変えてもう 1 つ試してみた。使用したモデルは三角メッシュモデルで、共に頂点数 242、面数 480 である。テクスチャ画像は、縦 512 ピクセル、横 512 ピクセルの画像を使用した。図 4.10 はテクスチャ画像を貼り付けた状態のソースモデル、図 4.11 はソースモデルに使用しているテクスチャ画像と UV マップである。図 4.12 はテクスチャ画像を貼り付けた状態のターゲットモデル、図 4.13 はターゲットモデルに使用しているテクスチャ画像と UV マップである。

ソースモデルとターゲットモデルのそれぞれ UV マップから補間用 UV マップを作成した結果は図 4.14 のようになった。こちらも三角形の裏返りや潰れがなく

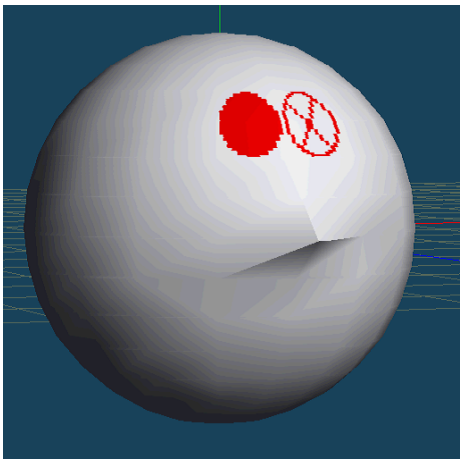

図 4.10: ソースモデル

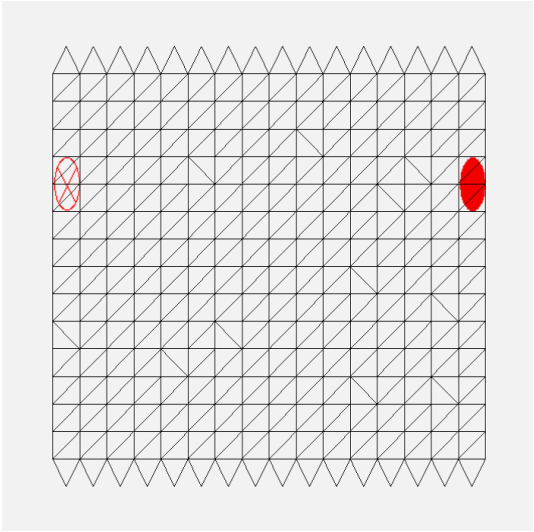

図 4.11: ソースモデルのテクスチャ画像

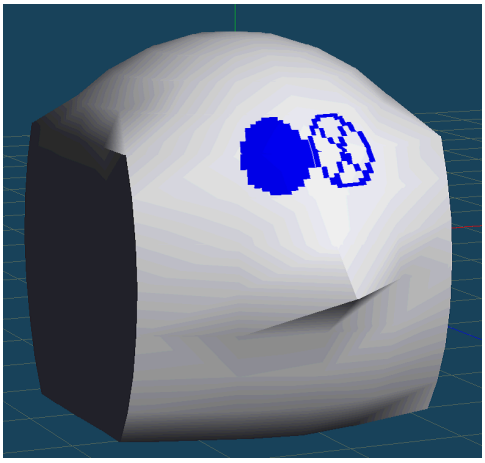

図 4.12: ターゲットモデル

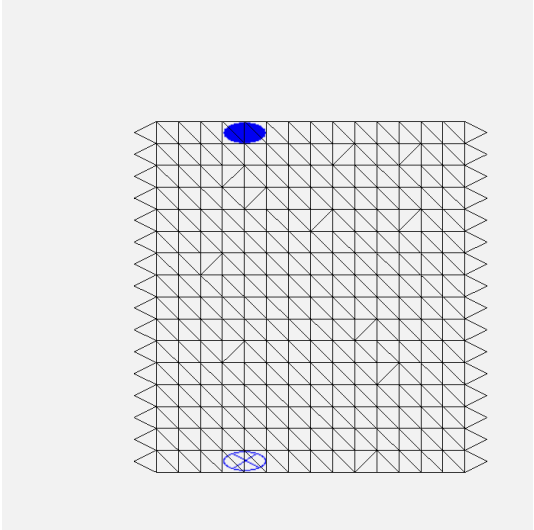

図 4.13: ターゲットモデルのテクスチャ画像

作成することができた。

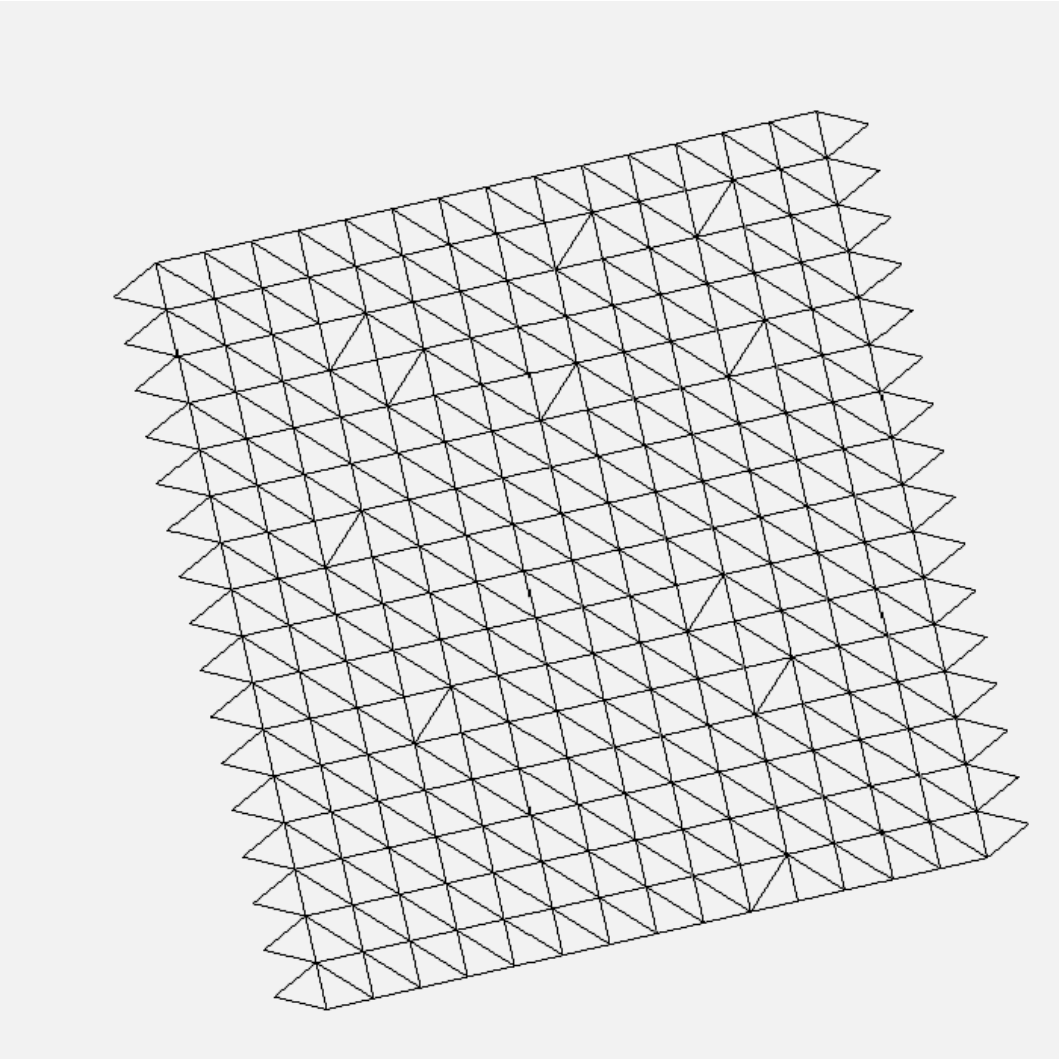

図 4.14: 作成した補間用 UV マップ

補間用 UV マップに適したように画像変換をした結果はソースモデルは図 4.15、ターゲットモデルは図 4.16 のようになった。同じメッシュの位置に対応した印が描かれているのがわかる。

これらの処理を行った後に実行したモーフィング結果は図 4.17 のようになった。時間 $t = 0$ のときはソースモデル、$t = 1$ のときはターゲットモデルになっている。また、経過途中のモデルはテクスチャの特徴箇所がおおよそ同じ位置に表れていて、モーフィング実行前の処理が上手くいっているのがわかる。

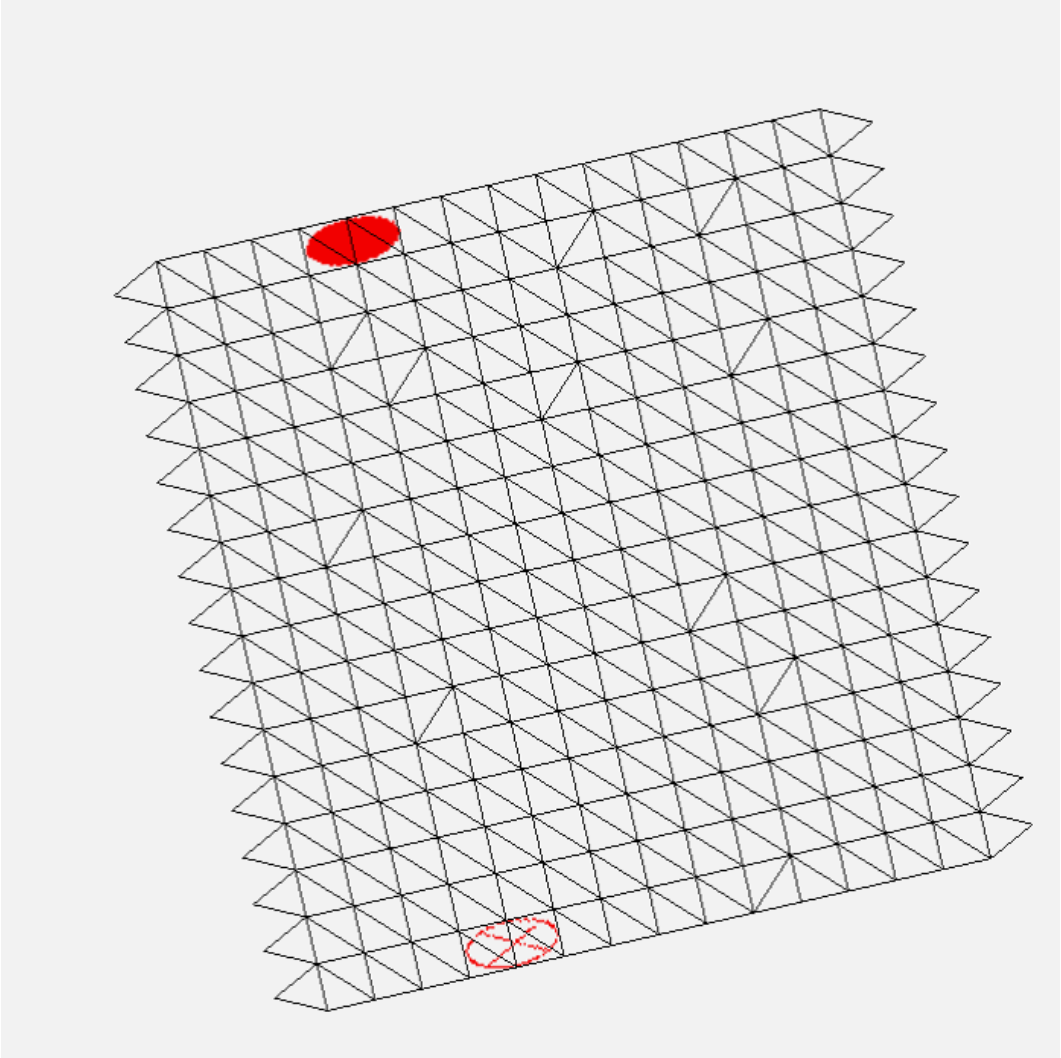

図 4.15: 画像変換後のソースモデルのテクスチャ画像

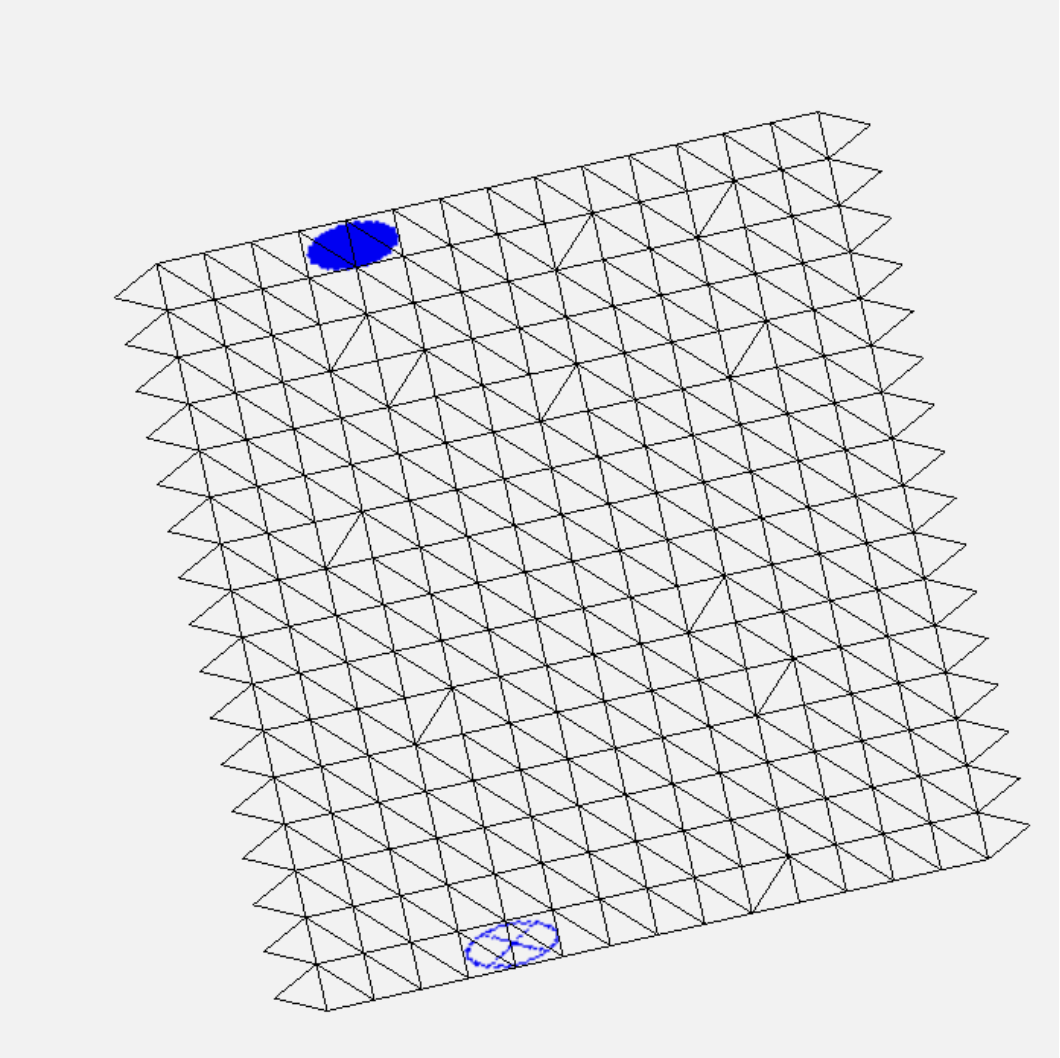

図 4.16: 画像変換後のターゲットモデルのテクスチャ画像

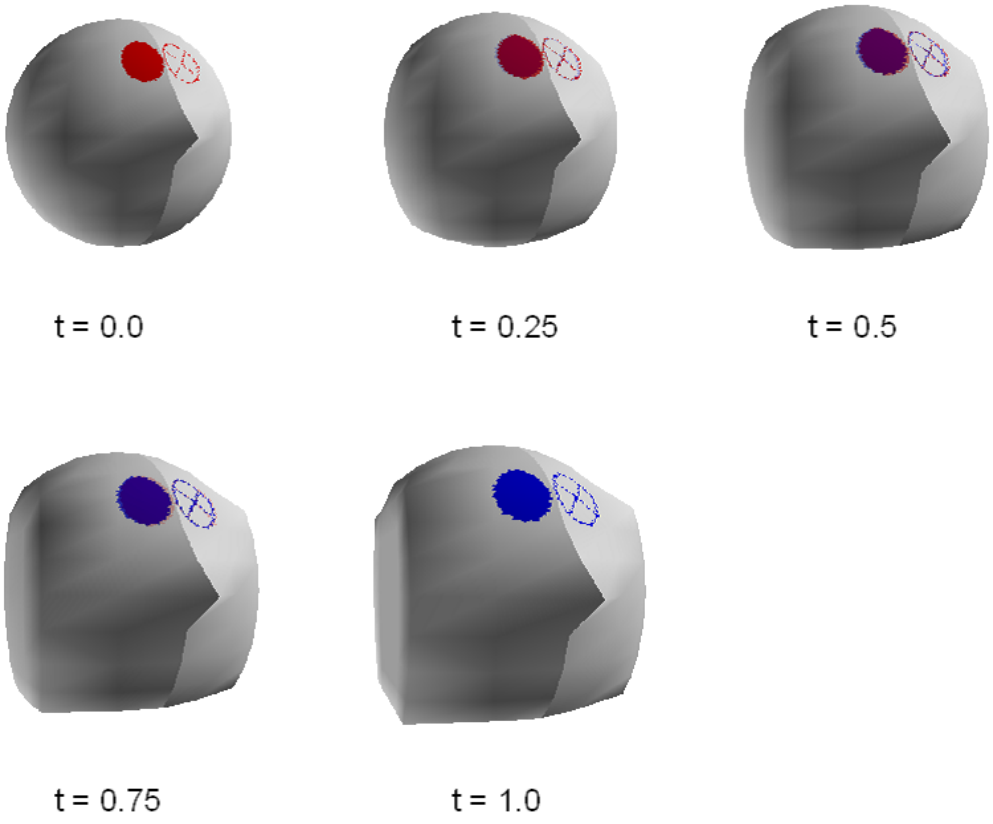

図 4.17: モーフィング実行結果

本研究の提案手法によって、UV マップや画像の位置が異なっていてもモーフィング経過中に特徴箇所のテクスチャ画像がずれることなく、モーフィングを行うことができた。

4.3 問題点

時間 $t = 0$ のときはソースモデルである直方体のモデル、時間 $t = 1$ のときはターゲットモデルである球体のモデルと同一であることが良い。図 4.18 の (a) はソースモデルの補間用に何も処理を加えていない状態、(b) はソースモデルの補間用に処理を加えた状態である。図 4.19 も同様に (a) はターゲットモデルの補間用に何も処理を加えていない状態、(b) はターゲットモデルの補間用に処理を加えた状態である。図 4.18、図 4.19 でそれぞれ (a) と (b) の状態も見比べてみると、補間用に処理を加えた状態はテクスチャ画像の線が掠れていたり、曲線が歪んでいたりするのが表れている。これはテクスチャ画像を補間用 UV マップに適するように変換を行った処理に原因がある。

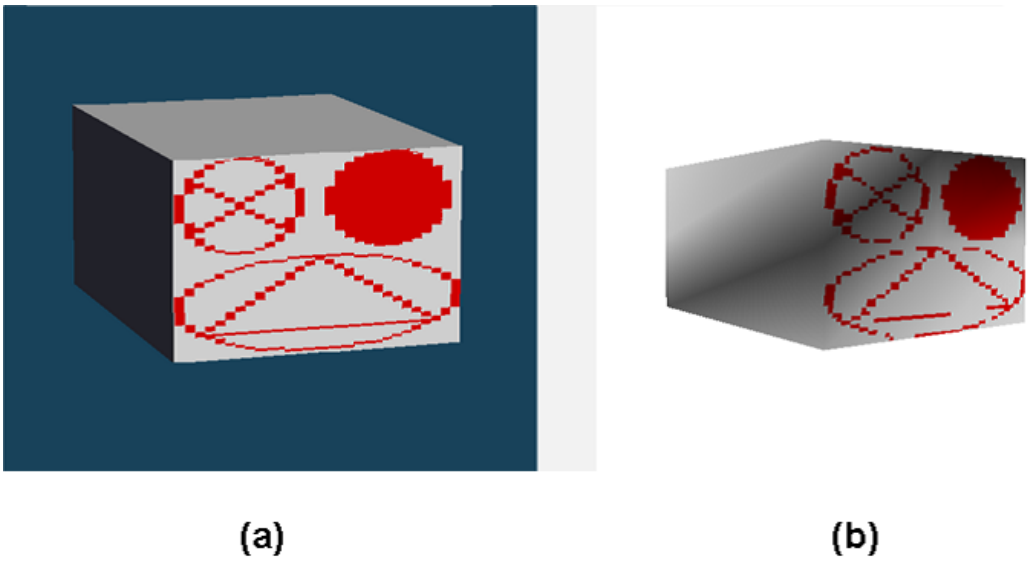

図 4.18: ソースモデルの補間用処理の前後

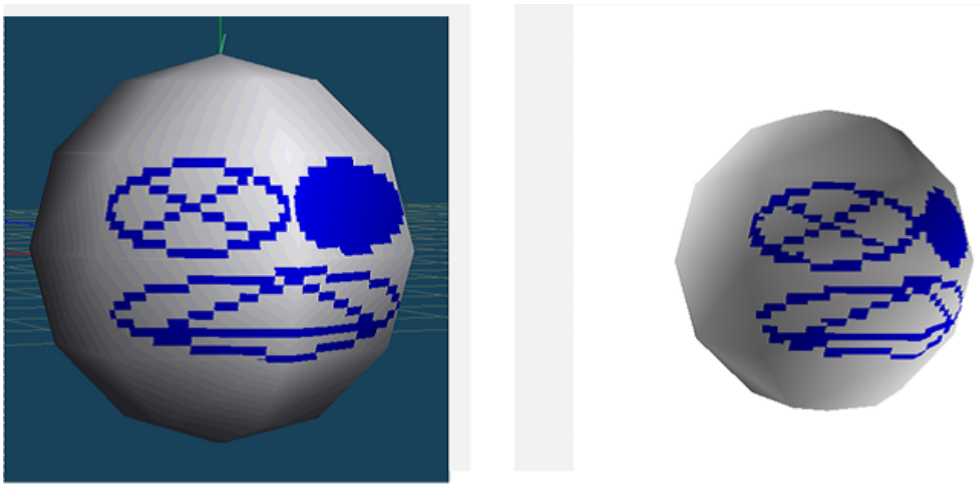

図 4.19: ターゲットモデルの補間用処理の前後

第 5 章

まとめ

5.1 本研究のまとめ

本研究ではモーフィングを行うことを前提としてUVマップとテクスチャ画像を作成していない2つのモデルをテクスチャの特徴箇所を維持したままモーフィングを行う手法に取り組んだ。モーフィング実行前にUVマップとテクスチャ画像をモーフィング用に変換を加えることで実現した。本研究によって、3DCGモデルを使用したモーフィングがより幅広く使うことができるだろう。

5.2 今後の展望

補間用に画像変換処理を行ったときに、処理を行う前に比べテクスチャ画像の線が掠れていたり、曲線が歪んでしまう問題については、2つの解決方法が考えることができる。1つはテクスチャ画像の解像度自体を上げてしまうことで、ある程度の掠れや歪みを防ぐことができる。しかし、テクスチャ画像の解像度を上げるにつれて、補間用にテクスチャ画像を変換する際とモーフィング実行時に処理量が増えてしまう。特にモーフィング実行時に処理量が増えてしまうと、滑らかな変化を表現することができなくなってしまう。2つ目はテクスチャ画像の変換方法を変えることである。テクスチャ画像の変換方法を画像処理の分野で使うバイリニア補間やバイキュービック補間 [22] などにすることで、改善できるだろう。

形状のモーフィングは変形技術の1つであるといえる。変形技術を応用することで、モーフィングの視覚的効果はさらに増すことができる。Alexa[23] は頂点を近傍頂点で構成する多角形の重心と差分によるラプラシアン表現を用いて、形状の局所的な変形を可能にした。道川ら [24] は多重解像度表現を補間メッシュに応用し、頂点数や面数が異なる形状間でも高品質なモーフィングを行い、効率的なデータの保持を実現した。このような局所的な変形や形状に変換を加える手法には、その都度UVマップを書き換えなければ適切なテクスチャの描画が行えない。

また、UVマップの作成方法も作者によりことなる。ソースモデルとターゲットモデル間でUVマップとテクスチャ画像の構成を同一にする手法を考案すること

で、テクスチャ付きのモーフィングが可能になる。これには形状の頂点を基に UV マップとテクスチャ画像を再構成することが考えられる。

本研究では UV マップとテクスチャ画像に焦点を当てたために、ソースモデルとターゲットモデルのモーフィングに必要な形状の条件はあらかじめ揃えておいた。1.1 節で挙げたように、形状のモーフィングを行うためにソースモデルとターゲットモデルを再構成する研究がある。これらの研究と本研究の手法を組み合すことができれば、モーフィングを行うことを前提として 3DCG モデルを作成する必要がなくなり、すでに作成してある 3DCG モデルを自由にソースモデルとターゲットモデルにしてテクスチャ付の 3DCG モーフィングが行えるようになる。それにより 3DCG モデルの利用方法や表現方法がさらに幅広くなるだろう。

謝辞

本研究を進めるにあたり、温かいご支援、ご指導いただきました東京工科大学メディア学部の渡辺大地講師に心より感謝いたします。

日頃から本研究のサポートをしていただいた、研究室のメンバーに厚く御礼申し上げます。

本研究にご協力していただいたすべての皆様に心から感謝致します。

参考文献

[1] Thaddeus Beier, Shawn Neely, ”Feature-Based Image Metamorphosis”, Siggraph92, Vol.26, No.2, pp35-42, 1992.

[2] GAME Watch, ”3D ゲームファンのための PSP グラフィックス講座”, http://watch.impress.co.jp/game%2Fdocs/20040406/psp.htm/.

[3] 寺沢幹雄, 坂倉守昭, 小高金次, ”Web2.5D:実時間視点モーフィングを用いた Web 用立体表示”, インタラクション 2001 p.133-134, 2001.

[4] 水田忍, 安中英邦, 船冨卓哉, 美濃導彦, ”軌跡を考慮した 3 次元メッシュモーフィングによるヒト胎児の成長表現”, 情報処理学会研究報告 グラフィクスと CAD 研究会報告 2001-CG-104-11pp. 41-46, 2001.

[5] 別宮喬, 真鍋知久, 金田和文, ”照明条件の異なる画像を用いたイルミネーションモーフィング手法”, 情報処理学会研究報告 2004-CG-117, 2004.

[6] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, コンピュータグラフィックス 理論と実践, (佐藤義雄 訳), オーム社, 2001.

[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”, グラフィクスと CAD, 1995.

[8] Arthur Gregory, Andrei State, Ming C. Lin, Dinesh Manocha, Mark A. Livingston, ”Feature-based Surface Decomposition for Correspondence and Morphing between Polyhedra”, In Proceedings of the Computer Animation, pp. 64-71. IEEE Computer Society, 1998.

[9] 川井雅典, 大渕竜太郎, ”スペクトル分解を用いた 3 次元メッシュのモーフィング”, 情報処理学会研究報告 グラフィクスと CAD 研究会報告 2001-CG-105-7 pp. 33-38, 2001.

[10] 舩富卓哉, 飯山将晃, 水田忍, 美濃導彦, ”局所的自己交差を回避する 3 次元パッチモデルモーフィングの軌跡生成手法”, 情報科学技術フォーラム 2002, No.3 J-42, p.285-286, 2002.

[11] 道川隆士, 金井喬, 鈴木宏正, ”多重解像度補間メッシュにおける近似剛体補間”, グラフィクスと CAD 合同シンポジウム 2004, 2004.

[12] 白石路雄, 山口泰, ”3 次元的な傾斜を用いた絵画風画像モーフィング”, 情報科学技術フォーラム 2002, 2002.

[13] 白濱洋平, 大野義夫, ”多数決にもとづくアニメーションキャラクターのモーフィング手法”, グラフィクスと CAD, 2001.

[14] Autodesk, ”Maya”, http://www.autodesk.co.jp/.

[15] Autodesk, ”3ds Max”, http://www.autodesk.co.jp/.

[16] 平岡和幸, 堀玄, プログラミングのための線形代数, オーム社, 2004.

[17] Han-Bing Yan, Shi-Min Hu, Ralph Martin, ”Morphing Based on Strain Field Interpolation”, Computer Animation and Virtual Worlds, 2004.

[18] Marc Alexa, Daniel Cohen-Or, David Levin, ”As-Rigid-As-Possible Shape Interpolation”, In Proceeddings of the 27th annual conference on Computer graphics and interactive techniques, pp. 157-164, 2000.

[19] 秋元毅, ”モーフィング処理による顔表情変化シミュレーション”, 東京工科大学 卒業論文, 2004.

[20] 渡辺大地, ”FK Toolkit System”, http://www.teu.ac.jp/aqua/ earth/FK/.

[21] Q.Mizno, ”Metasequoia”, http://www.metaseq.net/.

[22] ディジタル画像処理編集委員会, ディジタル画像処理, CG-ARTS 協会, 2006.

[23] Marc Alexa. ”Local control for mesh morphing”, In Proceedings of the International Conference on Shape Modeling and Applications, pp. 209-215, 2001.

[24] 道川隆士, 金井崇, 藤田将洋, 千代倉弘明, ”多重解像度補間メッシュ”, 2001 年度画像電子学会, 2001.