修士論文　　平成 22 年度 (2010)

ベクター形式による
変位マップアニメーションに関する研究

東 京 工 科 大 学 大 学 院
バイオ・情報メディア研究科 メディアサイエンス専攻

武田 巧視

修士論文　　平成 22 年度 (2010)

ベクター形式による
変位マップアニメーションに関する研究

指導教員　渡辺 大地 講師

東 京 工 科 大 学 大 学 院
バイオ・情報メディア研究科 メディアサイエンス専攻

武田 巧視

論 文 の 要 旨

論文題目	ベクター形式による 変位マップアニメーションに関する研究
執筆者氏名	武田 巧視
指導教員	渡辺 大地 講師
キーワード	変位マッピング、局所変形、メッシュモーフィング、メッシュ合成

[**要旨**]

コンピュータが一般に普及し、3DCG に関する技術は大きく向上した。ビデオゲームや映像作品におけるリアルタイム 3DCG では、ポリゴンメッシュ(以下、メッシュ) を、一般的な形状表現として用いる。メッシュで表現したキャラクタなどのアニメーションは、メッシュを構成する頂点の位置を変更することでメッシュの形状を変えてアニメーションを行うのが一般的である。しかし、この手法はループの接続関係などの位相が変化することを想定していない。本研究では、メッシュアニメーションが苦手とする、メッシュが部分的、局所的に大きく形状変形する変形アニメーションに着目した。位相変化せずに変形アニメーションを実現する場合、変形後の形状を表現可能な位相を事前に用意しておく必要がある。変形形状が複雑であれば、それだけメッシュのループ数は大量になり、変形アニメーション処理も煩雑になる。どこが変形するか事前に分かっているのであれば、位相変化せずにアニメーションを行うことが可能だが、変形する場所が不定の場合は動的な位相変化が必要になるため、リアルタイムにアニメーションを行うことが難しい。

そこで本研究では、位相変化を簡略化することで、局所的に複雑な変形が発生した、局所的形状変形手法の実現を目的とする。本手法では、3 次元メッシュである付加メッシュを、面や稜線が重ならないように 2 次元状になるように頂点座標を操作し、2 次元メッシュを作成する。最終的に基礎メッシュと付加メッシュを合成して局所的形状変形を行う。2 次元メッシュの形状を正方形に限定することで、位相変化を伴う合成処理を簡略化した。本手法では、2 次元メッシュを変位マップとして扱うため、ラスター画像を生成する必要がない。提案手法をプログラムで実装し、検証を行い、提案手法の有用性を確認した。

A b s t r a c t

Title	Displacement Mapping Animation by Vector-format
Author	Takeda Takumi
Advisor	Lecturer Watanabe Taichi
Key Words	Displacement mappinng, local modification, mesh morphing, mesh marging

[**summary**]

Computers spread generally, and the technology about 3DCG improved greatly. Real-time 3DCG in video game and a picture work use polygon mesh. Character animation by polygon mesh deforms shape of the polygon mesh. But, This technique does not assume topology deformation. Mesh animation is weak in local deforming animation. This paper focuses local deforming animation of mesh. When not deform topoogy animation must prepare for topology after the deformation. If the shape that deformed is complicated number of the loops increases and deform animation becomes complicated, too. If I know a place to deform cuts it in animation without changing topology. But, When I do not know a place to deform deform of the topology is necessary. Therefore, It is difficult to animation in real time.

In this paper, I simplify deforming topology and aimed for realization of the local deforming technique. This technique convert 3D polygon mesh into 2D polygon mesh. Finally compose Based mesh and Add mesh. Limited shape of the 2D polygon mesh to a square. Therefore composition became simple. This technique do displacement map and treat 2D polygon mesh. Therefore it is not necessary to make raster image. I implemented suggestion technique by a program. I inspected it and confirmed utility.

目 次

図 目 次

第 1 章

はじめに

1.1 研究背景と目的

コンピュータが一般に普及し、その性能の向上と共に 3 次元コンピュータグラフィクス (以下、3DCG) を用いたコンテンツも増加している。3DCG コンテンツの増加に伴って 3DCG を用いた様々な表現が多く開発されるようになり、3DCG に関する技術は大きく発展した。ビデオゲームや映像作品におけるリアルタイム 3DCG では、ポリゴンメッシュ(以下、メッシュ) と呼ばれる、多角形を組み合わせた区分線形曲面を、一般的な形状表現として用いる。メッシュにおける、区分した 1 つの領域をループと呼ぶ。1 つのループは閉じた多角形で表し、ループ内部に稜線は無いものとする。本論文では、メッシュを構成する頂点、稜線、ループを要素と呼び、要素同士の繋がりに関する情報を位相と呼称する。そして、ポリゴンメッシュの断裂やメッシュを構成する要素の分割など、要素同士の繋がりが変化することを位相変化と呼ぶ。メッシュで表現したキャラクタなどのアニメーションを行う場合、要素の接続関係つまり位相を変えることなく、頂点の位置座標を変更することでメッシュの形状を変え、アニメーションを行うのが一般的である。スキンメッシュアニメーションと呼ぶこの手法では、キャラクタの向きや位置のみの変更によるアニメーション以外にも、人間の肘や膝などの関節部分のように、ひと繋がりになっている部分の曲げ伸ばしなどの動作アニメーションが表現可能であり、広く一般的な手法となっている。 図 1.1 は、メッシュによって表現したキャラクタである。しかし、この方法では表現が困難なアニメーション表現も存在する。例えば、液体のような分裂などが頻繁に発生する不定形形状の変形アニメーションや、人型の形状が魚型になるといった、形状の一部が別の形状に変化することによって、メッシュの位相が変化するようなアニメーションである。一般的なメッシュアニメーションはループの接続関係などの位相が動的に変化することを考慮していない。そのため、位相が変化する不定形形状などの分裂、融合が頻繁に発生するアニメーションや、形状が大幅に変わってしまうようなアニメーションには向いていない。

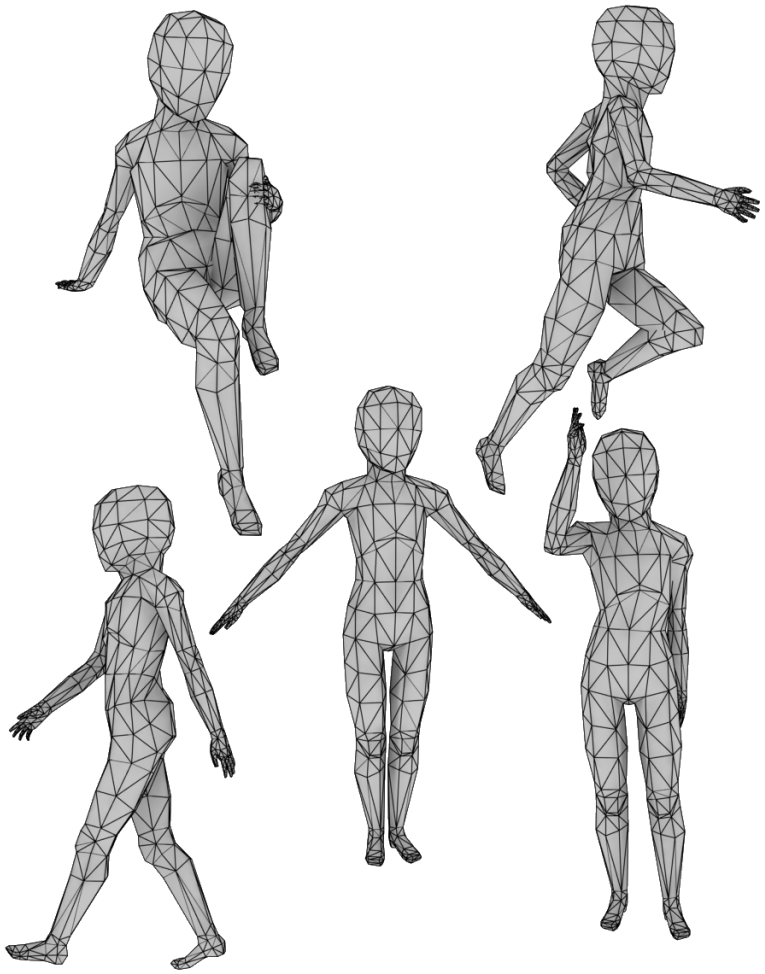


図 1.1: メッシュで表現したキャラクタ

本研究では、メッシュアニメーションが苦手とする表現の中でも、部分的、局所的に位相が変化する変形アニメーションに着目した。局所的な変形アニメーションには、もぐらなどが地中を移動して地面が盛り上がるような表現や、キャラクタなどの怒りを表現するために額に浮き出る血管などがある。また、ある形状の 1 部分から別の形状が現れるような表現なども局所的な変形アニメーションに含まれる。 図 1.2 は 3 次元形状において、局所的な形状変形を表した図である。これらの表現はもととなる形状全体、前述の例で言えば円形状などは大きく変化しないが、局所的な部分だけが大きく変化する。

局所的な形状変形手法としては、メッシュを滑らかに細分割する細分割曲面

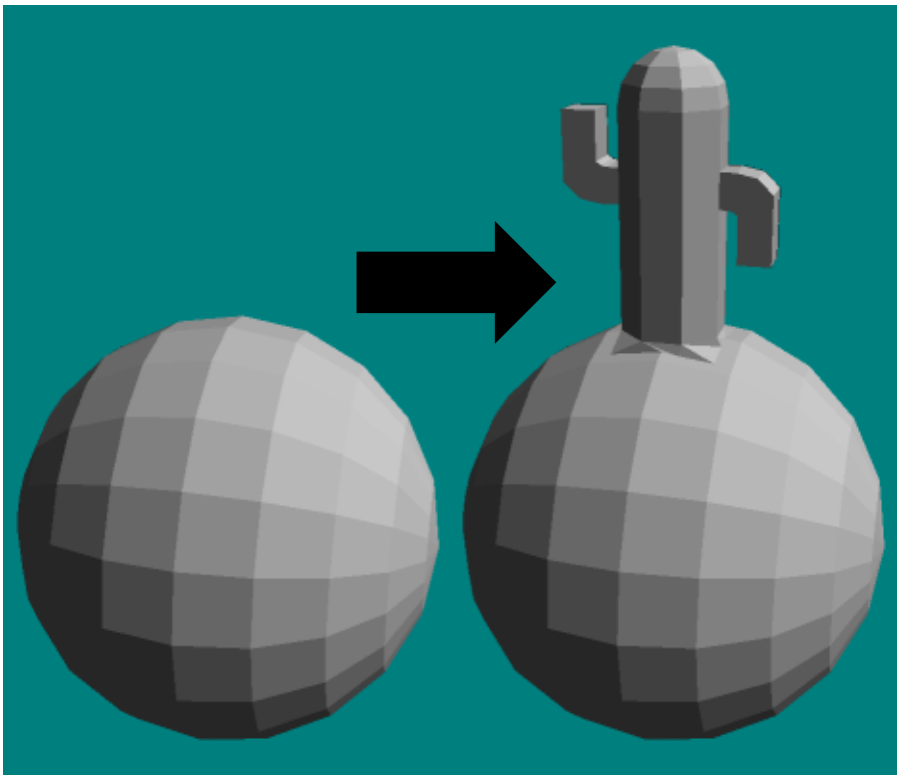


図 1.2: 3 次元形状における局所的な形状変形

[1][2][3] の局所変形を行う手法 [4][5][6][7][8] や、3 次元メッシュを別の 3 次元メッシュと合成する手法 [9][10][11][12][13] などがある。しかし、局所変形を行う手法は、元の形状を変形することを目的としているため、別に作成しておいた形状を付加するような表現は難しい。また、メッシュを合成する手法は 3 次元形状同士の合成のため、接続部の探索が難しく計算量が膨大になるため、リアルタイムにアニメーションを行うことは難しい。

メッシュアニメーションが可能な局所的な形状変形手法のひとつに変位マッピング [14] がある。変位マッピングとは、テクスチャマッピング手法の一種である。テクスチャマッピングとは、メッシュの 3 次元形状に対して 2 次元の画像データをマッピングし、物体の質感表現を向上する手法である。メッシュの各ループの頂点と 2 次元画像上の任意の点で対応をとることでループに画像を貼り付けるような表現を行う。コンピュータで扱う 2 次元画像には大きくラスター画像とベクター画像の 2 種類が存在するが、テクスチャマッピングではラスター画像を用いるの

が一般的である。ラスター画像とは、ピクセルと呼ばれる小さな点が集合して表す画像である。テクスチャマッピングでは通常、ラスター画像の各ピクセルにおける色情報を直接色として扱うが、物体の透明度や光の反射率など様々なデータを色としてラスター画像に書き込み、1つの画像をデータベースのように扱う手法[15]も存在する。

変位マッピングは各ピクセルにおける色情報をベクトル情報として扱う、画像をデータベースとして扱う手法の一種である[16]。通常のテクスチャマッピングはメッシュ形状における質感表現向上が目的であるが、変位マッピングではメッシュ頂点の座標値を直接変更して形状を直接変形し、複雑な凹凸表現を行うことを目的とする。変位マッピングにおけるテクスチャ画像を、特に変位マップと呼称する。本論文ではさらに、変位マップをマッピングするメッシュ形状を基礎メッシュと呼称することとする。変位マップは各ピクセルに方向と量を表すベクトル情報を保持する。このベクトル情報を変位ベクトルと呼称する。変位マップは通常、ある3次元のメッシュの頂点位置情報を元に作成する。本論文では、変位マップの元となる3次元メッシュ形状を付加メッシュと呼称する。変位ベクトルには、付加メッシュを構成する頂点の座標値が該当する。基礎メッシュと変位マップの対応をとり、変位マップの各ピクセルに対応した基礎メッシュの頂点位置を変位ベクトルに従い変更することで、複雑な形状変形を表現可能とする。変位マップのマッピング範囲を限定的なものとすることで、メッシュ形状の局所的な形状変形を表現する。複数の変位マップを作成し、変位マップを切り替えたりモーフィングすることにより、メッシュ形状におけるアニメーション表現を行うことが可能となる。図1.3は変位マップと変位マッピングによって形状変形を行ったメッシュである。

以上のように変位マッピングを用いた局所的形状変形アニメーション表現は有効なものであるが、複雑な凹凸表現にはいくつか解決しなければならない問題が存在する。

第1に、付加メッシュのデータが変位マップの解像度や色深度に依存するという問題である。変位マップはラスター画像で構成するが、このラスター画像の解像

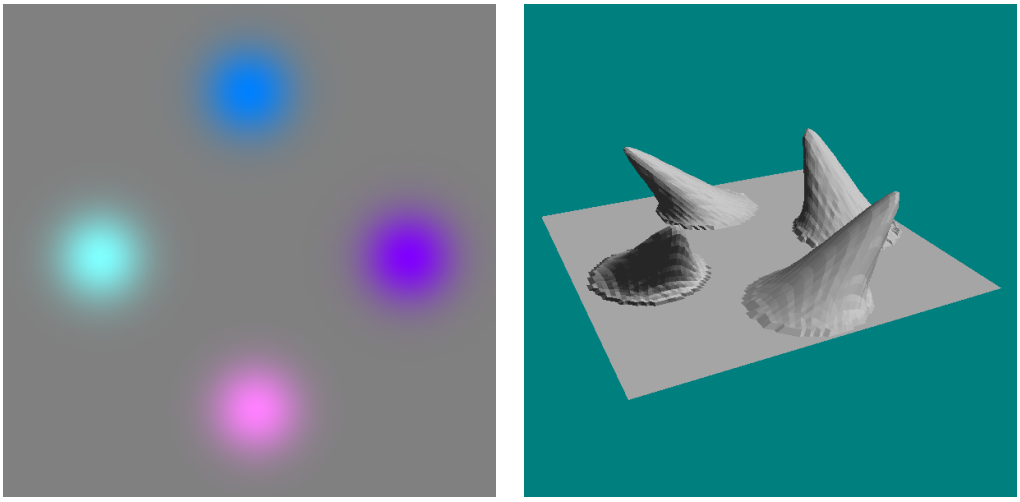
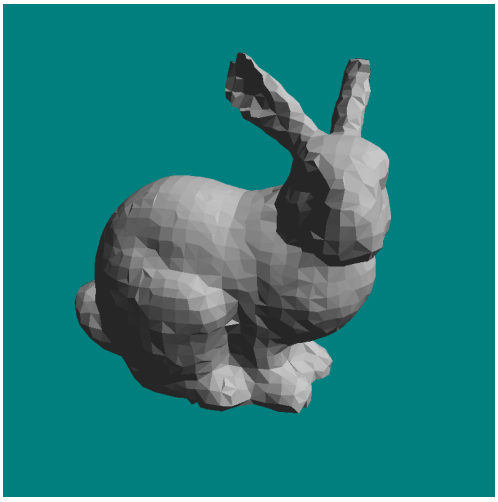
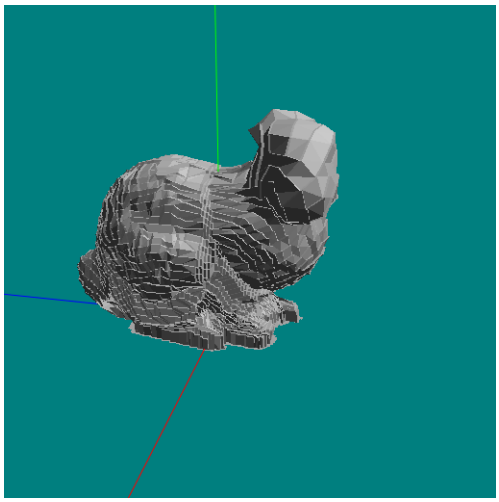


図 1.3: 変位マッピング

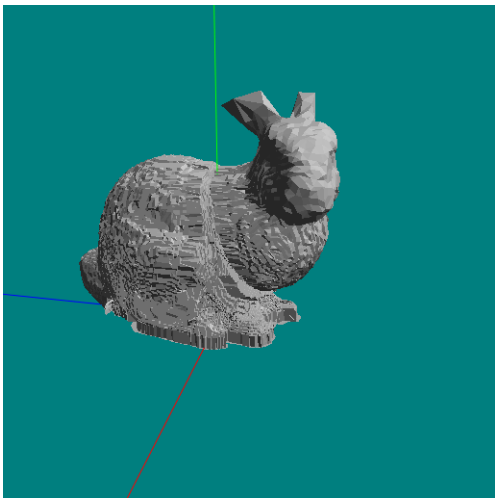
度に付加メッシュの 3 次元形状が依存する。付加メッシュが高精細なものであれば、変位マップの解像度も大きくする必要が生じる。付加メッシュがあまりに複雑な形状の場合、リアルタイム 3DCG で扱うには現実的でない大きさの変位マップになる可能性もある。また画像の色値として座標データを保存するため、色値以上の精度を保存できないという問題も生じる。 図 1.4 は同一の付加メッシュから解像度の違うラスター画像を生成し、変位マッピングを行った場合の比較である。 図 1.4(a) は付加メッシュとなる元形状である。 図 1.4(a) からそれぞれサイズの異なるラスター画像を生成し、それを変位マップとして、格子状にループが並んだ平面メッシュに変位マッピングを行った図である。 図 1.4(b) は、64×64 のサイズのラスター画像を生成した例である。 図 1.4(c) は、2048×2048 のサイズのラスター画像を生成した例である。 図 1.4(b) は、 図 1.4(c) に比べ耳の形状が落ちてしまっているのがわかる。また、 図 1.4(b) は、木目のような縞模様が目立ってしまっている。 図 1.4(c) も、耳の形状を完全に再現できているとはいえず、図 1.4(a) を完全に再現するためには、さらに大きなサイズのラスター画像を用意する必要がある。



(a) 元形状



(b) 64×64

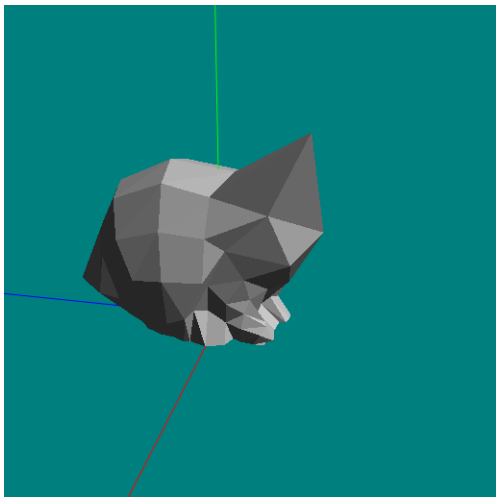


(c) 2048×2048

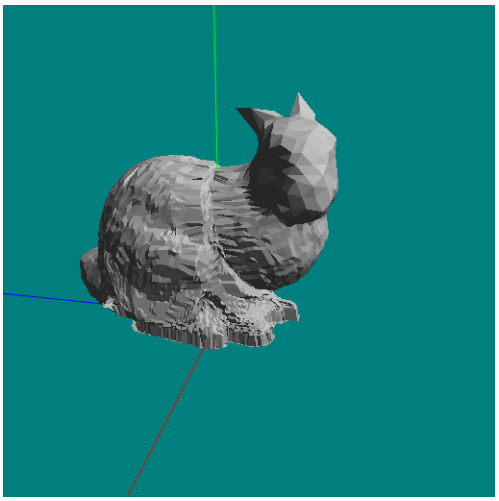
図 1.4: 変位マップ解像度依存

第 2 に、基礎メッシュの変形部分における頂点数の問題である。変位マップが高解像度になった場合、全ての変位ベクトルを活用するためには基礎メッシュの変形部分も、変位マップの解像度と同等の頂点数を必要とする。上記の問題点と同様にして、複雑な形状を表現する場合、リアルタイム 3DCG で扱うには現実的でない数の頂点数を必要とする場合がある。 図 1.5 は同一の変位マップを用いて、頂点数の違う基礎メッシュで変位マッピングを行った場合の比較である。付加メッシュには 図 1.4(a) を用いた。 図 1.5 で用いた基礎メッシュは格子状の平面メッシュで

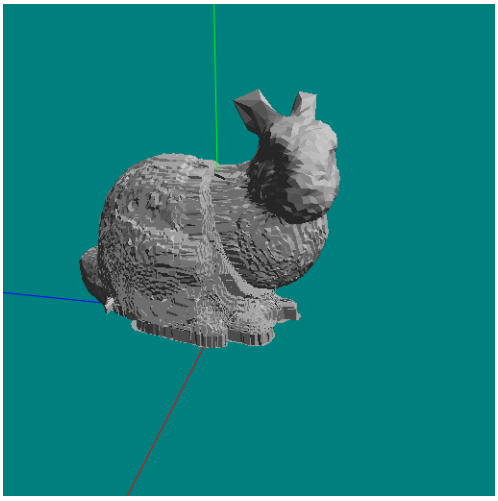
ある。格子の縦横の分割数は同じものとし、分割数はそれぞれ、 図 1.5(a) が 10 分割、 図 1.5(b) が 100 分割、 図 1.5(c) が 200 分割である。 図 1.5(a) は付加メッシュがなんであるか判別するのが難しく、 図 1.5(b) でも頭の部分など細かいディテールまでは出ていない。 図 1.5(c) も耳のような細かいディテールは出ていないが、他の部分に関してはいいように見える。しかし、 図 1.5(c) は縦横 200 分割なので、8 万もの 3 角形ループを用いているため、現実的なループ数とはいえない。



(a) 分割数 10



(b) 分割数 100



(c) 分割数 200

図 1.5: 基礎メッシュの頂点数依存

第 3 に、複数の変位マップの同時マッピングにおける問題である。1 つの基礎

メッシュに対して複数の変位マップをマッピングする際、マッピング部分が重複する箇所が存在する場合がある。変位マップは付加メッシュを元にした変化量データの集合であるため、重複部分はそれぞれの付加メッシュ形状から歪んだ状態になってしまう。

これらの問題を改善する手法として、変位ベクトルを活用する際に動的に基礎メッシュを細分割する手法 [17][18][19]、ボリュームデータを変位マップとして扱う研究 [20][21][22] などがある。しかし、動的に基礎メッシュを細分割する手法は、変位マップのマッピング範囲にかかわらず基礎メッシュ全体を一括で細分割するので、不要な頂点が増えてしまうことがありうる。また、ボリュームデータは計算速度が増大したり、ボリュームデータのアニメーションが難しいなど、別の問題も発生する。以上の問題点から、変位マッピングを用いた局所的形状変形アニメーション表現には改善の余地がある。

位相変化せずにこれらの変形アニメーションを実現する場合、変形後の形状を表現可能な位相を事前に用意しておく必要がある。変形形状が複雑であれば、それだけメッシュのループ数は大量になり、変形アニメーション処理も煩雑になる。どこが変形するか事前に分かっているのであれば、位相変化せずにアニメーションを行うことが可能だが、変形する場所が不定の場合は動的な位相変化が必要になるため、リアルタイムにアニメーションを行うことが難しい。

そこで本研究では、位相変化を簡略化することで、局所的に複雑な変形が発生して、変形箇所もリアルタイムに変化するアニメーションが表現可能な、局所的形状変形手法の実現を目的とする。本研究で目的とする局所的形状変形手法の確立は、変位マッピングにおける変位マップの解像度依存問題や基礎メッシュの頂点数依存問題が解消し、複数の変位マップの同時マッピングも可能になるなど、更なる演出効果が望める。

本手法では、3次元メッシュである付加メッシュを、ループや稜線が重ならないように2次元状になるように頂点座標を操作し、2次元メッシュを作成する。この2次元メッシュを変位マッピングにおける変位マップのように扱って基礎メッシュ

にマッピングする。最終的に基礎メッシュと付加メッシュを合成して局所的形状変形を行う。2次元メッシュの形状を正方形に限定することで、位相変化を伴う合成処理を簡略化した。 図 1.6 は本手法を用いて局所的形状変形を行った図である。

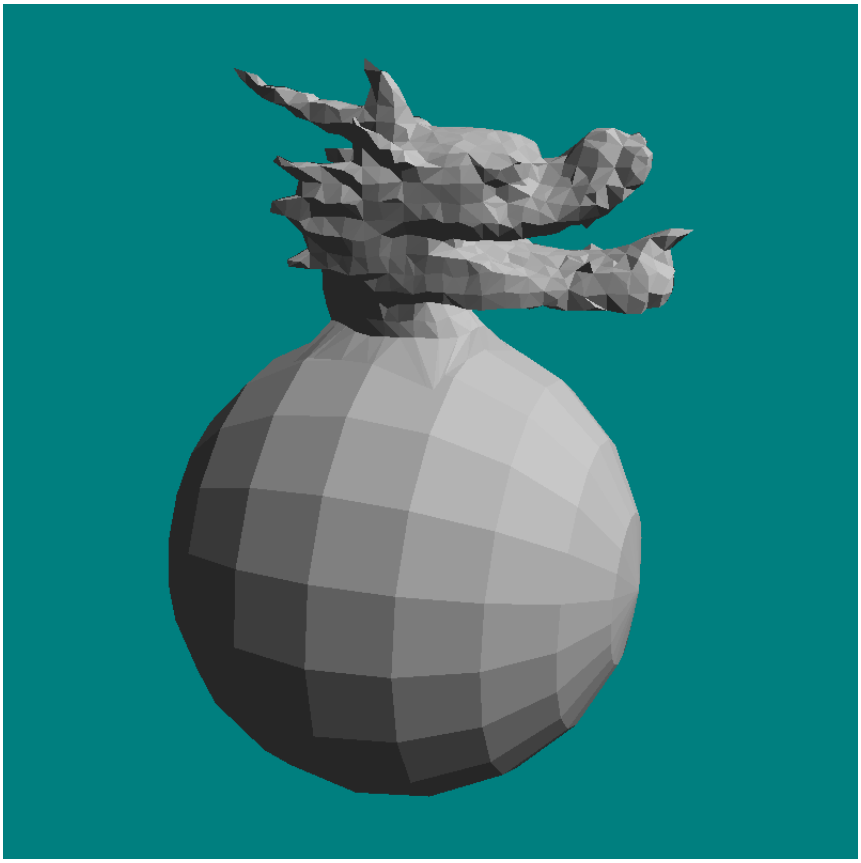


図 1.6: 提案手法

本手法では、2次元メッシュを変位マップとして扱うため、ラスター画像を生成する必要がない。そのため、既存の変位マッピングのようにラスター画像の解像度に依存するという問題がない。また、変位マップとする2次元メッシュを基礎メッシュと合成することで、基礎メッシュの不用意なループの増加を無くし、データを損失することなく3次元形状の復元が可能となった。さらに、変位マップ同士が重なり合った場合も、復元後の3次元形状を考慮した復元が可能である。提案手法をプログラムで実装し、検証を行い、提案手法の有用性を確認した。

1.2 論文構成

論文の構成は以下の通りである。 第 2 章では、本研究で提案する局所的形状変形手法について述べる。 第 3 章では、本研究で開発した局所的形状変形手法のプログラムにより、その結果の検証と考察を行う。 第 4 章では、本研究の成果と意義をまとめ、今後の展望について述べる。

第 2 章

提案手法

本章では、本研究で提案する局所的な変形手法の手順について述べる。提案手法では、変位マッピングにおける変位マップに2次元メッシュを用いる。本章における、基礎メッシュと付加メッシュの関係は、前章で述べた変位マッピングにおける基礎メッシュと付加メッシュの関係と同様である。

2.1 メッシュ表現

本論文におけるメッシュの数式表現は以下のように定める [23]。本手法では、メッシュを構成するループはすべて3角形であるものとし、図中で4角形で表現しているループも3角形の集合である。

- メッシュは (P, K) のペアで表現し、文中では $\mathcal{P}$ と表す。ここで、P は N 個からなる頂点座標で、次の 式 (2.1) で表し、K は位相情報である複体を表す。

$$\mathbf{p}_i = [(x_i, y_i, z_i) \in \mathbf{R}^3 (1 \leqq i \leqq N)] \tag{2.1}$$

- 座標値 $\mathbf{p}_i$ をもつ頂点位相を $\{i\}$ と表す。

- 複体 K は頂点集合 $v = \{i\} \in K$、稜線集合 $e = \{i, j\} \in K$、ループ集合 $f = \{i, j, k\} \in K$ の3種類の要素から成り立つ。ただし、$\{i, j\}$ は $\{i\}$ と $\{j\}$ を両端点に持ち、ループ $\{i, j, k\}$ は $\{i\}$、$\{j\}$、$\{k\}$ を頂点に持つ。

- $\{i, j\} \in K$、すなわち稜線 $\{i, j\}$ が複体 K 中に存在するとき、$\{i\}$ と $\{j\}$ は隣接するという。

- $N_v(i)$ を次のように定める。

$$N_v(i) = \{j | \{i, j\} \in K\} \tag{2.2}$$

すなわち、$N_v(i)$ は頂点 $\{i\}$ と隣接する頂点の集合である。 図 2.1 は頂点 $\{i\}$ と $N_v(i)$ の関係を示した図である。図中の黒丸の集合が $N_v(i)$ である。

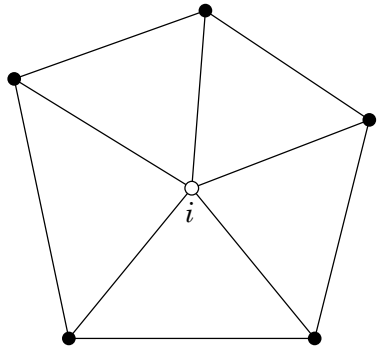


図 2.1: $N_v(i)$

- $e = i, j$ において、$N_e(e)$ を次のように定める。

$$N_e(e) = \{k | \{i, j, k\} \in K\} \tag{2.3}$$

すなわち、$N_e(e)$ は稜線 e を含む面の集合を構成する頂点のうち、$\{i\}$ と $\{j\}$ 以外のものを指す。 図 2.2 は稜線 e と $N_e(e)$ の関係を示した図である。図中

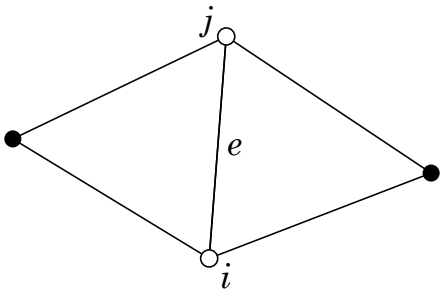


図 2.2: $N_e(e)$

の黒丸の集合が $N_e(e)$ である。

2.2 手法の流れ

本手法は通常の変位マッピングと同様の流れでメッシュの局所変形を行う。ただし、本手法では付加メッシュはラスター画像ではなく、2 次元メッシュに変換し、これを変位マップとする。メッシュ合成手法において、3 次元メッシュから 2 次元パラメータ座標系への写像を求めて合成する方法 [9] は一般的に行われている。しかし、変位マップに 2 次元メッシュを用いる手法はまだない。入力は付加メッシュ

と基礎メッシュの 2 つのメッシュとする。まず、付加メッシュから 2 次元メッシュを求め、これをパラメータ化 [24][25][26] する。次に、基礎メッシュを 2 次元空間上に展開し、このメッシュ上に正方領域を規定し、正方領域内に収まるようにメッシュを求める。次に基礎メッシュの形状に沿うようにメッシュを求める。その後、変位マップの変位を行う。最後に基礎メッシュと変位マップを合成して、本手法によるメッシュの局所変形は終了である。以降の節で各工程について細かく述べる。図 2.3 は本手法の流れを例示したものである。

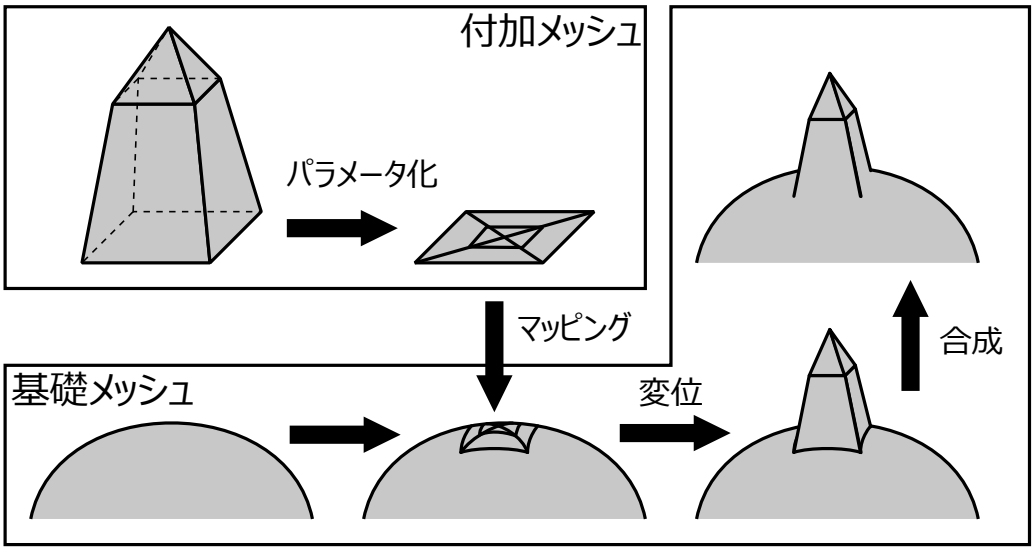


図 2.3: 本手法の流れ

2.3 付加メッシュのパラメータ化

本節では、付加メッシュのパラメータ化を行う方法について述べる。パラメータ化は付加メッシュから 2 次元メッシュを求めることで行う。本手法ではマッピングを容易にするために、2 次元メッシュは正方形となるようにする。付加メッシュとして、本手法で想定している形状は以下の 3 点を満たす形状とする。

1. 任意の稜線 e に対し、$N_e(e)$ の個数が 2 個以下であること。

2. 頂点は必ずループを構成している頂点であること。

3. 基礎メッシュと接続する場所でのみ開いた形状であること。

3番目の条件により、付加メッシュは開いた形状となるため、境界稜線が存在する。この境界稜線を構成する頂点群が付加メッシュをマッピングするときの接続部となる。ここでは接続部となる頂点を、境界頂点と呼ぶこととする。3次元メッシュである付加メッシュから2次元メッシュを求める工程は、境界頂点群とそれ以外の頂点群という2段階に分けて行う。まず、境界頂点群を元に平面を、直交する2つのベクトルと、任意の点によって定義する。この2つのベクトルの外積ベクトルと合わせた3つの直交ベクトルを座標軸とし、任意の点を原点とするローカル座標系 Λ を規定する。Λ において次の 式 (2.4) を満たすような領域 L を規定する。

$$L = \{\mathbf{l} | -1 \leqq l_x \leqq 1, l_y = 0, -1 \leqq l_z \leqq 1\} \tag{2.4}$$

このとき、領域 L をパラメータ化する際の正方領域とする。付加メッシュをこの領域 L 内に収まるように、2次元メッシュを生成する。

付加メッシュをメッシュ$\mathcal{A}$とし、求める2次元メッシュを$\mathcal{B}$とする。メッシュ$\mathcal{B}$の初期状態は、メッシュ$\mathcal{A}$と同様の位相と頂点座標を持つものとする。また、メッシュ$\mathcal{B}$が持つ頂点集合を $B = \{B_1, B_2, B_3, \dots\}$ とする。まず、正方領域 L の境界に対し、B の境界頂点を操作する。 図2.4(a) はメッシュ$\mathcal{B}$と正方領域 L の位置関係を示した図である。また、図中の黒丸が境界頂点、白丸がその他の頂点を表している。境界頂点同士の隣接関係が崩れないように、境界頂点を均等に配置する。このとき、境界稜線が重なる場合は適宜頂点の位置を変更して、稜線が重ならないようにする。さらに、2次元メッシュが正方形となるように正方領域の4つの角に必ず頂点を配置する。 図2.4(b) は正方領域の境界に境界頂点を移動したメッシュ$\mathcal{B}$を表している。次に、境界頂点以外の B_i が正方領域に収まるように変換する。B における $N_v(i)$ からなる多角形の重心を求め、B_i を重心位置に移動する。境界頂点以外のすべての B_i に対して、B が正方領域 L 内に収まるように、上記の処理を繰り返す。境界頂点を平面上に配置しているため、上記の処理を繰り返すことで、すべての頂点が正方領域内の平面上に収束することになる。 図2.4(c) は正方

領域に収束したメッシュ$\mathcal{B}$である。メッシュ$\mathcal{B}$を求めたら、Bをパラメータ化し、

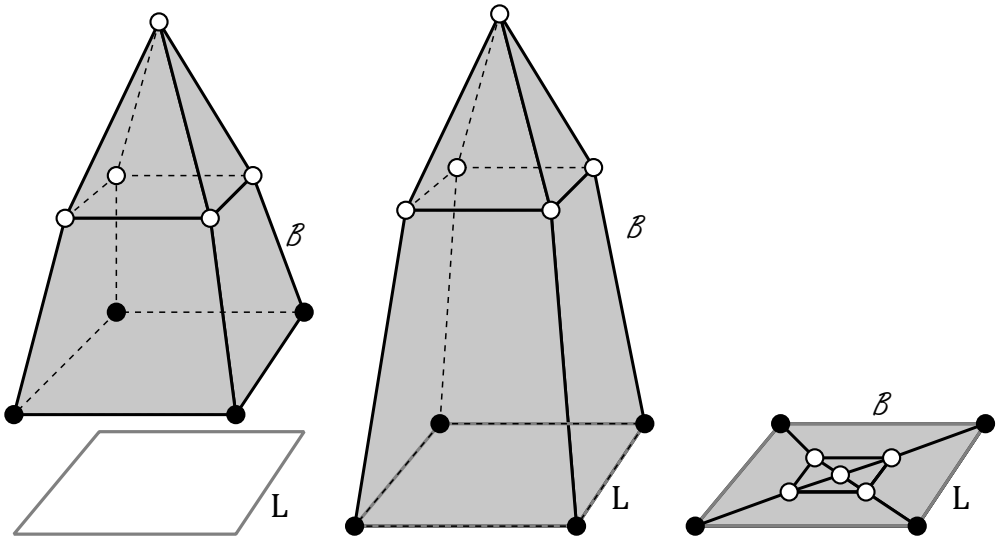


図2.4: 付加メッシュのパラメータ化

パラメータ化した頂点集合を$Q = \{\mathbf{q}_i\}$とする。正方領域Lを 式(2.4) と定めたので、パラメータは以下の 式(2.5) によって求める。

$$\mathbf{q}_i = \frac{\mathbf{b}_i}{2} + \left(\frac{1}{2}, \frac{1}{2}\right) \tag{2.5}$$

図2.5 は、図2.4 で作成した2次元メッシュ$\mathcal{B}$と座標軸の関係を示した図である。

2.4 付加メッシュの配置

本節では、付加メッシュの基礎メッシュへの配置について述べる。基礎メッシュを$\mathcal{M}$とし、2次元化した付加メッシュ$\mathcal{B}$の配置を決めるため、メッシュ$\mathcal{M}$から2次元メッシュ$\mathcal{W}$を求める。メッシュ$\mathcal{M}$の頂点集合を$M = \{M_1, M_2, M_3, \dots\}$としたとき頂点$M_i$が持つ位置ベクトルを$\mathbf{m}_i$とし、$M_i$に2次元座標を持つ頂点$W_i$を対応付ける。$W_i$によって構成されるメッシュを$\mathcal{W}$とする。メッシュ$\mathcal{W}$の位相はメッシュ$\mathcal{M}$と同様である。このとき、メッシュ$\mathcal{W}$の稜線やループが重なり合わないように2次元座標$\mathbf{w}_i$を規定する。メッシュ$\mathcal{M}$のような3次元メッシュに対し、メッ

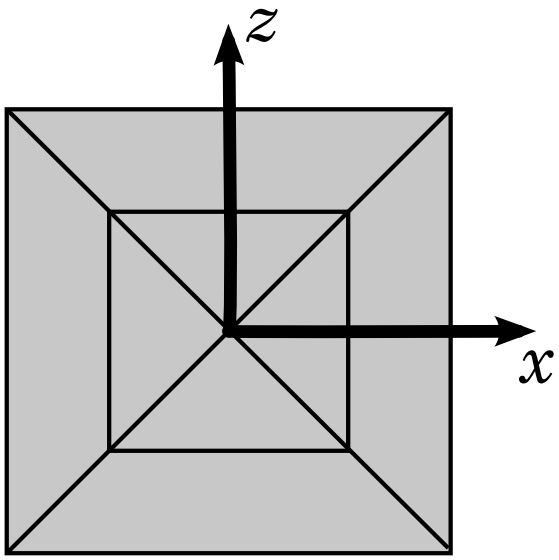


図 2.5: 2 次元座標系

シュ$\mathcal{W}$のような 2 次元メッシュを一意に求めるには、ABF[27][28] や LSCM[29] などの多くの手法 [30][31] を用いることで可能である。このメッシュ$\mathcal{W}$ 上で正方領域を決定する。メッシュ$\mathcal{W}$ 上で、正方領域の 4 隅のうちの 3 点となる点 P_1, P_2, P_3 を規定し、P_1, P_2, P_3 を用いて正方領域を定める。このとき、$\overrightarrow{P_1P_2}$ と $\overrightarrow{P_1P_3}$ が直交し、正方領域がメッシュ$\mathcal{W}$ 内に収まるように P_1, P_2, P_3 を規定する。 図 2.6 は、決

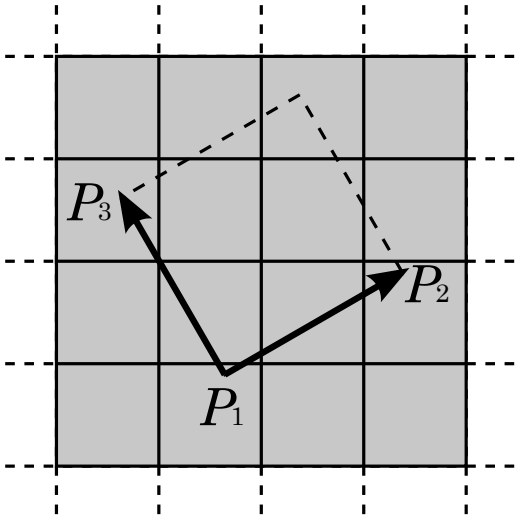


図 2.6: 正方領域の角 3 点

定した正方領域と、3 点 P_1, P_2, P_3 の位置関係を示す図である。 2.3 節で求めた Q

と、3 点 P_1, P_2, P_3 を用いて次の 式 (2.6) によってメッシュ$\mathcal{W}$ に配置するメッシュ$\mathcal{C}$ を求める。メッシュ$\mathcal{C}$ の頂点集合を $C = \{C_1, C_2, C_3, \dots\}$ としたとき頂点 C_i が持つ位置ベクトルを $\mathbf{c}_i$ とし、メッシュ$\mathcal{C}$ の位相はメッシュ$\mathcal{A}$ と同様である。

$$\mathbf{c}_i = \mathbf{p}_1 + q_{ix}(\mathbf{p}_2 - \mathbf{p}_1) + q_{iy}(\mathbf{p}_3 - \mathbf{p}_1) \tag{2.6}$$

基礎メッシュ$\mathcal{M}$ とメッシュ$\mathcal{W}$、メッシュ$\mathcal{C}$ から、メッシュ$\mathcal{M}$ 上にあり、メッシュ$\mathcal{M}$ の形状に沿うようなメッシュ$\mathcal{D}$ を求める。メッシュ$\mathcal{D}$ の頂点集合を $D = \{D_1, D_2, D_3, \dots\}$ としたとき頂点 D_i が持つ位置ベクトルを $\mathbf{d}_i$ である。メッシュ$\mathcal{W}$ において、$\mathbf{c}_i$ を内包するループ f を探す。f を構成する 3 頂点を $W_\alpha, W_\beta, W_\gamma$ としたとき、C_i との関係は次の 式 (2.7) によって成り立つ。

$$\mathbf{c}_i = \mathbf{w}_\alpha + s(\mathbf{w}_\beta - \mathbf{w}_\alpha) + t(\mathbf{w}_\gamma - \mathbf{w}_\alpha) \qquad (0 \leqq s, t \leqq 1) \tag{2.7}$$

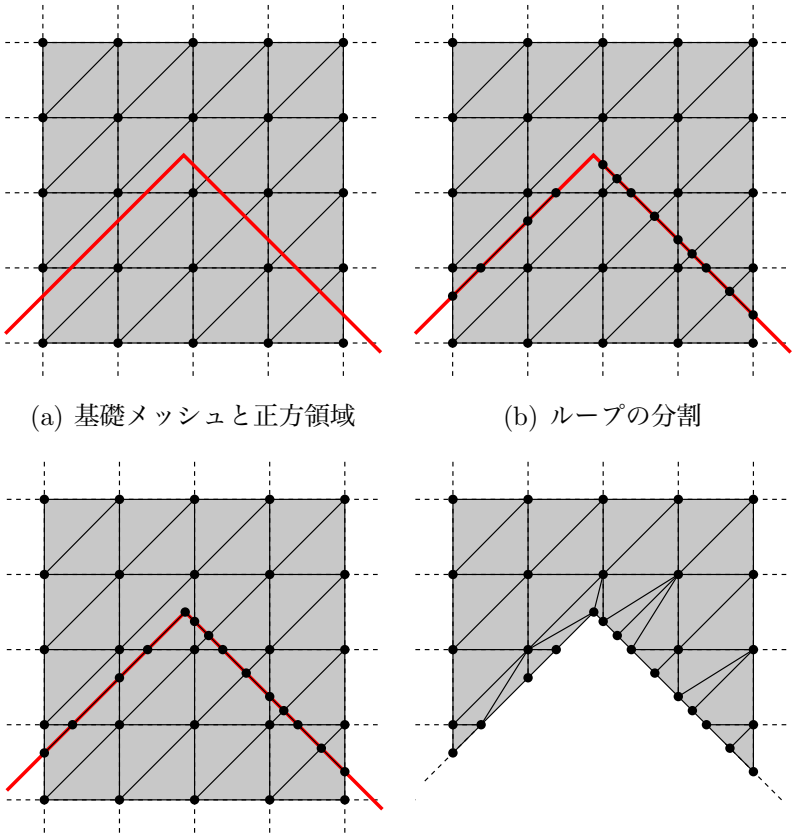
式 (2.7) から実数 s, t を求め、$W_\alpha, W_\beta, W_\gamma$ に対応するメッシュ$\mathcal{M}$ 上の頂点 $M_\alpha, M_\beta, M_\gamma$ から D_i を次の 式 (2.8) によって求める。

$$\mathbf{d}_i = \mathbf{m}_\alpha + s(\mathbf{m}_\beta - \mathbf{m}_\alpha) + t(\mathbf{m}_\gamma - \mathbf{m}_\alpha) \qquad (0 \leqq s, t \leqq 1) \tag{2.8}$$

これで、メッシュ$\mathcal{D}$ は基礎メッシュ$\mathcal{M}$ に沿うような形状となる。

2.5 基礎メッシュのトリミング

本節では基礎メッシュのトリミングについて述べる。基礎メッシュと付加メッシュを合成するためにメッシュ$\mathcal{W}$ に対しトリミングを行う。メッシュ$\mathcal{W}$ の稜線と正方領域の境界稜線との交点に新たな頂点を生成する。正方領域の角を内部に含まないループは正方領域の境界に沿って稜線を追加し、ループを分割する。正方領域の角を含むループは正方領域の角の位置に新たな頂点を追加し、正方領域の稜線に沿うようにループを分割する。正方領域の内部に位置するループを削除し、非三角形のループに三角形分割を行ってトリミングは終了である。トリミングを行ったメッシュを $\mathcal{W}$ とする。 図 2.7 は基礎メッシュのトリミングの様子を表し



(a) 基礎メッシュと正方領域　(b) ループの分割

(c) 角を含むループの分割　(d) 内部のループを削除

図 2.7: 基礎メッシュのトリミング

た図である。メッシュ$\mathcal{W}$をもとに、対応するメッシュ$\mathcal{M}$もトリミングし、これをメッシュ$\mathcal{T}$とする。

2.6　変位マップの変位

本節では、本手法における変位マップの変位について述べる。復元する割合をパラメータとしてメッシュ$\mathcal{A}$とメッシュ$\mathcal{D}$の頂点座標を補間することで、変位マップの形状が徐々に生えてくるようなアニメーションが可能となる。復元するためのパラメータをsとし、補間した中間頂点をH_iとしたとき、次の 式 (2.9) によって中間頂点を求める。

$$\mathbf{h}_i = (1-s)\mathbf{d}_i + s\mathbf{a}_i \qquad (0 \leqq s \leqq 1) \tag{2.9}$$

図 2.8 は、H_i によって構成されるメッシュ$\mathcal{H}$ のアニメーションの例である。メッ

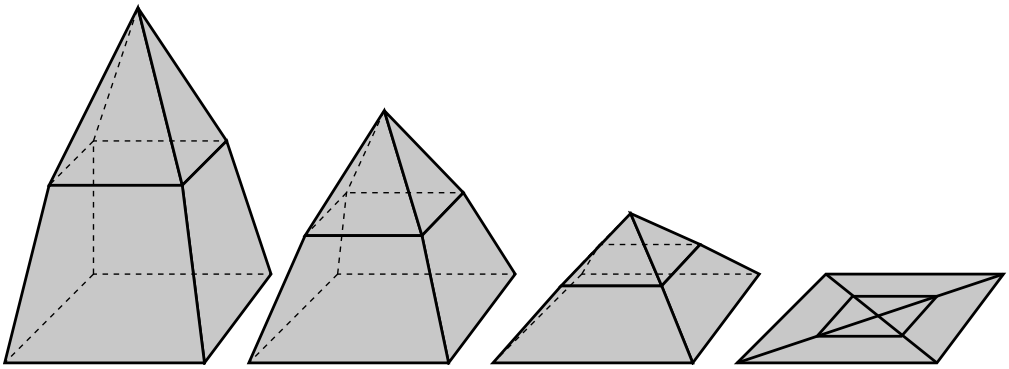


図 2.8: 付加メッシュのアニメーション

シュ$\mathcal{A}$ の頂点集合を複数もつ事によって、付加メッシュのモーフィングアニメーションも可能である。複数の頂点集合を $A_1 = \{\mathbf{a}_{1,i}\}, A_2 = \{\mathbf{a}_{2,i}\}$ とし、モーフィングするためのパラメータを t としたとき、中間頂点 H_i は次の 式 (2.10) によって求める。

$$\mathbf{h}_i = (1-s)\mathbf{d}_i + s((1-t)\mathbf{a}_{1,i} + t\mathbf{a}_{2,i}) \qquad (0 \leqq s, t \leqq 1) \tag{2.10}$$

図 2.9 は、$s = 1$ のときの、H_i によって構成されるメッシュ$\mathcal{H}$ のモーフィング

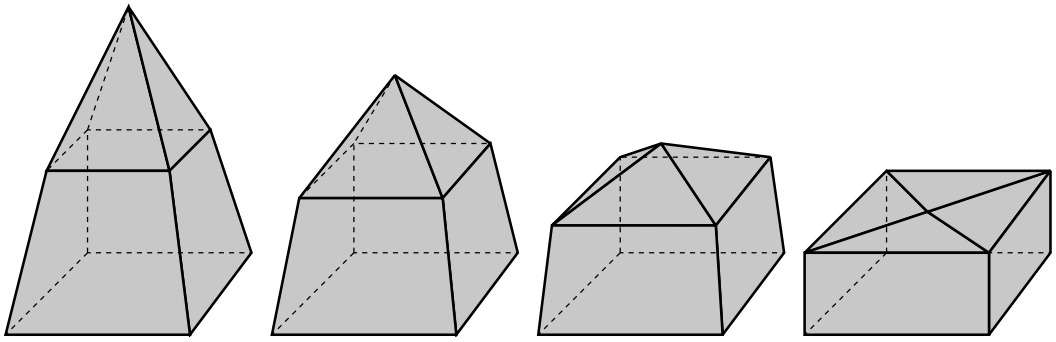
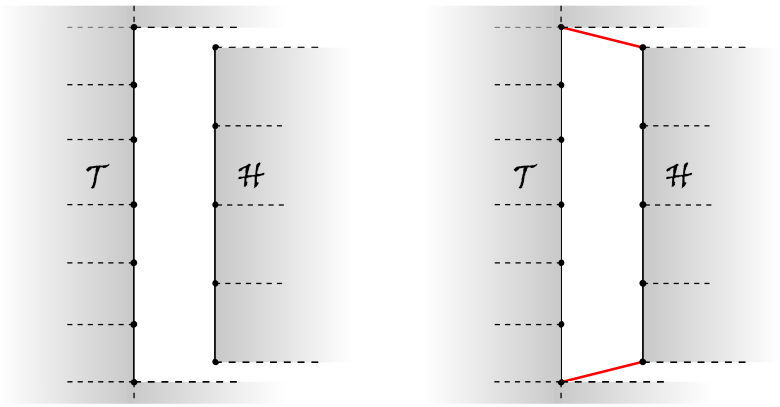


図 2.9: 付加メッシュのモーフィングアニメーション

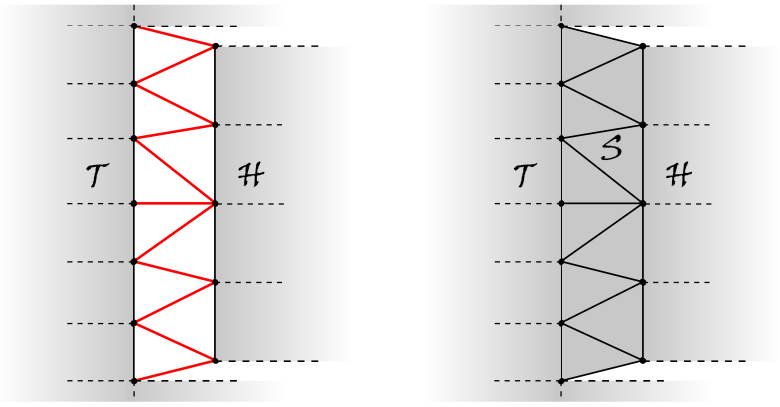
アニメーションの例である。

2.7　基礎メッシュと付加メッシュの合成

本節では基礎メッシュと付加メッシュの合成について述べる。トリミングした基礎メッシュ$\mathcal{T}$と変位したメッシュ$\mathcal{H}$を合成する。メッシュ$\mathcal{T}$とメッシュ$\mathcal{H}$がひとつのメッシュに見えるように、間を接続するようなメッシュ$\mathcal{S}$を生成する。メッシュ$\mathcal{S}$は、メッシュ$\mathcal{M}$からメッシュ$\mathcal{T}$を生成する際にトリミングを行ってできた境界稜線と、メッシュ$\mathcal{H}$の境界稜線を繋ぐように生成する。生成は正方領域の1辺ごとに行う。1辺の端点に位置する頂点を結んで稜線を生成。あとはループが3角形になるように稜線を生成し、メッシュを生成する。この処理を4辺全てに行いメッシュ$\mathcal{S}$を生成する。 図2.7 はメッシュ$\mathcal{S}$生成の様子を表した図である。



(a) 基礎メッシュと付加メッシュ　　(b) 端点に稜線を生成



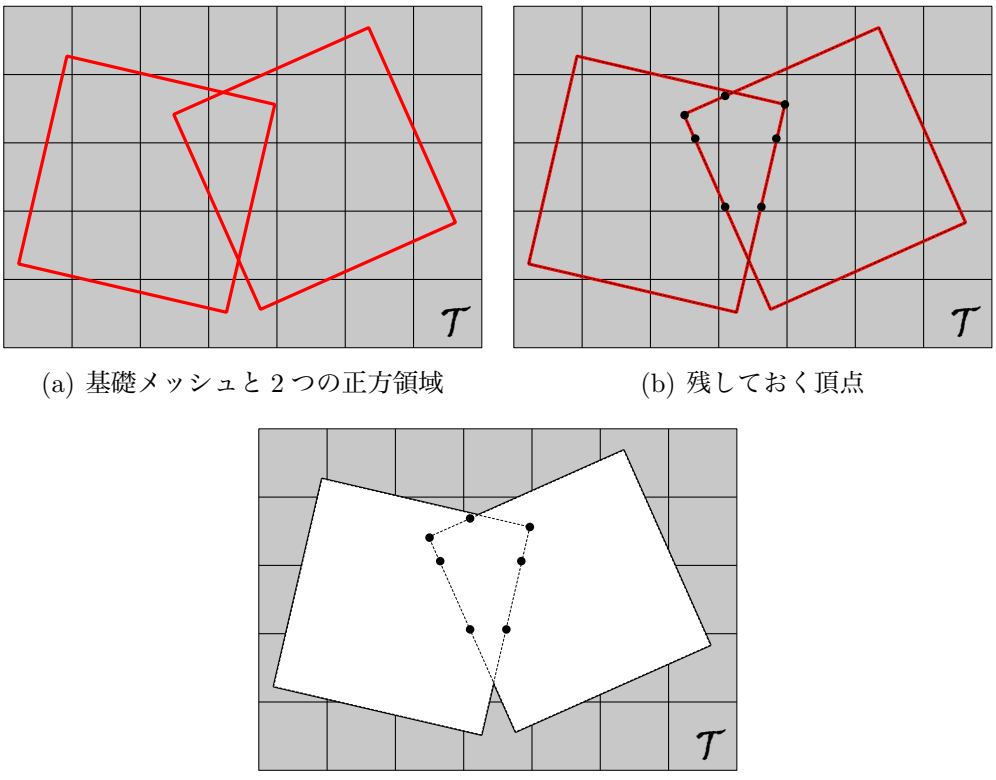
(c) その他の稜線を生成　　(d) ループを生成

図2.10: 接続メッシュの生成

最後に 3 つのメッシュ$\mathcal{T}, \mathcal{H}, \mathcal{S}$ を合成して、新たな合成メッシュ$\mathcal{U}$ とする。これで、本手法によるメッシュの局所変形は終了である。基礎メッシュ$\mathcal{M}$ を局所変形したメッシュが$\mathcal{U}$ となる。

2.8 複数の付加メッシュを用いたマッピング

複数の付加メッシュを用いる場合も、各付加メッシュを配置する際に用いた正方領域で基礎メッシュをトリミングし、付加メッシュをそれぞれ正方領域をもとに配置する。本手法では、処理を簡略化するために正方領域の境界に位置する頂点は残しておく。残しておいた頂点を用いることで、基礎メッシュと付加メッシュを接続するメッシュを生成できる。 図 2.11 は、正方領域が重なっている状況でのトリミングの様子を表した図である。付加メッシュの配置も正方領域が重なって



(a) 基礎メッシュと 2 つの正方領域　(b) 残しておく頂点

(c) トリミングした基礎メッシュ

図 2.11: 同時マッピングにおけるトリミング

いる場合でも無関係に付加メッシュを配置する。個々の付加メッシュにおいて独

立して処理を行うことで、互いに干渉することなく復元が可能となる。付加メッシュが復元する前の状態では、付加メッシュのループ同士が重なる事になるが、形状に影響をあたえることはないので、本手法では考慮しない。また、付加メッシュに付加メッシュをマッピングするような表現はできない。

第 3 章

評価と検証

本章では、本研究で提案した局所変形手法に沿って実装したプログラムを使用し、その有用性を検証する。本研究で試作したプログラムは、グラフィクス API の OpenGL[32] をベースとした 3 次元グラフィクスツールキットである「FK Kernel Tool Kit System」[33][34] を用いて実装した。本章では円柱面状の基礎メッシュを用い、様々な付加メッシュをマッピングした。検証に用いた環境は次の 表 3.1 のとおりである。

表 3.1: 検証に用いた環境

CPU	:	AMD Phenom(tm) II X4 945 Processor 3.00 GHz
RAM	:	4.00
GPU	:	NVIDIA GeForce GTX 260

3.1 2 次元メッシュ化

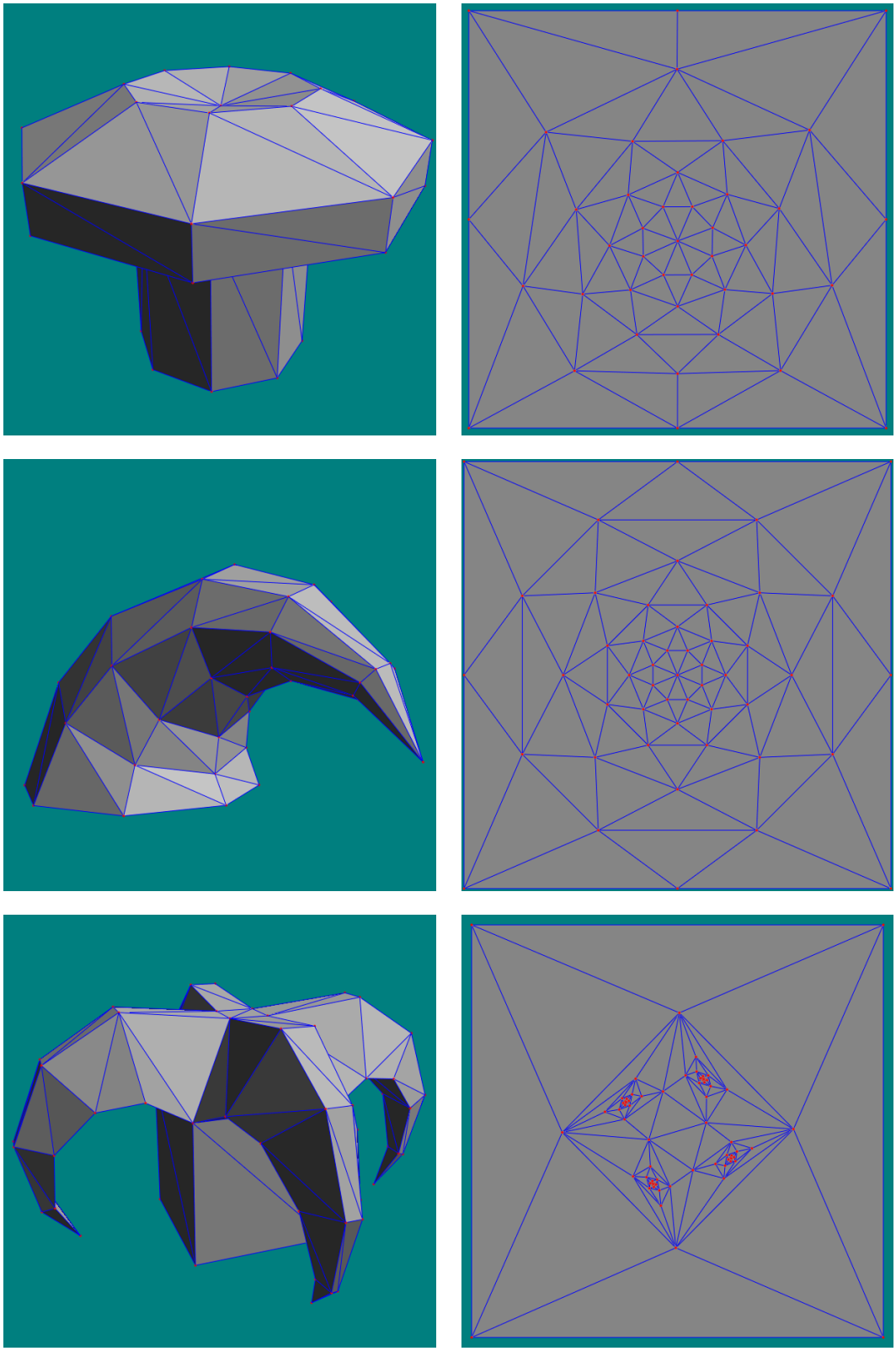
まず、複数の付加メッシュを 2 次元メッシュに変換する。付加メッシュは接続する位置で開いた形状になるように加工してある。各付加メッシュのループ数と 2 次元メッシュへの変換にかかった時間は 表 3.2 に示す。

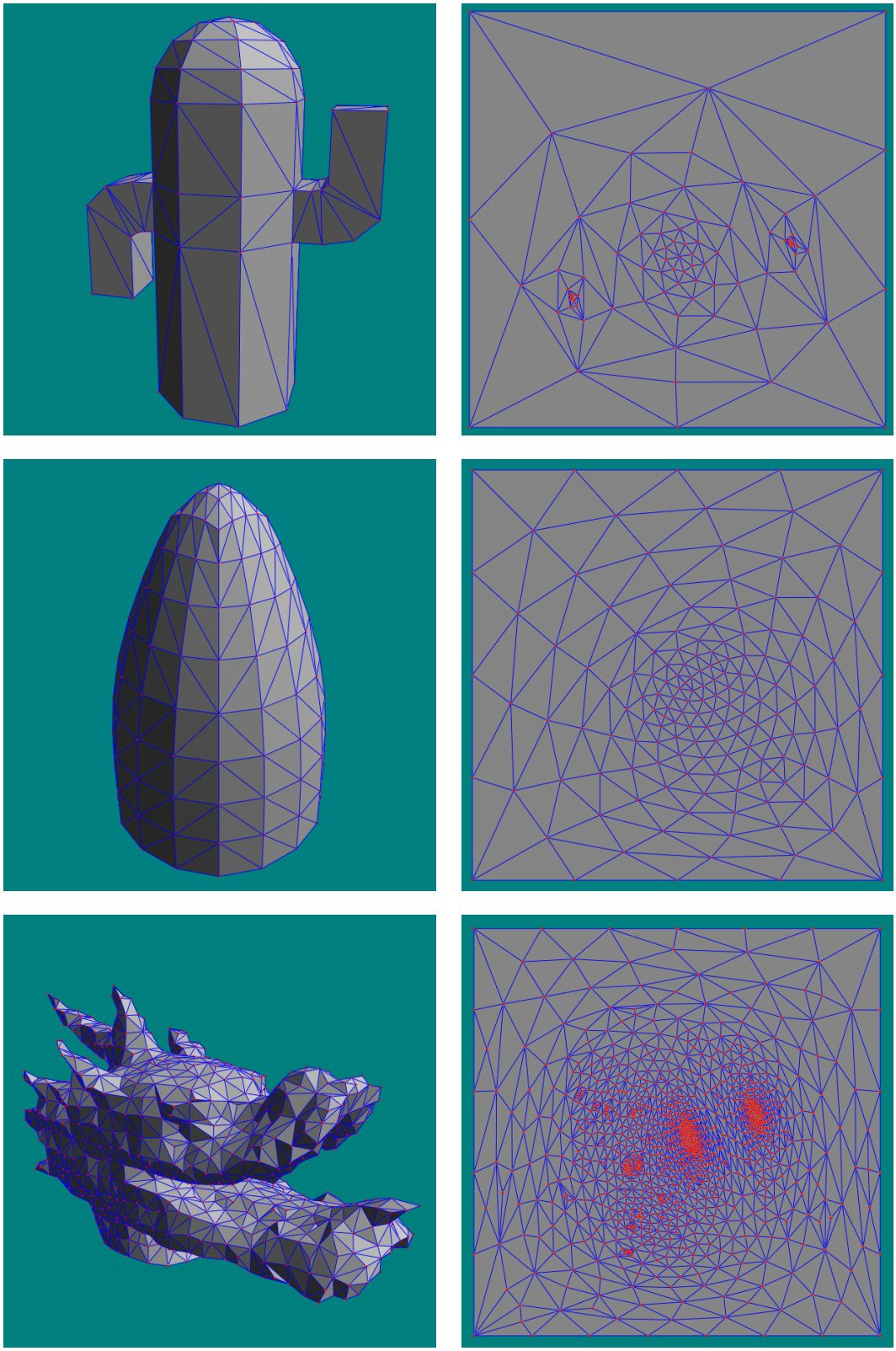
表 3.2: 2 次元メッシュへの変換にかかった時間

名前	ループ数	時間 (秒)
きのこ	72	0.296
つの	88	0.370
王冠	154	0.421
はにわ	168	0.670
凸	288	2.193
龍頭	2343	75.629
うさぎ	3388	212.145
龍	10089	1,331.026

ループ数が増えるに連れて処理にかかる時間も増えているのがわかる。ループ数が数千を超えると変換に数分かかるようになるが、2 次元メッシュへの変換は事前に行っておけばよいので、本手法におけるリアルタイム性には影響しない。付加メッシュの形状と、変換した 2 次元メッシュの形状を次の 図 3.1 に示す。 図 3.1

は左が付加メッシュ、右が 2 次元メッシュを示している。図の掲載順は 表 3.2 と同様である。





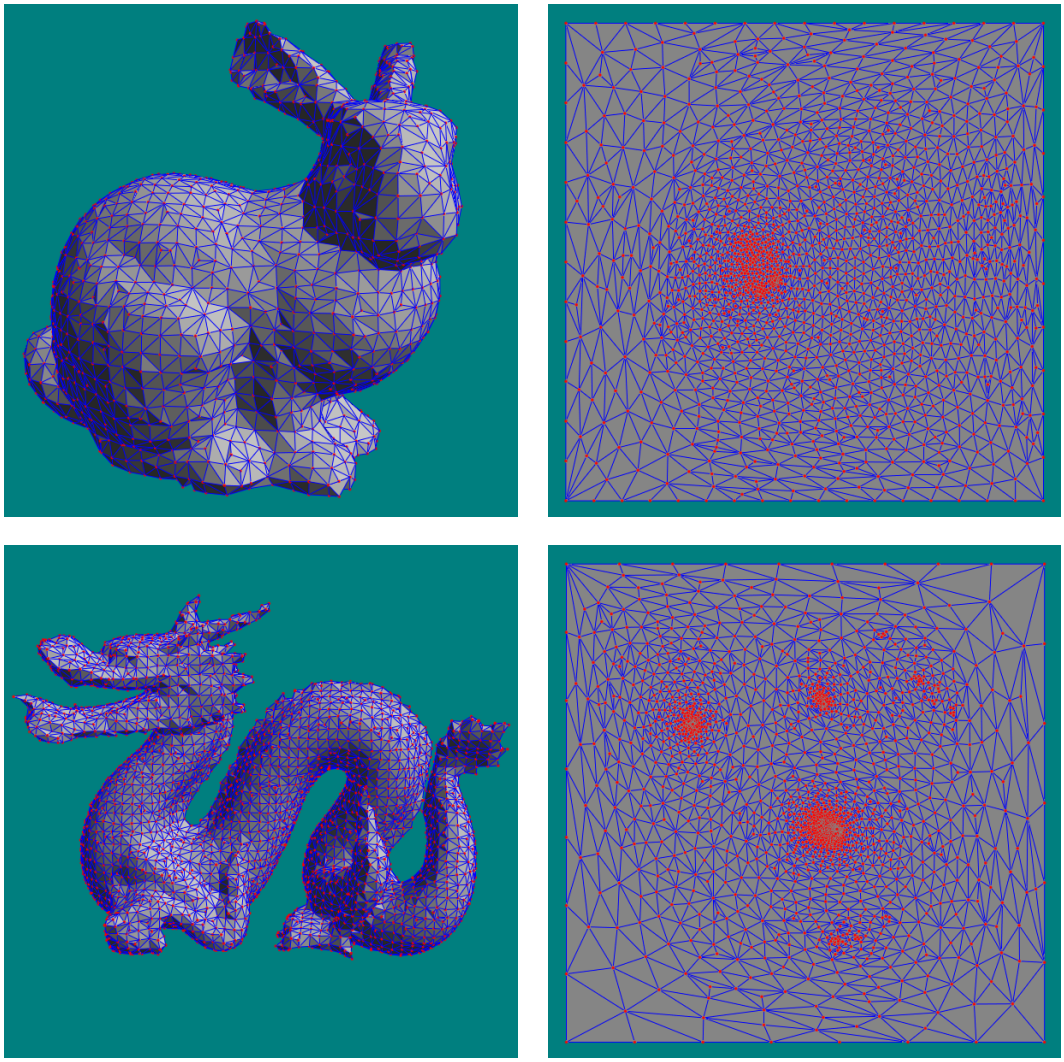
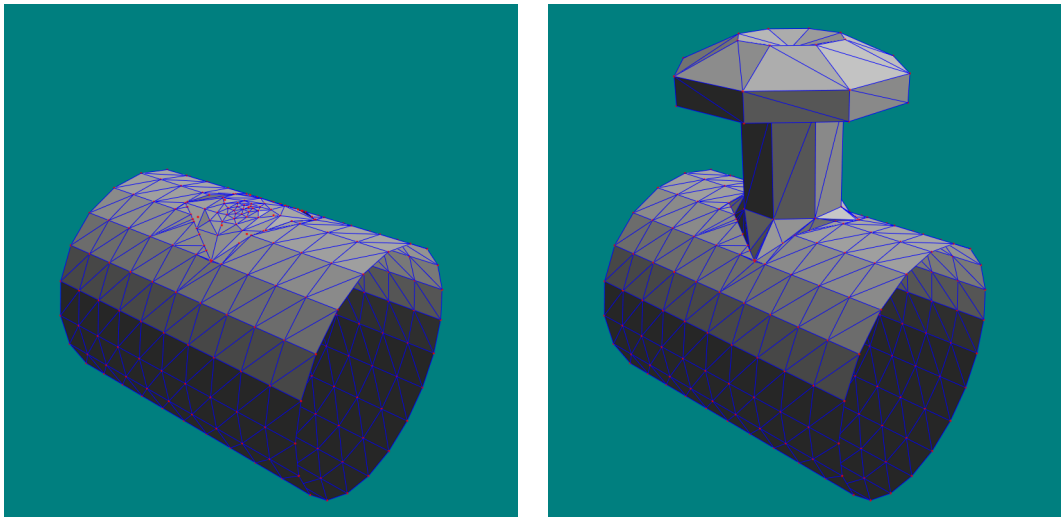


図 3.1: 付加メッシュの 2 次元メッシュ化

3.2　本手法の実行結果

本提案手法の実行結果を以下に示す。 図 3.2 は、きのこの形をした付加メッシュを用いて結果である。 図 3.2(a) は、基礎メッシュに付加メッシュをマッピングした合成メッシュである。 図 3.2(b) は、 図 3.2(a) から付加メッシュの形状を復元したメッシュを表す。



(a) (b)

図 3.2: 実行結果：きのこ

きのこ以外の様々な形状を用いた結果も次に示す。 図 3.3 はドラゴンの頭を用いた結果、 図 3.4 は龍の全身、 図 3.5 はうさぎを用いた結果である。各々、左が復元前のメッシュ、右が復元後のメッシュである。 2.3 節で示した条件さえ満たした形状であれば、付加メッシュとして利用可能である。

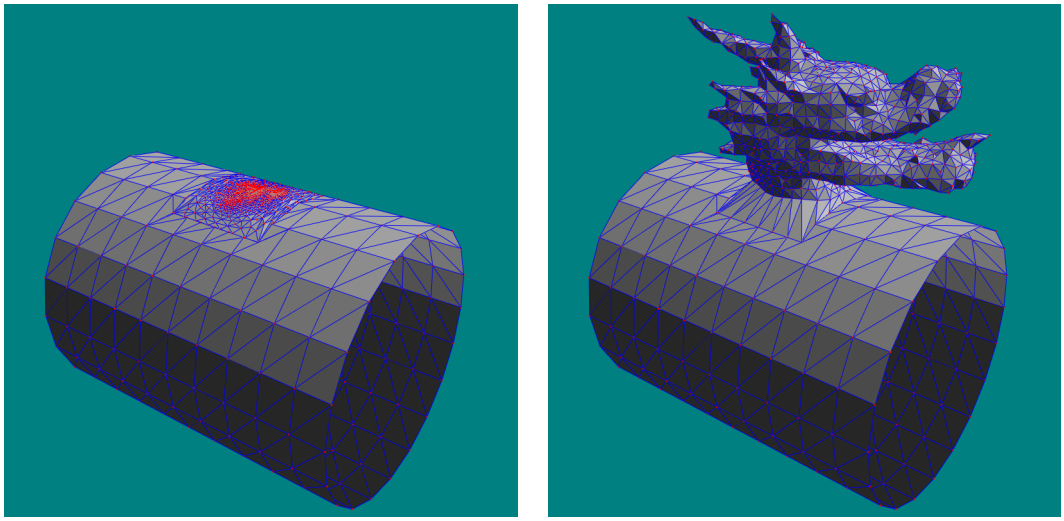


図 3.3: 実行結果：龍頭

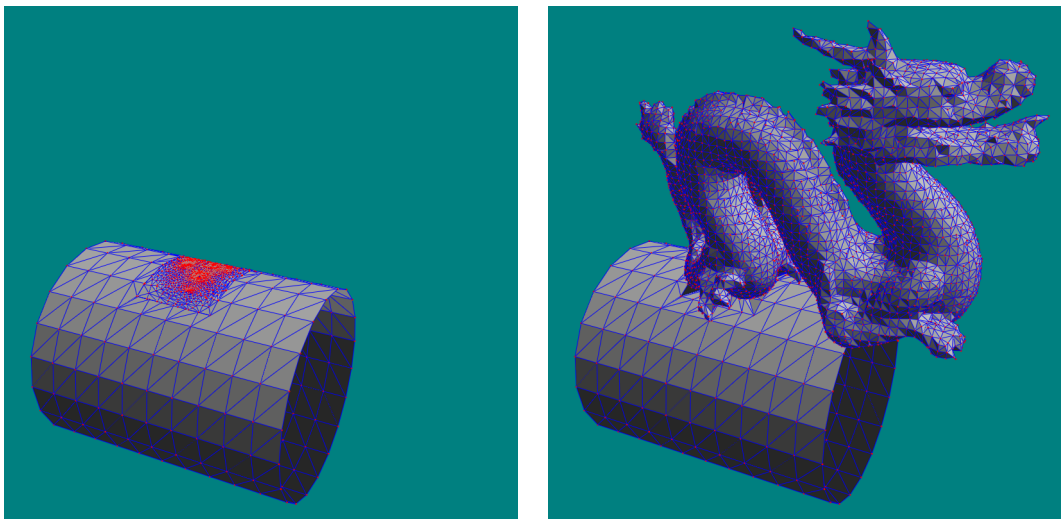


図 3.4: 実行結果：龍

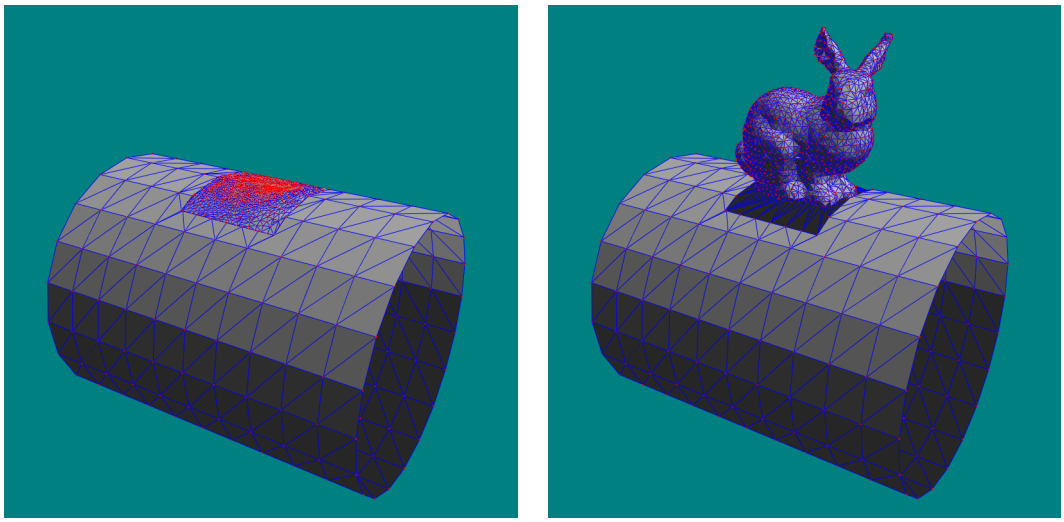
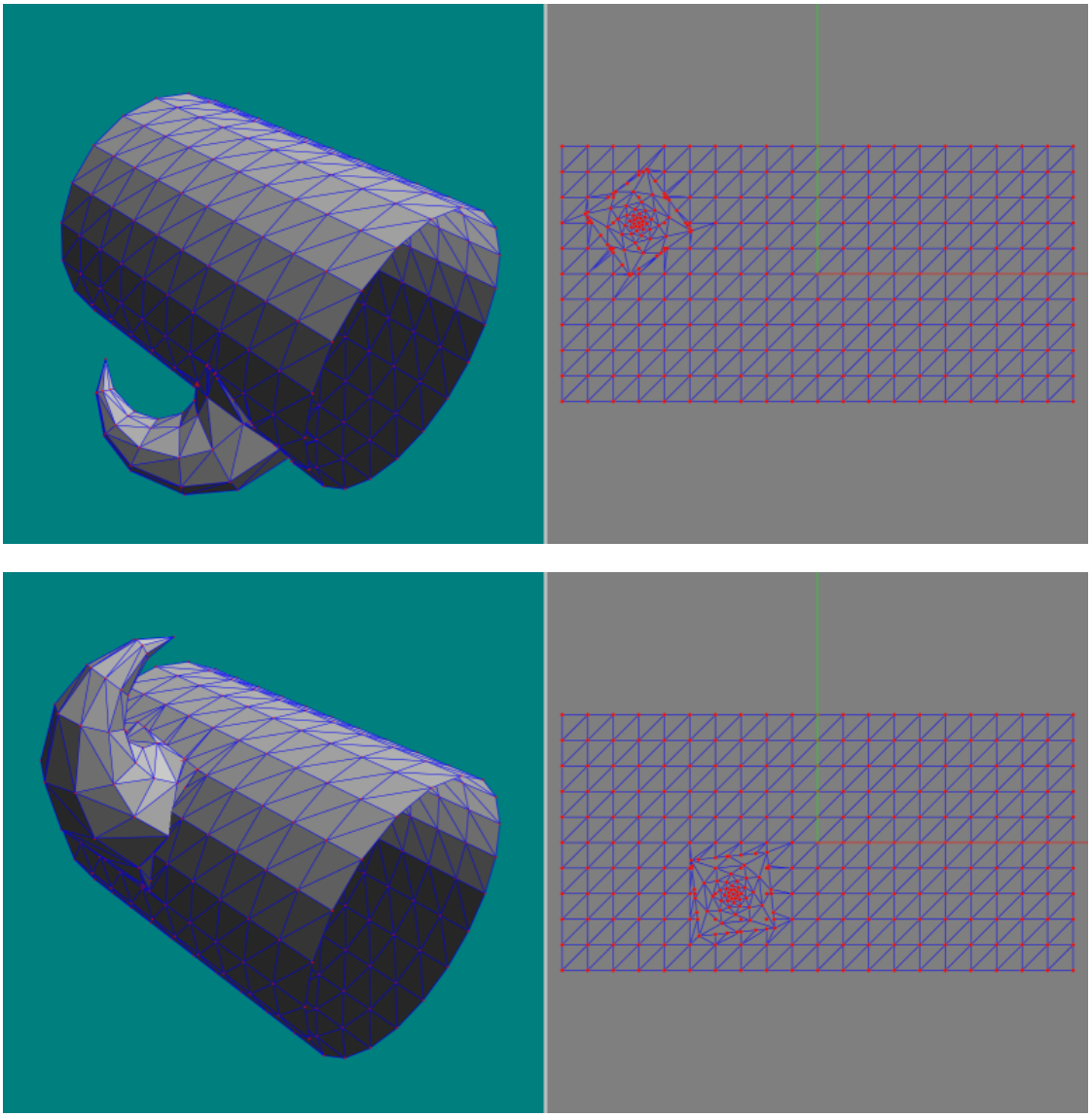


図 3.5: 実行結果：うさぎ

3.3　局所変形アニメーション

局所変形アニメーションについての結果を示す。まず、マッピング位置を動的に変更し、合成メッシュの生成を以下に示す。



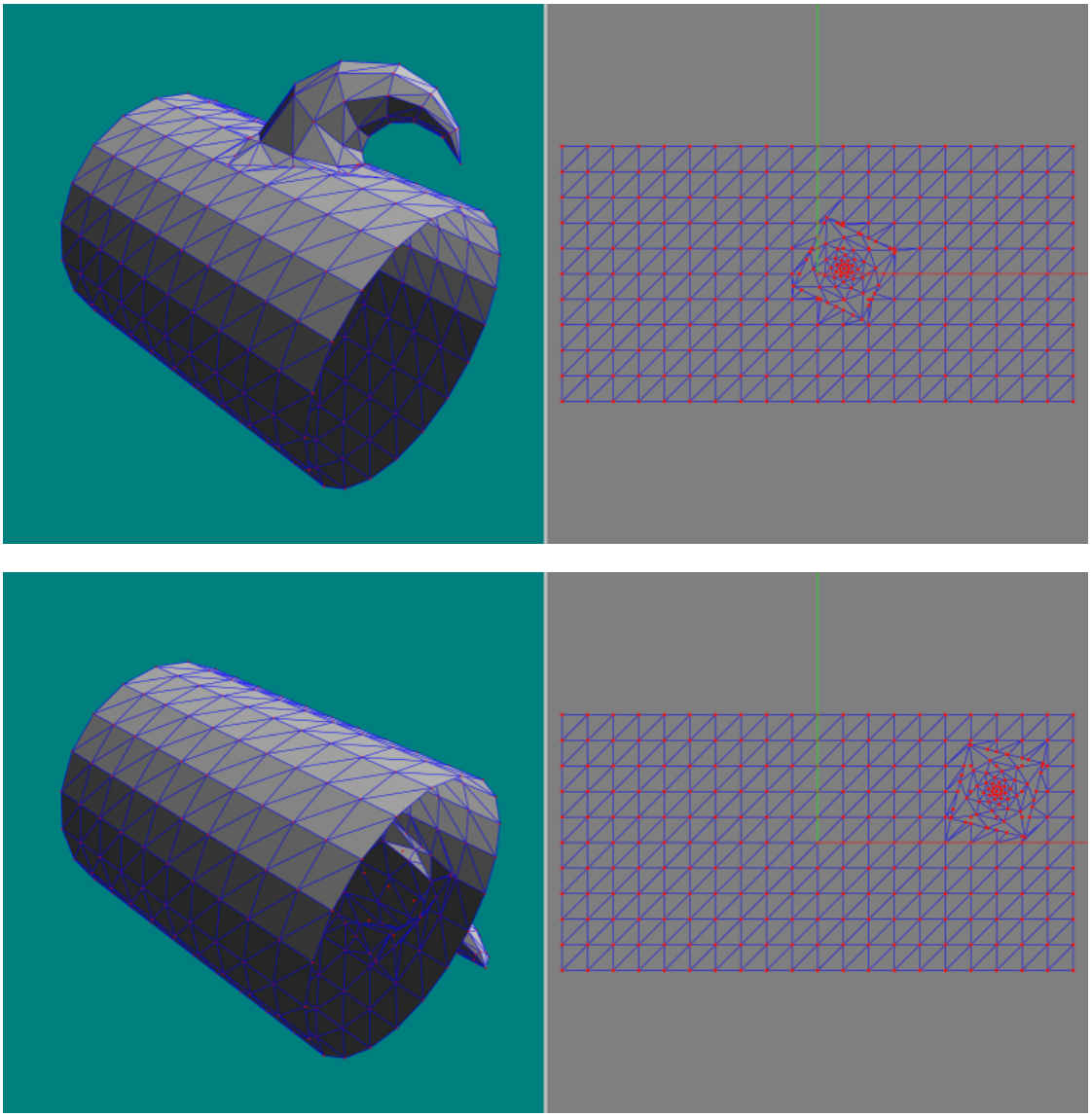


図 3.6: 実行結果：マッピング位置の変更

図 3.6 は、マッピング位置を変更して合成メッシュを生成した図である。左側は 3 次元空間上のメッシュ、右側は 2 次元空間上のメッシュである。

さらに、マッピング位置の変更を行うアニメーションを生成し、その際の描画速度を計測した。描画速度の単位は FPS(Frame Par Second) であり、1 秒間に可能な描画処理の回数を表す。検証方法には、ループ数の異なる 2 種類の付加メッシュを用いた。マッピング位置の変更のみ、変位マップの変位アニメーションのみ、マッピング位置の変更と変位マップの変位アニメーションを同時に行なった

場合の 3 パターンで FPS を計測した。FPS は 5 分間の平均 FPS である。付加メッシュのループ数と FPS の関係は以下の 表 3.3 とおりである。

表 3.3: アニメーション速度の測定

名前	ループ数	マッピング位置のみ	変位のみ	両方
つの	88	43.324	139.087	38.261
凸	288	32.174	99.324	28.124

表 3.3 より、付加メッシュのループ数が増えると FPS は低下するが、いずれもリアルタイム性は実現できたと言える値となっている。変位アニメーションのみのときに FPS が倍以上の値となっているのは、計算量が多くなる基礎メッシュのトリミング処理が最初の 1 度だけでしか行わないからである。

3.4 付加メッシュの再現性

変位マッピングを行った際、合成メッシュに現れる付加メッシュの形状の再現性について検証する。次の図は本手法による変位マッピングを行った合成メッシュである。いずれも、付加メッシュの位相を変位マップとして利用するのため、形状のディテールを失うことなく変位マッピングが行えている。 図 3.7 では、従来の変位マッピングで再現が難しかった耳の形状を失うことなく変位マッピングが行えている。 図 3.8 の先端部のような尖った形状や、回りこむような形状も再現できている。 図 3.9 のような細かい形状も正しく再現可能である。

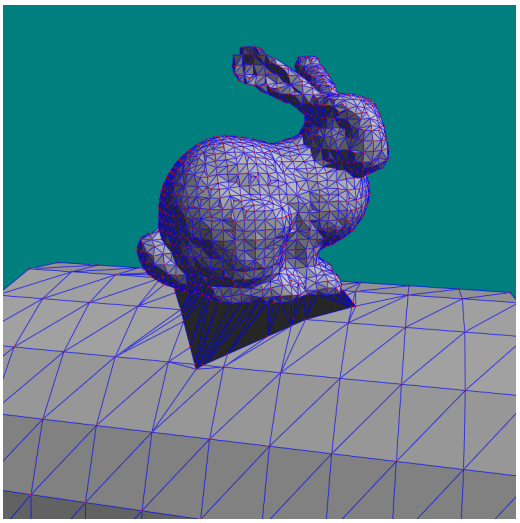


図 3.7: うさぎ

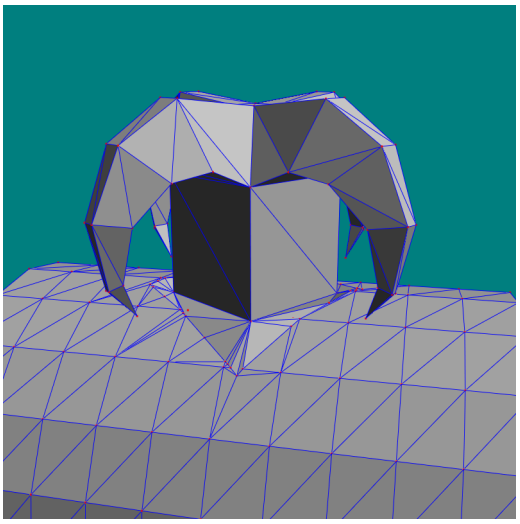


図 3.8: 王冠

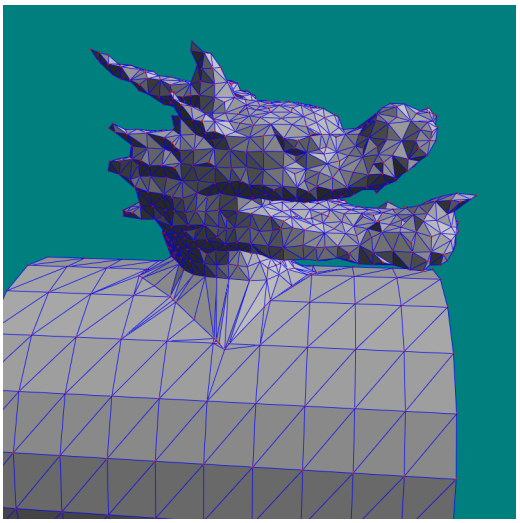


図 3.9: 龍頭

次に、複数の変位マップの同時マッピング時の変位マップ同士が重複した場合の再現性について検証した。 図 3.10 は、この検証で用いる 2 つの付加メッシュである。 図 3.11 は、2 つの変位マップが重なったときの様々な状況を示した図である。 図 3.11(a) は、変位マップの復元前のメッシュである。変位マップが互いに干渉していないのがわかる。 図 3.11(b) は、復元する割合が違う 2 つの変位マップを表している。復元アニメーションは変位マップごとに独立して行うことが可能である。 図 3.11(c) と 図 3.11(d) は、2 つの復元した変位マップが重なり合った状態を表している。変位マップ同士が重なり合ったとしても、どちらかの形状が歪むことはない。

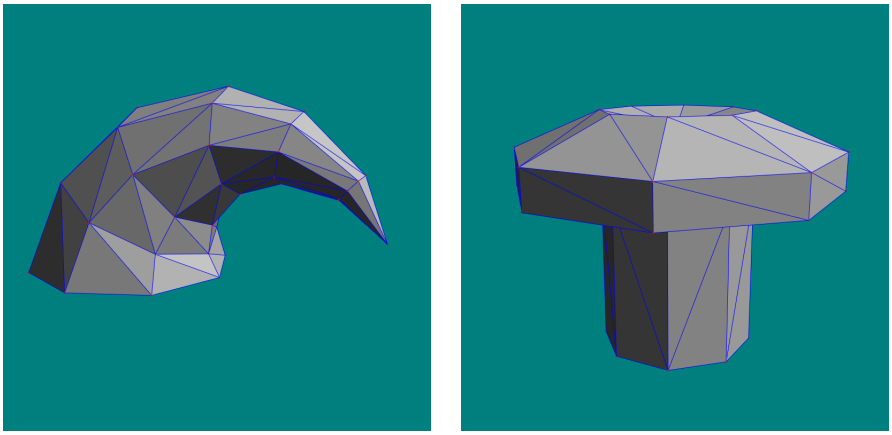
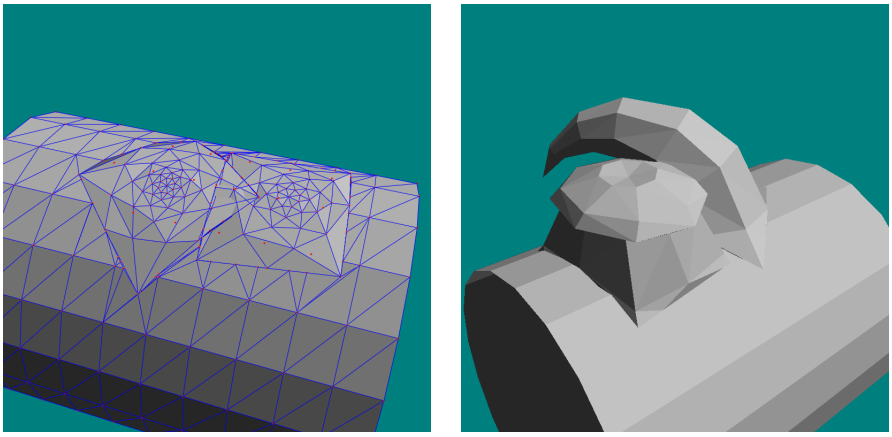
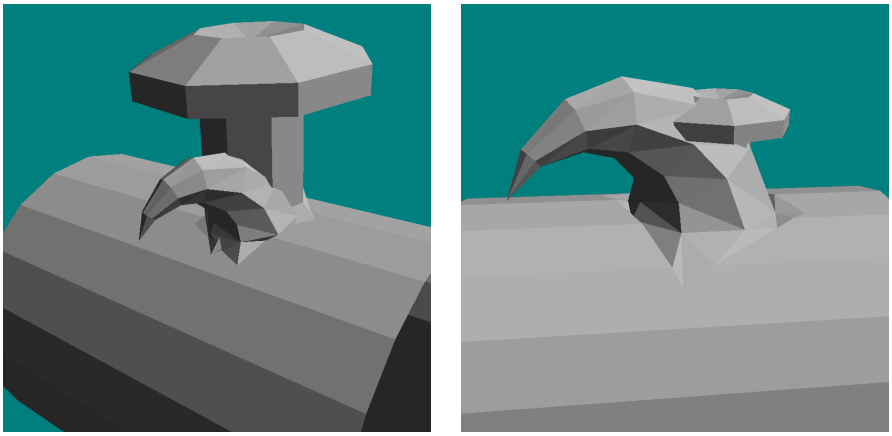


図 3.10: 基礎メッシュ



(a) (b)



(c) (d)

図 3.11: 変位マップの重なり

3.5 問題点

現在の問題点として、第１にマッピングを行った際に基礎メッシュの形状を再現しきれないという問題がある。変位マップをマッピングする領域を決定したあと、領域内のメッシュと変位マップを入れ替えているため、変位マップのループ数によっては基礎メッシュの元の形状が崩れていしまう。この問題の解決にはメッシュの集合演算などの処理が必要になるだろう。第２にメッシュ合成時のループの増大が挙げられる。現状では合成メッシュを生成する際に、微小なループが発生してしまう。近接頂点などを１つにまとめる等の処理が必要になるだろう。第３に、付加メッシュのループ数増加に伴う処理速度の低下が挙げられる。これは合成メッシュの生成に伴う計算量が増加する部分が大きい。この問題は合成演算のアルゴリズム改良により処理速度の向上が望めると考える。

第 4 章

おわりに

本研究では、リアルタイムコンテンツにおいて新たに、アニメーション可能な局所的形状変形手法を提案した。メッシュを局所的に変形するための技術である変位マッピングの概念を踏襲しつつ、変位マップに 2 次元メッシュを用いることにより、動的なアニメーションが可能な、局所的形状変形が可能となった。その結果として、従来の手法では難しかった、局所的形状変形による複雑な凹凸表現を実現した。また、変位マップの複数同時マッピングを行った際に、変位マップが重複しても正しく形状を復元することを実現し、局所的形状変形手法の有用性を高めた。本研究で提案した手法を用いることで、インタラクティブゲームやアニメーションなどのインタラクティブコンテンツでの局所的形状変形の表現の幅を拡げるという目的において、大いに役立つだろう。

今後の展望として、 3.5 節で問題点としてあげた合成処理の更なる高速化手法の考察が挙げられる。アニメーション可能な局所的形状変形は実現できたが、あまりにメッシュのループ数が増加すると、リアルタイムでのアニメーションが難しくなる。高速なメッシュ合成が可能になれば、更なる局所的形状変形表現の向上に繋がるだろう。

本研究は芸術科学会第 26 回 NICOGRAPH 論文コンテストにおいて”ベクター形式による変位マップアニメーション”[35] として発表した内容を含む。

謝辞

本研究を締めくくるにあたり、研究の指針から開発の手法、論文の執筆と幅広いご指導ご教授を頂きました、本校メディア学部の渡辺大地先生、並びに副査の三上浩司先生と宮岡伸一郎先生に厚く感謝いたします。そして、様々な相談に乗ってくださった先輩方や後輩達、多くの苦楽を共にした研究室の仲間たちに感謝いたします。

我唯一唯我也。何は無くとも地球は廻る。

参考文献

[1] Doo, D. and Sabin, M., "Behaviour of recursive division surfaces near extraordinary points," Computer-Aided Design 10(6), 356–360 (1978).

[2] Catmull, E. and Clark, J., "Recursively generated B-spline surfaces on arbitrary topological meshes," Computer-Aided Design 10(6), 350–355 (1978).

[3] Loop, C. T., "Smooth subdivision surfaces based on triangles /," (1987).

[4] 徳山 喜政, 今野 晃市, 曽根 順治, and Janaka Rajapakse R.P.C., "曲線メッシュをベースにした細分割曲面の局所変形," 芸術科学会論文誌 9(1), 1–9 (2010).

[5] Khodakovsky, A. and Schröder, P., "Fine level feature editing for subdivision surfaces," in [Proceedings of the fifth ACM symposium on Solid modeling and applications], SMA '99, 203–211, ACM, New York, NY, USA (1999).

[6] Biermann, H., Martin, I. M., Zorin, D., and Bernardini, F., "Sharp features on multiresolution subdivision surfaces," Graph. Models 64, 61–77 (March 2002).

[7] Sederberg, T. W., Zheng, J., Sewell, D., and Sabin, M., "Non-uniform recursive subdivision surfaces," in [Proceedings of the 25th annual conference on Computer graphics and interactive techniques], SIGGRAPH '98, 387–394, ACM, New York, NY, USA (1998).

[8] Zorin, D., Schröder, P., and Sweldens, W., "Interactive multiresolution mesh editing," in [Proceedings of the 24th annual conference on Computer graphics and interactive techniques], SIGGRAPH '97, 259–268, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1997).

[9] Kanai, T., Suzuki, H., Mitani, J., and Kimura, F., "Interactive mesh fusion based on local 3d metamorphosis," in [Proceedings of the 1999 conference

on Graphics interface '99], 148–156, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999).

[10] Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., and Shum, H.-Y., "Mesh editing with poisson-based gradient field manipulation," ACM Trans. Graph. 23, 644–651 (August 2004).

[11] Fu, H., Au, O., and Tai, C., "Effective derivation of similarity transformations for implicit Laplacian mesh editing," in [Computer Graphics Forum], 26(1), 34–45, John Wiley & Sons (2007).

[12] Sharf, A., Blumenkrants, M., Shamir, A., and Cohen-Or, D., "Snappaste: an interactive technique for easy mesh composition," Vis. Comput. 22, 835–844 (September 2006).

[13] Lin, J., Jin, X., and Wang, C. C. L., "Fusion of disconnected mesh components with branching shapes," Vis. Comput. 26, 1017–1025 (June 2010).

[14] Cook, R. L., "Shade trees," SIGGRAPH Comput. Graph. 18, 223–231 (January 1984).

[15] 加藤 良章, "リアルタイム 3DCG における薄膜厚みの動的変化を考慮した構造色描画手法," (2008).

[16] 西川 善司, 『ゲーム制作者になるための 3D グラフィックス技術』, 株式会社インプレスジャパン (2009).

[17] Krishnamurthy, V. and Levoy, M., "Fitting smooth surfaces to dense polygon meshes," in [Proceedings of the 23rd annual conference on Computer graphics and interactive techniques], SIGGRAPH '96, 313–324, ACM, New York, NY, USA (1996).

[18] Lee, A., Moreton, H., and Hoppe, H., "Displaced subdivision surfaces," in [Proceedings of the 27th annual conference on Computer graphics and interactive techniques], SIGGRAPH '00, 85–94, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000).

[19] Loop, C., Schaefer, S., Ni, T., and Castaño, I., "Approximating subdivision surfaces with gregory patches for hardware tessellation," ACM Trans. Graph. 28, 151:1–151:9 (December 2009).

[20] Kautz, J. and Seidel, H.-P., "Hardware accelerated displacement mapping for image based rendering," in [No description on Graphics interface 2001], GRIN'01, 61–70, Canadian Information Processing Society, Toronto, Ont., Canada, Canada (2001).

[21] Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., and Shum, H., "Generalized displacement maps," in [Eurographics Symposium on Rendering], 2004, 227–234 (2004).

[22] Hirche, J., Ehlert, A., Guthe, S., and Doggett, M., "Hardware accelerated per-pixel displacement mapping," in [Proceedings of Graphics Interface 2004], GI '04, 153–158, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (2004).

[23] 渡辺 大地, 千代倉 弘明, "任意三角形メッシュからの特徴稜線抽出," 電子情報通信学会論文誌. D-II, 情報・システム, II-パターン処理 83(5), 1344–1352 (2000-05-20).

[24] Tutte, W., "How to draw a graph," Proceedings of the London Mathematical Society 3(1), 743 (1963).

[25] Maillot, J., Yahia, H., and Verroust, A., "Interactive texture mapping," in

[Proceedings of the 20th annual conference on Computer graphics and interactive techniques], SIGGRAPH '93, 27–34, ACM, New York, NY, USA (1993).

[26] Floater, M. S., "Mean value coordinates," Comput. Aided Geom. Des. 20, 19–27 (March 2003).

[27] Sheffer, A. and de Sturler, E., "Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening," Engineering with Computers 17, 326–337 (2001).

[28] Sheffer, A., Lévy, B., Mogilnitsky, M., and Bogomyakov, A., "Abf++: fast and robust angle based flattening," ACM Trans. Graph. 24, 311–330 (April 2005).

[29] Lévy, B., Petitjean, S., Ray, N., and Maillot, J., "Least squares conformal maps for automatic texture atlas generation," ACM Trans. Graph. 21, 362–371 (July 2002).

[30] "A fast and simple stretch-minimizing mesh parameterization (figures 1 and 9)," in [Proceedings of the Shape Modeling International 2004], 390–, IEEE Computer Society, Washington, DC, USA (2004).

[31] Sheffer, A., Praun, E., and Rose, K., "Mesh parameterization methods and their applications," Found. Trends. Comput. Graph. Vis. 2, 105–171 (January 2006).

[32] OpenGL.org, "OpenGL." `http://www.opengl.org/`.

[33] 渡辺 大地, "リアルタイムグラフィックスのためのツールキットに関する研究," 修士論文, 慶應義塾大学大学院政策・メディア研究科 (1996).

[34] 渡辺　大地, “Fine Kernel Tool Kit System.” http://fktoolkit.sourceforge.jp/.

[35] 武田 巧視, 渡辺 大地, “ベクター形式による変位マップアニメーション,” 第 26 回 NICOGRAPH 論文コンテスト (2010).

発表論文

武田巧視, 渡辺大地, ”ベクター形式による変位マップアニメーション”, 第 26 回 NICOGRAPH 論文コンテスト (2010).