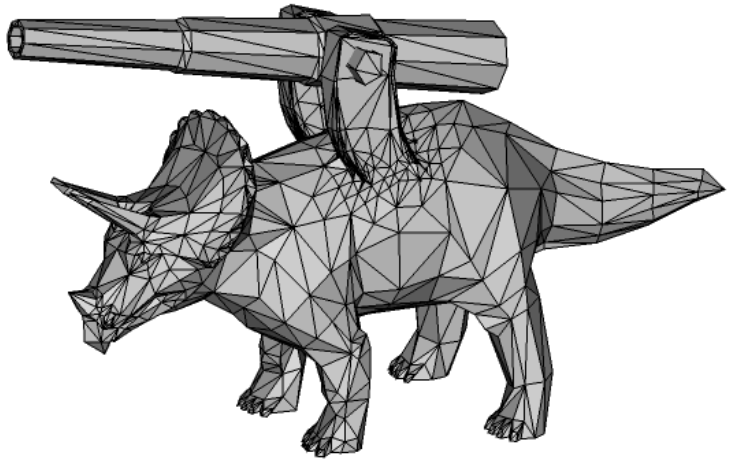
局所的細分割に基づく
任意 3 角形メッシュの接続操作

A Joining Operation of Arbitrary Triangular Meshes Based on Local Subdivision



2001 年 1 月 16 日

山田 英史

Hideshi Yamada

慶應義塾大学 大学院 政策・メディア研究科

Graduate School of Media and Governance

KEIO UNIVERSITY

修士論文要旨　2000 年度 (平成 12 年度)

局所的細分割に基づく任意 3 角形メッシュの接続操作

3 角形メッシュによる 3 次元形状表現は, 3 次元コンピュータグラフィックスの最も基本的な表現形式である. 3 角形メッシュ表現は, 3 次元形状を自由な位相で構成できるので滑らかな形状や微少な凹凸が混在する細かい特徴を持つ複雑な形状を表現できる. 近年のハードウェアの進歩により, 大容量の 3 角形メッシュモデルを処理することができるようになったため, 高品質な 3 角形メッシュモデルを効率よく生成, 編集する技術が必要とされている.

3 次元形状モデリングの基本操作のうち, 複数の基本形状を組み合わせてモデリングを行うものとして貼り付け操作に注目する. 過去に 3 角形メッシュの貼り付け操作を実現しているものにメッシュ・フュージョンと呼ばれる操作が提案されている. しかしこの方法は, 不必要な領域のメッシュ数の増加, ユーザーによる接続領域指定が必要であることなどを問題点として持つ. より簡易な操作で 3 次元形状の貼り付け操作を実現する手法が望ましい.

本論文では、3 角形メッシュ同士を接続して複雑な形状をより簡単に作成するための新しいモデリング操作を提案する. 本操作は、3 角形メッシュの局所的な細分割操作を基本にしている。入力された 2 つの 3 角形メッシュの干渉からメッシュを局所的に細分割し, 干渉領域の接続操作によって結合された 3 角形メッシュを得る. 3 角形メッシュの局所的細分割により干渉領域を追跡するので, メッシュ数の不要な増加を抑えながらメッシュの形状を保存することができる. また, 細分割操作によってメッシュ形状を保存するので, 滑らかな部分と尖った部分を合わせもつような形状同士の接続を行うことができる.

本手法によって、メッシュ形状を 3 次元空間に配置するだけで 3 角形メッシュの任意の一部分を他のメッシュ上に貼り付けるといった 3 次元メッシュ形状の貼り付け操作を実現する. また本操作により既存の 3 角形メッシュ形状データ, またはその一部を自由に貼り付け操作へと利用することができ, 3 角形メッシュ接続操作をより簡易な方法で実現できる. 以上の操作を計算機上に実装し、その有効性を検討した.

キーワード

1. コンピュータ・グラフィックス	2. 3 角形メッシュ
3. 3 次元形状モデリング	4. メッシュ接続
5. 細分割曲面	

慶應義塾大学 大学院 政策・メディア研究科
山田　英史

Abstract of Master's Thesis Academic Year 2000

A Joining Operation of Arbitrary Triangular Meshes Based on Local Subdivision

Models represented by triangular meshes are the most widely used in the Computer Graphics field. As triangular mesh is constructed with arbitrary topology, it can represent a complex shape which has fine characteristics, such as a model with smooth or bumpy. While recent advance in computer hardware enables to process a large amount of triangular meshes, there is increasing demand to construct and to edit high quality triangular mesh models efficiently.

This paper focuses on an operation called pasting in the basic operations of 3D geometric modeling, in which one mesh is attached to another mesh. Previously, an operation called Mesh Fusion has been puslished to paste the meshes. However this approach has some issues that the number of faces would increase dramatically and that the user has to select boundaries to make a correspondence between two meshes. Especially the latter issue is often troublesome in the case of complex boundaries. It is desirable to realize an easier operation of 3D mesh pasting.

This paper proposes a new modeling operation to construct a complex shape easily by connecting triangular meshes. This operation is based on local subdivision of triangular meshes. Our method combine the meshes by subdividing only faces which are collided with each other and by stitching boundaries of meshes. By refining collided faces by local subdivision, it is possible to preserve the geometry of the meshes and to avoid the increase of the number of faces. Therefore this operation can connect shapes with preserving the smooth or sharp regions.

Using this method, we realize the 3D mesh pasting operation which attaches one mesh to another, by only placing meshes in 3D space. It utilizes the existing meshes or a part of them, then the efficient modeling can be done. We implement this operation on personal computer and discuss its availability.

Key Words

1. Computer Graphics	2. Triangular Mesh
3. 3D Geometric Modeling	4. Joining Mesh
5. Subdivision Surface	

Keio University Graduate School of Media and Governance
Hideshi Yamada

目 次

図 目 次

第 1 章

はじめに

1.1 3 次元 CAD/CG と 3 角形メッシュモデリング

近年, アニメーションの高品質化やインターネットの 3 次元コンテンツの普及にともない, 3 次元形状モデルの作成技術に対する要求が高まり, 様々な研究が行われている. このような形状モデルの表現には, いわゆるポリゴンモデルと呼ばれる多面体モデルや, NURBS(Non-Uniform Rational B-Spline) 曲面が用いられるが, 特にポリゴンモデルの中でも全ての面が 3 角形の 3 角形メッシュモデルに対する様々な技術が現れている.

3 角形メッシュは, CG の分野で最も普及しているデータ表現形式である. 特徴として, グラフィックスハードウェアに適していること, グラフィックスライブラリのサポートを受けられること, 曲面モデルからの変換が容易であることなどが挙げられる. また多くのモデリングシステムでサポートされている形状表現であり, システム間でのデータのやりとりが容易である. また形状モデリングにおいて, 従来の CAD や CAID(Computer Aided Industrial Design) システムは, 形状モデルを高い精度で作り込んでいくことはできても, 様々な形状を考える発想支援には使いにくいという欠点があった. 一方 3 角形メッシュは, 機械デザインのデータ定義には精度の問題から使うのは難しいとされていたが, 概形でもよいから自在に形状を表現したいという要求が高まっており, そのような用途に対して, 3 角形メッシュモデリング技術は有効である.

メッシュ編集操作 (Mesh Editing) には, 変形グリッドを用いた FFD(Free Form Deformation) [29, 23, 10] や, 多面体間の形状補間を行うモーフィング [15, 31, 19] などの操作がある. これらの処理の一部は既に 「変形のためのツール」として市販ソフトに実装されている. また 3 角形メッシュの凹凸を滑らかにしたり, 粗いメッシュを丸めて滑らかにする処理に平滑化 (fairing) [17, 5] がある.

1.2 貼り付け操作に基づく 3 次元形状モデリング

本研究の目的は, 3 角形メッシュモデリングの中で 2 つの既存の形状を使って新しい形状を生成するための方法を提供することである. ここでは, 任意の 3 次元形状, またはその一部を他の形状に接続するという操作に焦点を当てる. この操作は, 一般に貼り付け (pasting) と呼ばれる操作に相当し, 2 次元の描画用ソフトウェアやペイントソフトウェアなどでは必ず実装されている基本的な操作である. 本論文では, 3 角形メッシュモデリングの中でも, 既存の 3 角形メッシュ, もしくはその一部を別の 3 角形メッシュ上へと貼り付けるようなモデリング操作に関して考える. そのような操作を 3 次元形状同士で実現するためには, 2 つの形状の境界を検出し滑らかに接続するための方法が必要である.

2 つの形状を接続するという操作を実現する 1 つの方法として, 集合演算 [24] を用いて

2 つのメッシュの和を取り, 境界付近にブレンディング [12] を適用するという方法が考えられる. しかし, 集合演算を実装している市販ソフトウェアはプリミティブや自由曲面を対象としたものが多く, メッシュを対象とした集合演算とブレンディングの両方の機能を持つものは少ない.

また, メタボール (metaball) [14] や soft object [9] など陰関数によって表現された基本形状同士を融合接続する方法がある。この方法では, 球や楕円体などの基本形状を 3 次元空間内に配置し融合演算していくことにより, 滑らかに接続された 3 次元形状を定義できる. メタボールは, 中心となる点からの距離とともに減衰する影響力 (引力) の和が一定値となるような点によって形成される曲面である. 複数のボールがあると, これらを包むゴムの膜のような曲面を生成し, 点の位置の移動によって分離した曲面を連続的に融合できるという特徴をもつ. そのため, 雲や人体などの有機的な形状の表現に用いられることが多い. また過去には, 基本形状間の切り貼り操作 [26], B スプライン曲面の貼り付け操作 [2] などの研究も見られる. しかし、これらの手法はメッシュを対象とはしていない。メッシュの場合は、2 つの形状同士の境界線の対応付けが複雑であり、これらの手法をそのまま適用するのは難しい。

一方金井ら [16] は、メッシュの融合操作 (Mesh Fusion) と呼ばれる局所的なモーフィングによる 3 角形メッシュの切り貼り手法を提案した。この手法は、ユーザーが指定した境界によってメッシュ形状の一部を切り取り、調和写像 (harmonic maps) [6] を用いてメッシュ形状を合成する。そして合成されたメッシュの対応関係を求め, 補間により頂点座標を求めることで接続形状を定義する。このメッシュの融合操作は, 接合部分の形状の変化を局所化するため高速な処理が可能であり, またパラメータを変化させることで形状の操作, 修正を行うことができる. しかし, この方法では調和写像を利用した形状の合成を行うため, 明らかに不必要な部分の 3 角形数が増加してしまうという問題がある. またメッシュの張り付け領域の境界をユーザー入力によって行っており, 境界の形状が複雑なケースではユーザーインプットの負担が大きい.

このように, 現在までに 3 角形メッシュの接続操作へと応用できる決定的な手法は提案されていない.

1.3 局所的詳細化に基づく 3 角形メッシュの接続操作

そこで本論文では, 既存の 3 角形メッシュ形状同士を接続してより複雑な形状を作成するための新しいモデリング操作を提案する. 本操作は, 3 角形メッシュの局所的な細分割操作 [32, 18] を基本にしている. 切り貼りという点から考えて, 接合付近では一方の形状から

他方の形状へと形が変化し, その他の接合領域から遠い部分では切り貼り操作前の形状であることが望ましい. このことを実現するために, 本操作では入力された 2 つの 3 角形メッシュの干渉からメッシュを局所的に細分割する. これにより, メッシュ形状の変形を局所化することができる. 局所的に細分割した 3 角形メッシュの干渉領域の接続操作によって結合された 3 角形メッシュを得る. このように 3 角形メッシュの局所的細分割により干渉領域を絞りこむため, メッシュ数の不要な増大を抑えながら細かな形状の特徴を保存することができる. また, 細分割操作によってメッシュ形状を保存するので, 滑らかな部分と尖った部分を合わせもつような形状同士の接続を行うことができるのも利点の一つである.

本手法によって、メッシュ形状を 3 次元空間に配置するという簡易な操作で, 3 角形メッシュの任意の一部分を他のメッシュ上に貼り付けるといった 3 次元メッシュ形状の貼り付け操作を実現する. 本操作により, 既存の 3 角形メッシュ, またはその一部を自由に組み合わせて複雑な形状を作成することが可能であり, 3 角形メッシュ形状のインタラクティブなモデリングを支援することができる. 本論文で提案する操作を計算機上に実装し, その有効性を確認した.

1.4 本論文の構成

本論文の構成は以下の通りである. まず第 2 章では, 研究の背景として 3 次元形状の貼り付け操作に関連する研究について解説する. 次に第 3 章では, 細分割曲面表現と細分割曲面の角の拡張表現, 細分割曲面フィッティングについて説明する. さらに, 細分割曲面表現を任意の領域にのみ適用する局所的細分割手法について説明する. 続く第 4 章では, 本接続手法の詳細について述べる. また第 5 章では, 実際のシステムへの本操作の実装について解説する. 第 6 章では, 実行例を示しながら本操作を考察し評価する. 最後に第 7 章では, まとめと今後の展望を述べる.

1.5 本論文で用いる表記

本節では, 本論文で用いる表記に関して説明する.

- 形状モデルといった要素の集合を表す場合は $\mathcal{M}$ のようにカリグラフィクス文字で表す.
- 位相要素を表す場合は頂点 v や稜線 e, 面 f のように斜体細字で表す. また要素を区別するために, 要素の番号を右下に添え字として記す.
- 3 次元ベクトルは $\mathbf{v}$ のように太字で記す.

- スカラ量は k のとおり斜体細字で記す.
- ベクトルの内積は, $\mathbf{a} \cdot \mathbf{b}$ のように $\cdot$ で表す.
- ベクトルの外積は, $\mathbf{a} \times \mathbf{b}$ のように $\times$ で表す.
- ベクトルのノルムは $|\mathbf{a}|$ のように表す.

第 2 章

研究背景

本章では, 本論文で提案する操作の研究背景として, 3 次元形状の貼り付け操作に関する主要な研究を解説する.

一方の形状を他方の形状へ貼り付けるといった操作は, 2 次元グラフィックスソフトウェアでは必ず実装されている不可欠な機能であり, 3 次元においてこの貼り付け操作を実現するための研究がなされている.

2 つの形状を接続するという操作を考えたときに, その操作を実現する 1 つの方法として, 集合演算 [24] を用いて 2 つのメッシュの和を取り, 境界付近にブレンディング [12] を適用するという方法が考えられる. しかし, 集合演算を実装している市販ソフトウェアはプリミティブや自由曲面を対象としたものが多く, メッシュを対象とした集合演算とブレンディングの両方の機能を持つものは少ない.

一方, 貼り付け操作は, 陰関数曲面形式では容易に実現することが可能であり, 多数の手法が提案されている. 陰関数によって表現された基本形状同士を融合接続する方法として, メタボール (metaball) [14] や soft object [9] などがある。これらの方法では, 球や楕円体などの基本形状を 3 次元空間内に配置し, その中心となる点からの距離とともに減衰する影響力 (引力) の和が一定値となるような点によって曲面を形成する. 複数の基本形状がある場合, これらの形状を包み込むゴムの膜のような曲面となり, 基本形状の位置によって分離した曲面を連続的に融合できるという特徴をもつ. そのような特徴を持つため, 雲や人体といった有機的な形状の表現に用いられることが多い.

Stander ら [27] は, この融合接続をインタラクティブなモデリング環境において利用するために, 陰関数形式で定義した曲面の融合形状を多角形モデルへと変換する手法を提案した. これは, 基本形状の減衰する影響力が一定となる点を高速に計算し, 接続情報を持った多角形モデルへと変換する手法である. この方法では, 基本形状の位置を移動することで動的に多角形モデルを再構築して表示し, インタラクティブなモデリングを可能にしている. また Decaudin ら [3] は, このような陰関数形式で表現された形状の融合演算の応用として, 陰関数で定義された基本形状とメッシュ形状を融合演算する手法を提案した. この手法は, 陰関数形式の基本形状の影響力が一定値となる点にメッシュの頂点を移動することで, メッシュ形状を滑らかに変形させることを可能にしている. また過去には, 基本形状間の切り貼り操作 [26], B スプライン曲面の貼り付け操作 [2] などの研究も見られる.

しかし, これらの手法はいずれもメッシュ間の接続操作を対象とはしていない. メッシュの場合は, 2 つの形状同士の境界線の対応付けが複雑であり, これらの手法をそのまま適用することは難しい. また, 利用できる形状が基本形状に限られるので, 複雑な形状を生成するには困難を伴う. さらに, 他のモデリング操作と併用して用いるためには, 陰関数曲面の

多角形モデルやパッチ曲面への変換が必要である. 多角形から陰関数への逆変換は容易でないため, 他形式へ変換した陰関数表現形状を再度編集することは困難である.

一方金井ら [16] は, メッシュの融合操作 (Mesh Fusion) と呼ばれる局所的なモーフィングによる 3 角形メッシュの切り貼り手法を提案した. この手法は, ユーザーが指定した境界によってメッシュ形状の一部を切り取り, 調和写像 [6] を用いてメッシュ形状を合成する. そして合成されたメッシュの対応関係を求め, 補間により頂点座標を求めることで接続形状を定義する. このメッシュの融合操作は, 接合部分の形状の変化を局所化するため高速な処理が可能であり, またパラメータを変化させることで形状の操作, 修正を行うことができる. しかし, この方法では調和写像を利用した形状の合成を行うため, 明らかに不必要な部分の三角形数が増加してしまうという問題がある. またメッシュの貼り付け領域の境界をユーザー入力によって行っており, 境界の形状が複雑なケースではユーザーインプットの負担が大きい.

メッシュの貼り付け操作は, 少ないユーザーインプットで行うことが望ましい. また, メッシュ形状は接合部分以外において元の形状を維持することが必要であり, そのためにメッシュ数の増加を抑えながら接続メッシュ形状を得る手法が必要である.

第 3 章

関連研究

本章では, 始めに本接続操作の基本となる局所的細分割操作の元となっている細分割曲面表現について述べる. 続いて, その細分割曲面表現を拡張し, 角張った部分と平滑な部分を合わせ持った形状を表現できるよう拡張した区分平滑細分割曲面について述べる. さらに, 任意の 3 角形メッシュに対して制御メッシュを当てはめるためのフィッティング手法について述べる. 最後に, メッシュの任意の面に対して細分割操作を行う局所的細分割手法について述べる.

3.1 細分割曲面

本手法で適用する Loop [21] の細分割手法について説明する. 細分割曲面は, 入力された 3 角形メッシュを制御メッシュとして細分割操作を繰り返し行い, 制御メッシュを詳細化した極限として表現される.

一般に, 細分割操作は面の細分割と頂点の再配置の 2 つの手順によって行われる. 面の細分割では, 制御メッシュに対して新しい頂点を加え面を細分割する. 頂点の再配置については元の頂点への重み付けによる平均化処理によって, 元の頂点と新しく生成した頂点へ新しい座標を割り当てる. その 2 つの手順を繰り返し実行することで, 細分割曲面は定義される.

Loop の手法は, 3 角形メッシュを基本としており, 3 角形メッシュの一つの面を 4 つの 3 角形へと細分割する. 全ての稜線に新しい頂点を加え 2 つの稜線へと分割し, 新しく生成した頂点を結ぶ稜線を作成して 3 角形面を 4 つの 3 角形へと分割する. これは一般に 4to1 細分割操作と呼ばれる. その 4to1 細分割の様子を図 3.1 に示す.

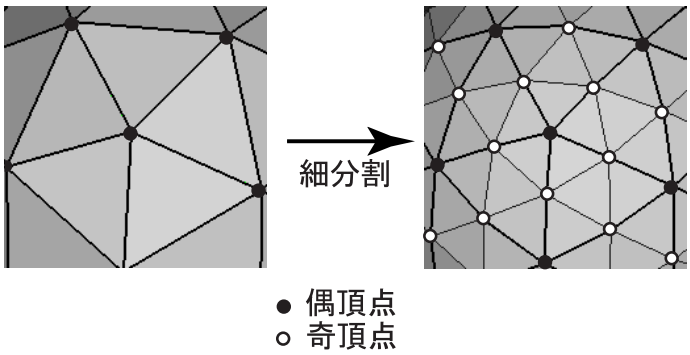


図 3.1: 4to1 細分割操作

この時, 新しく加えられた頂点を奇頂点 (*odd vertices*) , 元から存在する頂点を偶頂点 (*even vertices*) と呼ぶ. Loop の手法において 奇頂点の頂点座標は次式で求められる.

$$\frac{1}{8}(3p_1 + 3p_2 + p_3 + p_4),$$

p_1 と p_2 は稜線の 2 つの頂点座標を表し, p_3 と p_4 は稜線を共有する 3 角形の残りの頂点座標を表す. 奇頂点の新しい頂点座標は隣接する頂点座標の荷重平均によって得られる. 奇頂点の新しい頂点座標を荷重平均によって求めるマスクを図 3.2 (左) に示す.

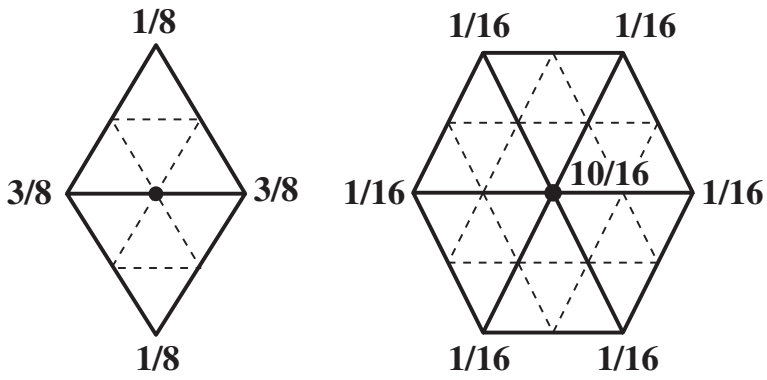


図 3.2: 細分割操作における頂点へのマスク

また, 偶頂点も隣接する頂点座標の荷重平均によって得られる. 偶頂点のうち稜線への接続数が 6 のものを正則頂点, 6 以外のものを特異頂点と呼び, それぞれ座標の求め方が異なる. 正則頂点は図 3.2 (右) のマスクで得られる. 一方, 頂点への稜線の接続数 k の特異頂点の座標は図 3.3 に示すマスクで求められる.

ここで, β は次の式で計算される.

$$\beta = \begin{cases} \frac{3}{16} & (k = 3) \\ \frac{1}{k}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k}\right)^2\right) & (k > 3) \end{cases} \tag{3.1}$$

境界上の頂点座標には同様な重み付けを行うことで座標を得る.

入力された制御メッシュに対し細分割操作を極限まで繰り返すと, 滑らかな曲面へと収束する. 極限で定義される曲面を細分割極限曲面と呼び, その曲面上の頂点を極限点と呼ぶ. 細分割曲面は正則頂点において C^2 連続となり, 特異頂点において C^1 連続となる. また, 細分割操作によって挿入される頂点は必ず正則頂点となる.

このように細分割曲面は細分割の極限によって曲面を定義するものであるが, 収束が十分に早いので, 数回の細分割だけでも, かなり滑らかなメッシュを生成することができる. そ

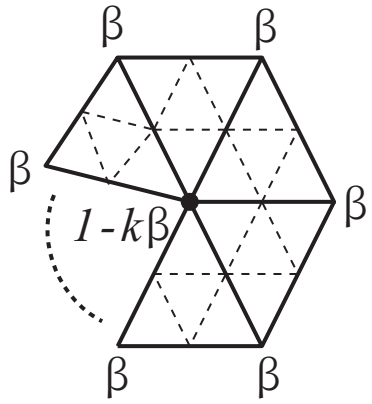
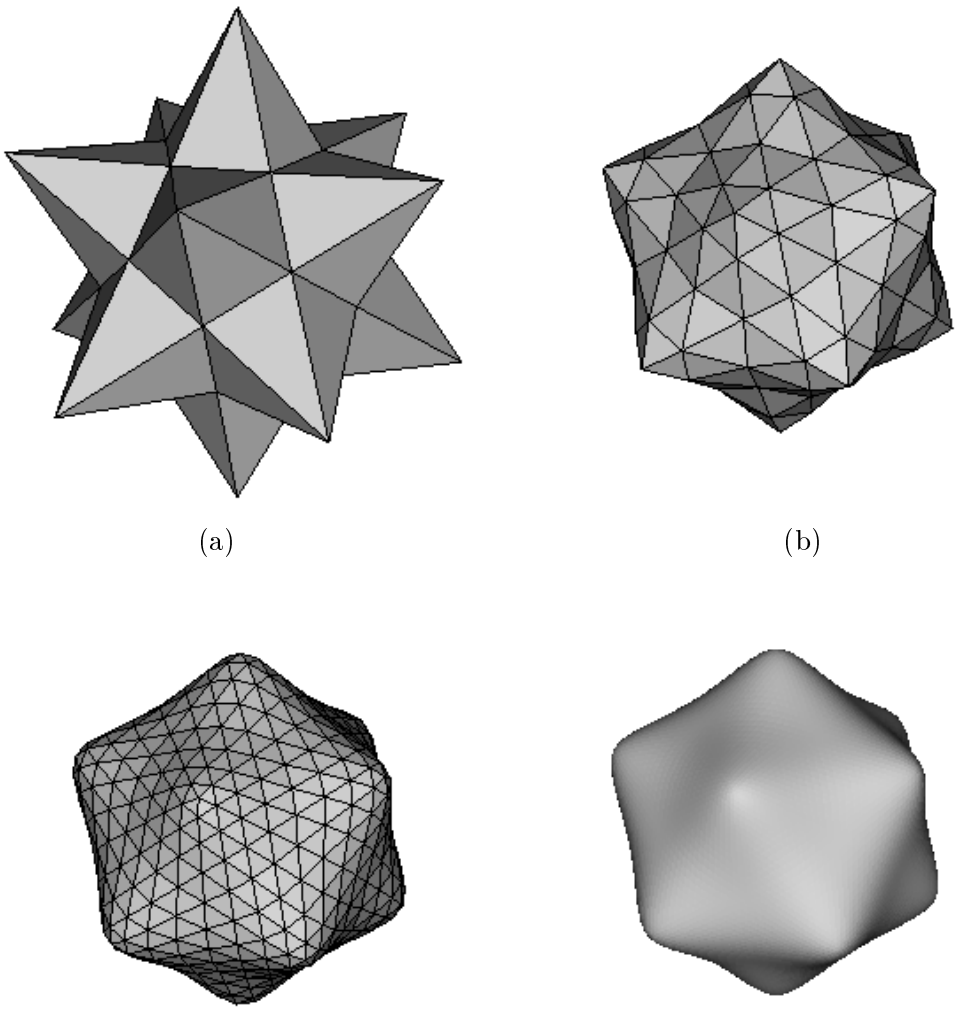


図 3.3: 特異頂点における頂点へのマスク

のため, 多重解像度表現 (Multiresolution) によるモデリング, アニメーション, データ圧縮などへの利用が研究されている [4, 32, 20].

図 3.4 に Loop の細分割手法を適用した例を示す. (a) は入力 3 角形メッシュ, (b) は 1 回,(c) は 2 回の細分割操作を適用したメッシュ, (d) は細分割極限曲面を表している.



(a) (b)

(c) (d)

図 3.4: Loop 手法による細分割操作

3.2 細分割規則の拡張による尖稜線の表現

細分割曲面において, 平滑でない部分を含む形状を表現できるよう拡張したものに区分平滑細分割曲面 (Piecewise smooth subdivision surface) [11] がある. この細分割曲面の拡張は, 3 角形メッシュ形状の平滑でない箇所の稜線に *sharp edge*(尖稜線) のタグを付けておき, その両側の面の接続における不連続性を表現するものである. これらの細分割規則は, *crease*(折り目), *corner*(角), *dart*(端) と呼ばれる鋭角な形状特徴を生成する. この区分平滑細分割規則による細分割曲面は, 折り目や角以外のあらゆる所で接平面連続を満たしている.

区分平滑細分割規則では, 3 角形メッシュの各頂点を頂点に接続する稜線の数と *shape edge* の配置に基づいて 5 つのタイプに分類する (図 3.5). *smooth vertex* は接続する *sharp edge* の数 s が 0 本の普通の頂点である. $s = 1$ のときが *dart vertex*, $s = 2$ のときが *crease vertex*, $s > 2$ のときが *corner vertex* となる. *crease vertex* は稜線の配置によってさらに 2 つに分類される. *crease vertex* の価数が 6 で *smooth edge* が 2 本ずつ折り目の両側に配置されているときこれを規則的 (regular) という. 境界上の *crease vertex* は価数が 4 のとき規則的である. その他の全ての *crease vertex* は 非規則的 (non-regular) に分類される.

また, 稜線細分割マスクに関しては 3 つのタイプを用いる. *sharp edge* のタグがない滑らかな稜線は *smooth edge* マスクにより細分割される. *sharp edge* の細分割マスクはその稜線の両側の頂点のタイプの組み合わせにより 表 1 のように決まる. 稜線細分割マスク 3 については, *regular crease vertex* の方の重みが 5 となる.

	dart	reg crease	non-reg crease	corner
dart	1	1	1	1
reg crease	1	2	3	3
non-reg crease	1	3	2	2
corner	1	3	2	2

表 1: 尖稜線の両端点タイプ別の細分割マスク

この区分平滑細分割曲面表現によりメッシュの中に滑らかな部分と角を持つ部分とを合わせ持つ形状を表現でき, パッチ曲面表現のような利用ができる. 図 3.6 に区分平滑細分割曲面の適用例を示す. 図 (a) の中の太線は, *sharp edge* タグを設定した稜線を示す. (a) は入力 3 角形メッシュ, (b) は 1 回,(c) は 2 回の細分割操作を適用したメッシュ, (d) は細分割極限曲面を表している.

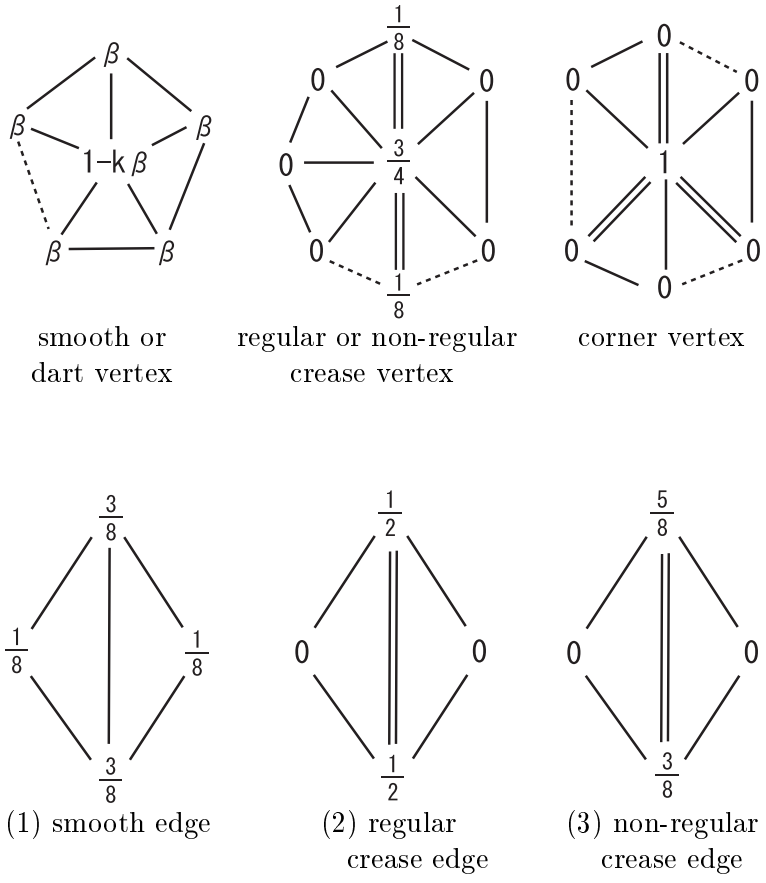
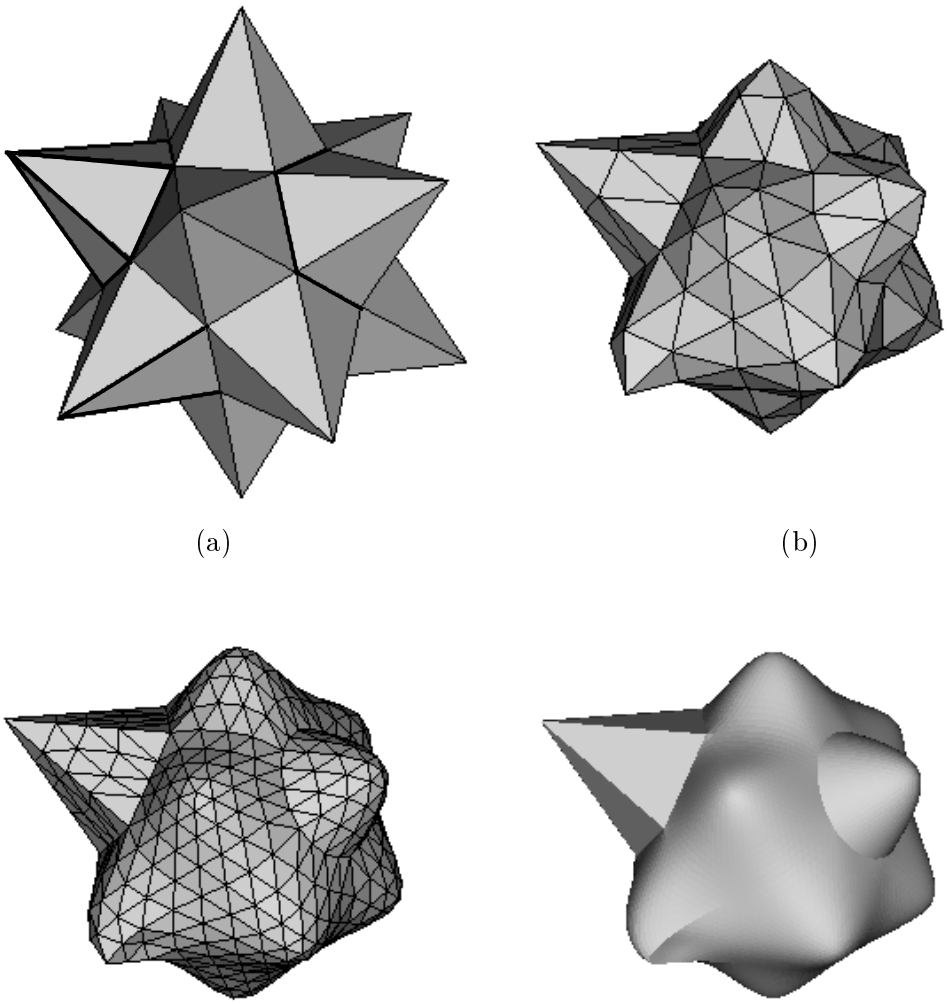


図 3.5: 区分平滑細分割曲面の頂点と稜線のマスク



(a) (b)

(c) (d)

図 3.6: 区分平滑細分割曲面

3.3　3 角形メッシュのフィッティング手法

ここでは, 任意の 3 角形メッシュに対して細分割曲面の制御メッシュをフィッティングする手法について述べる. 細分割操作によって入力形状を近似するために, 制御点座標と位相をまとめる制御メッシュを構築する. 制御点座標と位相は, ユーザーが指定する許容誤差 ε 以内に極限曲面が収まるように決定する. そのような手法には最小二乗法による補正 [11] や平滑化による補正 [8] などが挙げられる.

細分割近似曲面の生成は, 入力した 3 角形メッシュから抽出された頂点を p^{∞}, 制御メッシュの頂点を p としたとき, $p^{\infty} = Lp$ となるような制御点を見つけ出すことである. L は, 極限点を求めるための制御メッシュの頂点に対する重み付けである. 近似曲面の制御点を求めるには, $p = L^{-1}p^{\infty}$ となるような L^{-1} を得る必要があるが, 一般に L^{-1} の計算には非常に大きな計算が必要となる. そこで. 鈴木ら [28] は制御メッシュの各頂点に対して細分割極限点を求め. 入力モデルの頂点との距離の最小誤差問題として解く手法を提案した.

鈴木らの手法では, エネルギ関数 E は次のように定義される.

$$E = \sum_i E_i \ , \ E_i = |p_i^{\infty} - q_{i'}|^2$$

エネルギ関数は各制御点におけるエネルギ E_i の総和で定義され, エネルギ E_i は各制御点の極限点 p_i^{∞} と最近接点 $q_{i'}$ との距離の 2 乗で定義する. 曲面近似を実現するため, 制御メッシュの各制御点をエネルギ関数が最小化されるように移動し, 一定の閾値を下回るまで頂点座標の更新を行ったあと制御メッシュの細分割を行う. 各制御点の移動量 Δp_i は最近接点 $q_{i'}$ との距離を用いて求められ, 次の式で表される.

$$\Delta p_i \equiv p_i^{\infty} - q_{i'} \tag{3.2}$$

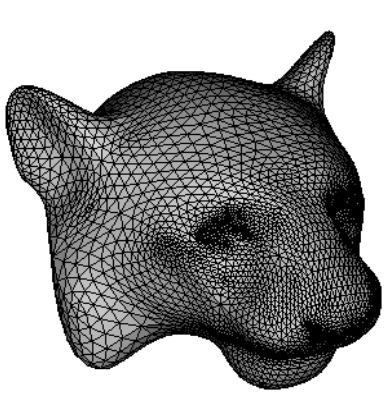
式 3.2で求められた各制御点 p_i における頂点誤差エネルギを最小化するために制御点を移動する. Δp_i に重み λ をかけたものを p_i の移動量とする.

$$p_i = p_i - \lambda \Delta p_i$$

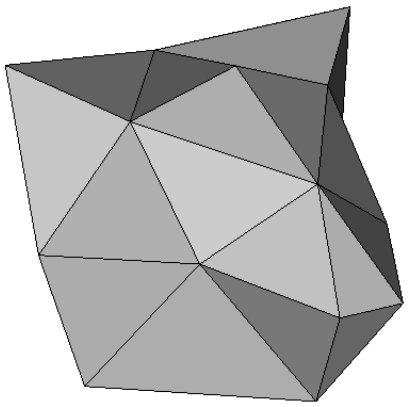
頂点座標を更新した後, エネルギ関数の再計算を行う. λ はエネルギ関数の収束性を表すパラメータとして扱われ, $\lambda = 1.0$ のときに収束性が最も良い. この方法によって得られる制御メッシュは, 初期入力制御メッシュ形状に大きな影響を受けるが, パラメータにより収束性をユーザーが調節でき, またアルゴリズムが高速であるという利点を持つ.

図 3.7 に鈴木らの細分割曲面近似を適用した例を挙げる. ここでは, 虎のモデルを入力メッシュとして, そのメッシュに対しユーザが定義した初期入力制御メッシュをフィッティ

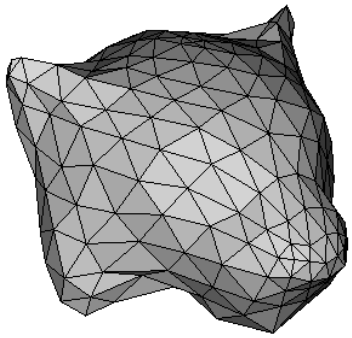
ングさせていく. 初期入力メッシュを図 3.7(a), 入力制御メッシュを図 3.7(b) に示す. また, フィッティング操作の結果出力された制御メッシュを図 3.7(c) に, 細分割近似曲面を図 3.7(d) に示す.



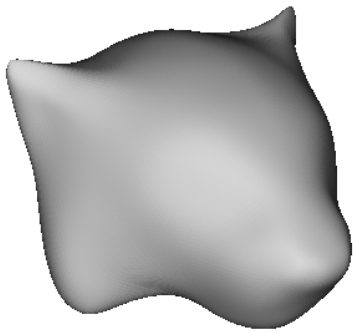
(a) 入力メッシュ



(b) 初期入力制御メッシュ



(c) 出力制御メッシュ



(d) 細分割近似曲面

図 3.7: 細分割曲面のフィッティング処理

3.4 局所的細分割手法

3 角形メッシュに対して規則的な細分割操作を施すと, 細分割レベルに伴いメッシュの面数が増加する. 細分割操作の繰り返しによって全体的に形状の滑らかさは増すが, 曲率の低い

領域にだけ細分割操作を施し、比較的平らな領域のメッシュを粗いままにしておくような細分割操作が提案されている．この拡張された細分割操作を局所的細分割 (local subdivision) 操作と呼ぶ．この局所的細分割操作は, 全体のメッシュの面数を減らしてレンダリングのパフォーマンスを向上させるといった目的で用いられる．また形状ファイル転送の際に, 始めに概形を表示し徐々に精度を上げていく Progressive Transmission [18] などで, モデルの場所によって解像度の異なる部分が必要な場合に用いられている．

3.4.1 面の細分割と T 節点の処理

局所的細分割操作は, 細分割レベルの異なる面同士が隣接する稜線にすきま (crack) を生じさせる．このすきまを形成する点は一般に T 節点 (T-node) と呼ばれる．すきまのないメッシュを生成するには局所的細分割操作において, その T 節点を修正する操作が必要である．この T 節点に対する処理回数を減らすために, 隣接する面の細分割レベルの差が 1 にするような操作が必要となる．

本論文では, *red-green* 3 角形化 (triangulation) [22] と呼ばれる操作を用いている．この方法で, 通常の面の 4-to-1 分割操作は *green* 分割と呼ばれる．また T 節点を修正するための 3 角形を 2 分する操作を *red* 分割と呼ぶ．この *red* 分割は隣接する 1 つの面が細分割された時にのみ用いられる．2 つ, または 3 つの隣接する面が細分割された場合は, その面に対して *green* 分割を施し, さらに必要ならば隣接する面に *red* 分割を施す．また, *red* 分割は, 一時的な分割である．もし *red* 分割された面に対してさらなる細分割操作が必要であるなら, 始めに *red* 分割を施す前の状態の 3 角形に戻し, 元の 3 角形に対して *green* 分割を施すという手順を踏む．

この red-green 3 角形化の様子を図 3.8 に示す．また, 図 3.9 にメッシュに対して, 局所的細分割操作を用いた例を示す．

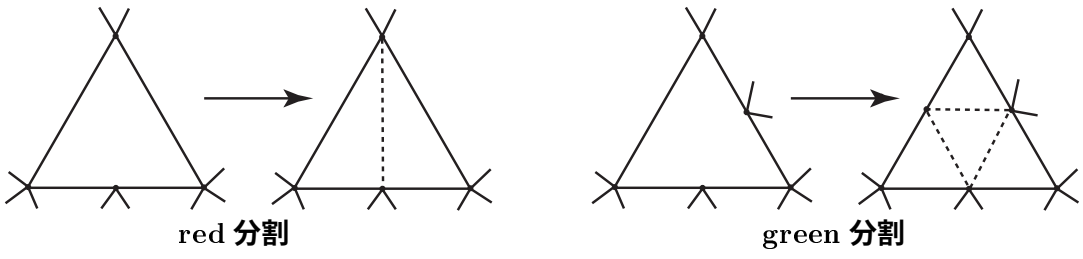


図 3.8: red-green 3 角形化

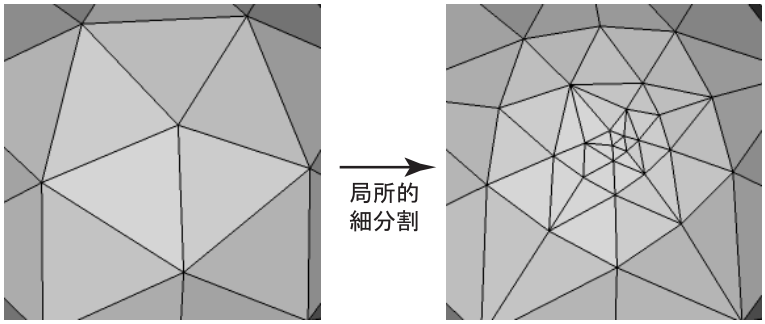


図 3.9: 局所的細分割操作

3.4.2 頂点の再配置

局所的細分割を適用する際に問題となるもう一つの点は, 局所的細分割操作を施したメッシュの新しい頂点座標の算出法である. これは, 細分割レベルの異なる面が隣接している面に対してさらなる細分割操作を必要とする場合に起こる. 細分割の頂点座標の算出は, 3.1章で説明した通り隅頂点の場合は自身の座標と接続する稜線のもう一対の頂点座標との荷重平均, 奇頂点の場合は稜線の両側の頂点座標と面の残りの頂点座標との荷重平均によって算出する. しかし, 局所的細分割を適用し隣接する面との細分割レベルの差がある場合, 必要な頂点座標が異なる細分割レベルのものであるため, 新しい頂点座標が計算できないことがある. そのようなケースを図 3.10に示す.

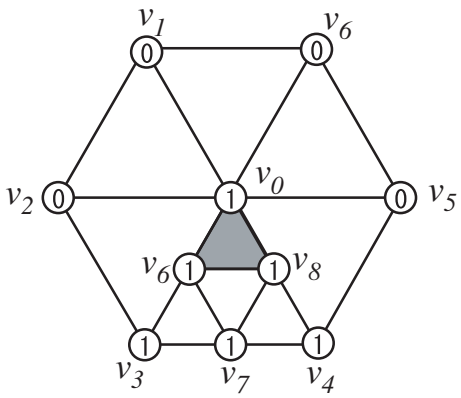


図 3.10: 局所的細分割の頂点再配置問題

この図で, 丸付き数字は頂点の細分割レベルを表しており, 面 $\{v_0, v_3, v_4\}$ を局所的細分割した状態を示している. ここで, 面 $\{v_0, v_6, v_8\}$ をさらに細分割し頂点 v_0 の隅頂点座標を求めることを考える. 頂点 v_0 の隅頂点座標を求めるには, 頂点 v_0 に接続する稜線の反対側の頂点座標を得る必要があるが, 頂点 v_1, v_2, v_5, v_6 との細分割レベルが異なるためそのまま利用できない. 稜線 $\{v_0, v_2\}$ 上の奇頂点を計算するとしても, 頂点 v_0, v_1 と v_2, v_3 の細分割レベルが異なる. そのため, 頂点 v_0 の次の細分割レベルの頂点座標を求めることができない.

正確に次の細分割レベルの隅頂点座標, 奇頂点座標を求めるために, 本操作で用いる局所的細分割では, 隅頂点, 奇頂点の事前計算によりこれを解決する. つまり, ある頂点の次の細分割レベルの座標を計算する際に, その頂点と隣接する同じレベルの頂点座標が得られればよいということである. そのために頂点の再配置を行った後に, その頂点の次の細分割レベルの座標とその頂点に接続する稜線の奇頂点座標を計算する.

この事前計算のために, まず全ての面と頂点に対して細分割操作によって面を分割するとともに細分割レベルを保存しておく. メッシュが入力された時点では, 全ての面, 頂点の細分割レベルは 0 とする. 細分割レベル i の面に対して細分割操作を施した場合, 生成される 4 つの面の細分割レベルは全て $i+1$ とする. また, その際に新しい座標が割り当てられた頂点の細分割レベルも全て $i+1$ となる.

頂点の次の細分割レベルの座標計算は以下のように行う. ある頂点の次の細分割レベルの座標を求める場合, まず頂点に接続する稜線の両端点の細分割レベルを比較する. 稜線の両端の頂点の細分割レベルが同じならば, そのままその頂点座標を採用して隅頂点座標の計算を行い, レベルが異なるならばその頂点との間の稜線に保存されている奇頂点座標を採用する. 図 3.10 で v_0 の次の細分割レベルの座標を求める場合, v_6, v_8 の座標と v_1, v_2, v_5, v_6 との間の稜線の奇頂点座標を用いる. また次のレベルの奇頂点座標を求める場合, 稜線の両端の頂点の細分割レベルと稜線の両側の面の細分割レベルを比較する. レベルが同じならば通常の奇頂点マスクを適用して座標を計算し, 異なるならば必要な稜線に保存されている奇頂点座標を取り出して採用する. 図 3.10 で稜線 v_6, v_0 の次の細分割レベルの奇頂点を求める場合, v_0, v_6, v_8 と v_0, v_2 の間の稜線に保存されている奇頂点座標を用いて計算する. このようにして, 面と頂点に保存された細分割レベルを参照し, 頂点と稜線に保存されている座標を取り出すことで必要な頂点座標を取得することができる.

第 4 章

3 角形メッシュ接続操作

本章では, 本研究の目的である局所的細分割操作に基づく 3 角形メッシュ接続手法について述べる. まず本操作の概要を述べる. 次に, 入力された 2 つの 3 角形メッシュの干渉計算から局所的細分割を行う方法を説明する. そして, そこで利用する干渉計算を効率的に行う方法について述べる. 続いて, 細分割された 3 角形メッシュを接続するアルゴリズムについて解説する. さらに, メッシュの接続形状を元の形状へと近似化するフィッティング操作について説明する.

4.1 接続操作の概要

本接続操作の概要を述べる. 本手法は 2 つの入力 3 角形メッシュ$\mathcal{M}^1$ 、$\mathcal{M}^2$ に対して操作を行う. はじめに、$\mathcal{M}^1$ 、$\mathcal{M}^2$ を 3 次元空間上の接続したい位置に配置する (図 4.1 (a)). 次に、3 角形メッシュ間の干渉計算を行い, 干渉している領域の 3 角形に対して局所的細分割操作を行い, 細分割された 3 角形メッシュ$\mathcal{M}^1_S$、$\mathcal{M}^2_S$ を得る (図 4.1 (b)). さらに, 再度 3 角形メッシュ$\mathcal{M}^1_S$, $\mathcal{M}^2_S$ の干渉計算を行い, 3 角形メッシュの干渉する領域の 3 角形を削除する (図 4.1 (c)). そして, 削除された時にできた境界稜線同士を接続する 3 角形を新たに生成していき, 2 つの 3 角形メッシュを接続した 3 角形メッシュ$\mathcal{M}_C$ を得る (図 4.1(d)). 最後に 3 角形メッシュの元形状への近似化のために 3 角形メッシュの頂点と極限曲面上の極限点とを比較し, フィッティングを行う.

以下の章でそれぞれの手順について述べる。

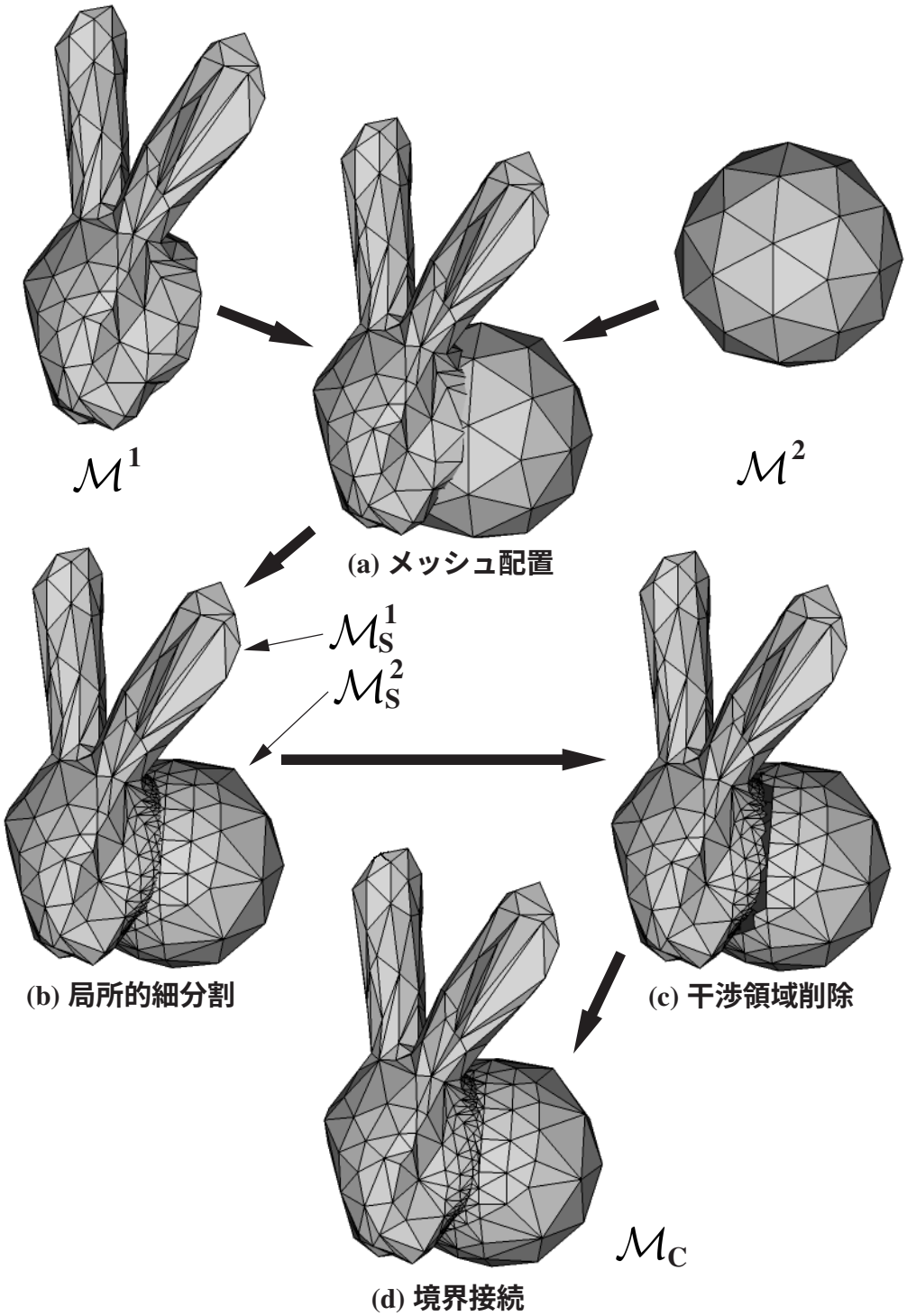


図 4.1: 本接続操作の概要

4.2 干渉領域の局所的細分割と削除

元の制御メッシュの局所的細分割操作は, 本接続手法において重要である. 細分割によって接合部分を局所的に扱うことができ, 操作後の 3 角形メッシュのサイズの減少に役立つ. また後で述べるメッシュの接続操作において, 境界の対応付けを容易にする.

細分割操作は以下の規則に基づいて実行する.

- 他方の曲面形状内に全体、または一部が含まれる 3 角形を削除する.
- 接続部分に近い領域では局所的に干渉形状に近似化する.
- 接続部分から遠い領域では 3 角形メッシュは変化しない.

以上の点を考慮した 3 角形メッシュの局所的細分割と削除処理を以下に示す.

4.2.1 干渉領域の局所的細分割

まず、2 つの制御メッシュ$\mathcal{M}^1$, $\mathcal{M}^2$ 間の干渉チェックを行い, 干渉している面を探す. これらの干渉している 3 角形を 3.4 章で述べた局所的細分割を用いて干渉領域の 3 角形を細分割する. また, 細分割操作の繰り返し回数は干渉領域の両方の 3 角形の面積が近づくようにユーザーが指定する. 細分割の回数を増やすことによって干渉稜線の精度を調節することができる. 局所的細分割操作によって分割した 3 角形に隣接する領域に T 節点を持つ 3 角形ができる. この T 節点は, 前述の *red-green*3 角形化によって 3 角形へと分割する. この処理を通して, 局所的に細分割された 3 角形メッシュ$\mathcal{M}_S^1$ 、$\mathcal{M}_S^2$ を得る.

4.2.2 干渉領域の 3 角形の除去

干渉部分において局所的に細分割した 3 角形メッシュ$\mathcal{M}_S^1$, $\mathcal{M}_S^2$ について, 再度干渉チェックを行う. そして互いに干渉している 3 角形メッシュの 3 角形, または他方の制御メッシュの内部に存在している 3 角形を除去する.

ここで 3 角形を除去するために, 3 角形が他方の形状の内部, または外部にあるかを判定する必要がある. しかし, 全ての 3 角形について内外判定を行うと計算量が増大してしまうので, 本操作ではメッシュを「外側」, 「干渉」, 「内側」という 3 つの領域に分ける領域分け操作を行い, その領域情報にしたがって干渉している面と内側に存在する面を削除する. 領域分け操作は, ある一つの面の内外判定を行い, その面から隣接する面に移動しながらその領域を広げていくというものである.

干渉領域にある面の中で, その隣接面にどの領域にも属していない面が一つでもあればその面の内外判定を行う (図 4.2 (左)). そして, この面を領域拡大の開始面として領域を広げていく. 内外判定は, その時の干渉面と内外判定を行う面の間にある稜線 (e) が干渉している他方のメッシュの面 (f_0^2, f_1^2) の外側にあるか内側にあるかを判定することで行う. 次にその面の隣接面の内, どの領域にも属していない面があれば, その面に自身の領域属性をコピーする. さらにその面の隣接面の領域設定を調べ, 領域属性を割り当てていく. この操作を全ての面に属性が割り当てられるまで再帰的に繰り返す. 図 4.2 (右) に領域拡大の様子を示す. 白色の面は干渉領域の面を表し, 矢印は領域拡大の進行方向を示す.

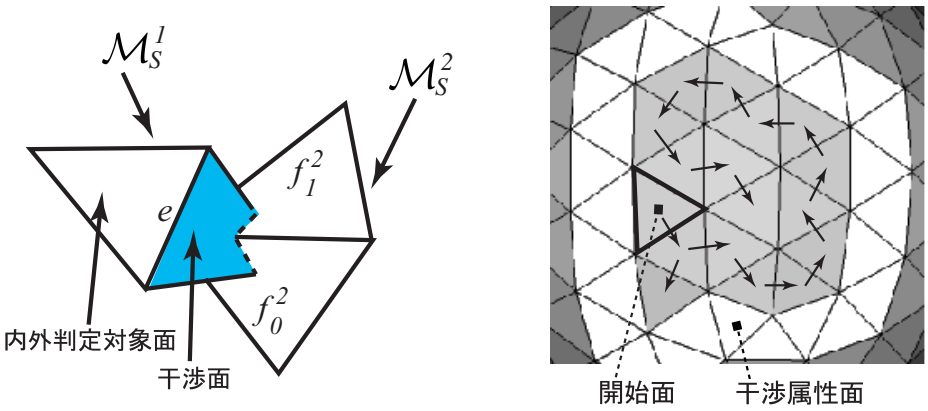


図 4.2: 内外判定と領域の拡大

4.3 干渉計算の効率化

2 つ以上の形状モデル間の干渉検出 (Interference Detection) は CG における基本的な問題である. 干渉検出を高速化することはシミュレーション, リアルタイムグラフィックス, モデリングといった分野で非常に重要な技術である. 複数の形状モデルの干渉を検出する方法は, その形状モデルを包み込むバウンディング・ボックスを生成して干渉しているかどうかのラフチェックを行うことが一般的である. 干渉検出を高速化するには, バウンディング・ボックスの生成方法とバウンディング・ボックス間の干渉計算の効率的に行うことが必要である.

Gottschalk らは, OBBTree(oriented bounding boxes tree) と呼ばれる高速な干渉検出手法を提案した [7]. OBBTree は OBBs(oriented bounding boxes) と呼ばれる 3 次元空間に

おいて任意の向きに配置される直方体のバウンディング・ボックスを用い, 階層的にデータを保持して干渉を計算する方法である.

この OBBTree は, 次のような特徴を持つ.

- 入力されたモデルに密接するバウンディング・ボックスを階層的に用いる.
- 分割軸 (separating axis) と呼ばれる手法を用いて, 2 つの密接なバウンディング・ボックスの干渉を調べる.

3 角形メッシュの OBBs を求める方法を以下に述べる.

OBBs の中心となる平均値 μ と OBBs の向きを表す分散行列 $\mathbf{C}_{jk}$ を求める. $\mathbf{C}_{jk}$ の固有ベクトルはそれぞれ垂直で, 単位ベクトル化することで座標系となる. n を 3 角形の数とし, i 番目の 3 角形の頂点の座標を $\mathbf{p}^i$, $\mathbf{q}^i$, $\mathbf{r}^i$ と表す. また $\mathbf{p}^i$, $\mathbf{q}^i$, $\mathbf{r}^i$ は $\mathbf{p}^i = (p_1^i, p_2^i, p_3^i)^T$ のように 3×1 のベクトルを意味する.

平均値 μ は次の式で計算する.

$$\mu = \frac{1}{24n} \sum_{i=1}^{n} \left(\frac{1}{m^i} \int_0^1 \int_0^{1-t} \mathbf{x}^i \, ds \, dt \right) = \frac{1}{6n} \sum_{i=1}^{n} \frac{1}{m^i} (\mathbf{p}^i + \mathbf{q}^i + \mathbf{r}^i)$$

ここで m^i は i 番目の 3 角形の面積を表し, $m^i = \frac{1}{2}|(\mathbf{q}^i - \mathbf{p}^i) \times (\mathbf{r}^i - \mathbf{p}^i)|$ で計算される. $\mathbf{x}^i$ は次の式のように i 番目の 3 角形を s, t を用いてパラメータ化したものである.

$$\mathbf{x}^i = \mathbf{p}^i + s(\mathbf{q}^i - \mathbf{p}^i) + t(\mathbf{r}^i - \mathbf{p}^i), \qquad s, t \in [0, 1]$$

また分散行列 $\mathbf{C}_{jk}$ は次の式で計算する.

$$\begin{aligned} \mathbf{C}_{jk} \quad = \quad & \frac{1}{24n} \sum_{i=1}^{n} m^i [(\bar{\mathbf{p}}_j^i + \bar{\mathbf{q}}_j^i + \bar{\mathbf{r}}_j^i)(\bar{\mathbf{p}}_k^i + \bar{\mathbf{q}}_k^i + \bar{\mathbf{r}}_k^i) \\ & + \bar{\mathbf{p}}_j^i \bar{\mathbf{p}}_k^i + \bar{\mathbf{q}}_j^i \bar{\mathbf{q}}_k^i + \bar{\mathbf{r}}_j^i \bar{\mathbf{p}}_k^i], \qquad 1 \le j, k \le 3 \end{aligned}$$

ここで, $\bar{\mathbf{p}}^i$, $\bar{\mathbf{q}}^i$, $\bar{\mathbf{r}}^i$ はそれぞれ $\bar{\mathbf{p}}^i = \mathbf{p}^i - \mu$, $\bar{\mathbf{q}}^i = \mathbf{q}^i - \mu$, $\bar{\mathbf{r}}^i = \mathbf{r}^i - \mu$ である. OBBs 生成の例として, 2 次元稜線列に対する OBBs 生成の様子を図 4.3 に示す. OBBs は密接したバウンディング・ボックスを最も長い軸に垂直な面で再帰的に分割していく.

次に軸投影による OBBs の干渉計算方法について述べる. 軸投影による干渉チェックは, バウンディング・ボックスを任意の軸に投影する方法で, 投影された軸の交わりから干渉を判定するものである. OBBs の中心を任意軸へ投影し, 投影されたボックスの半径を計算する. もし 2 つの OBBs の中心間の距離が 2 つの半径の和よりも大きければ, 2 つの OBBs は干渉しない.

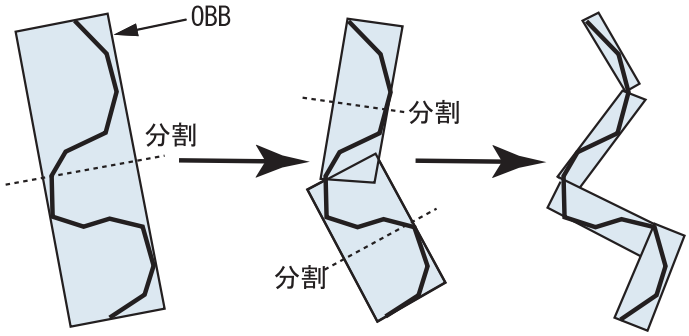


図 4.3: OBBs の構築と再帰的な分割

1 回の OBBs の干渉計算は一般的なバウンディング・ボックス同士の計算よりも非常に高速である. 何十万ポリゴンというモデル同士の干渉計算もインタラクティブな速度で実行可能である. この OBBTree による干渉計算は, 他の sphere tree [13] や AABBs(axis-aligned boundiing boxes) [30] と呼ばれる方法よりも高速で正確である.

本論文で提案する接続手法ではこの OBBTree を用いて, 3 角形面の干渉ペアを得る. 本操作では, 局所的細分割操作により細分割された面について OBBs を再構築し干渉を計算していく. 干渉していない OBBs は分割を行わず, 干渉領域の OBBs だけを細かく分割する. これにより OBBs は階層的に表現され, メッシュ形状の干渉計算を効率的に行うことができる.

4.4 境界接続アルゴリズム

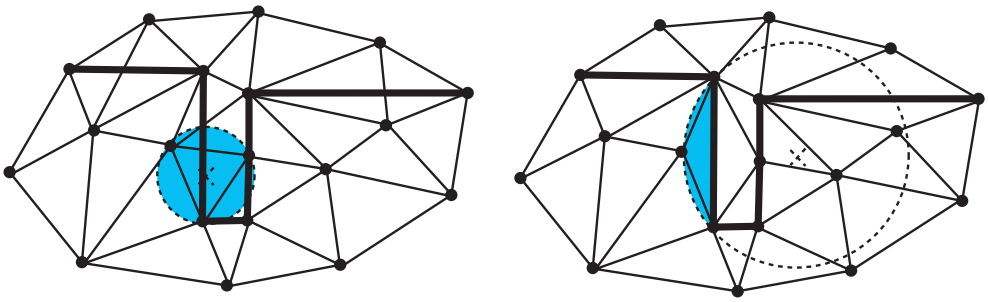
局所的細分割を適用し, 干渉面と相手内部に含まれる面を削除した 3 角形メッシュにできた境界を接続するアルゴリズムについて述べる.

本操作は, 2 つの 3 角形メッシュの境界上の頂点同士を結ぶ稜線を生成していくことで 3 角形メッシュを同士を接続するが, 稜線が交差したり位相的に正しくない稜線を生成しないようにしなければならない. そのために, 3 角形メッシュの境界を接続する 3 角形を生成することに注意を払う. 境界を接続する 3 角形を生成するには, 一方のメッシュの境界稜線の両端の 2 つの頂点と, もう一方の境界上の 1 つの頂点の組み合わせのうち最適なものを探索する方法が必要である. そのような 3 角形生成手法に拘束付ドロネー 3 角形化 (Constrained Delaunay Triangulation, 以下 CDT) [1] 手法がある. CDT は, 不規則な頂点と固定稜線として定義されたセグメントの入力から 3 角形集合を生成する手法である. ドロネー 3 角形

とは点集合を p_m としある 3 角形を p_i, p_j, p_k としたとき, その 3 角形の外接円内に p_i, p_j, p_k 以外の全ての頂点 p_m を含まない 3 角形である. このことを以下の式のように表すことができる.

$$|x - p_i| = |x - p_j| = |x - p_k| \quad \wedge \quad \forall m \neq i, j, k \quad |x - p_i| < |x - p_m|$$

ここで x は 3 角形 p_i, p_j, p_k の外心である. 2 次元において CDT により生成された 3 角形は, 入力したセグメントを含む 3 角形以外は次の式を満たす. セグメントを含む 3 角形を生成する場合, その 3 角形の外接円をセグメントにより分割し, 3 角形の存在する側の円弧とセグメントによってできる領域に他の頂点がなければよいという条件を用いている. この CDT の適用例を図 4.4 に表す. 図中の太線は入力セグメントを表す. 波線の円は 3 角形の外接円を表し, 色の塗ってある領域は頂点を含むかどうか考慮する範囲である. (a) は通常のドロネー 3 角形化 (Delaunay Triangulation), (b) は CDT を適用している.



(a) ドロネー 3 角形化　　(b) 拘束付ドロネー 3 角形化

図 4.4: 拘束付ドロネー 3 角形化

CDT は 2 次元では解決済みの問題であるが, 3 次元では多数の解決されるべき問題が残っている [25]. そのため本接続操作では, 2 次元の CDT のアルゴリズムを応用したものを用いる. CDT の実装は次の 2 つの方法を用いることが多い. 1 つは稜線交換 (edge flip) による方法で, 3 角形が定義する外接円内に他の頂点が存在した場合, その頂点を端点とする稜線を生成して 3 角形を再構築する方法である. しかし, 3 次元のメッシュにおいてこの方法を実装することは非常に複雑であり, 多数の位相チェックが必要である. もう 1 つは, 3 角形の外接球内に他の頂点が含まれない組み合わせを見つけ, 逐一 3 角形を生成していく方法である. この方法は前者に比べ剛健であり, 実装も比較的容易である.

これらを踏まえて本接続操作で用いる 3 角形生成アルゴリズムは以下の条件を満たす固定稜線と頂点の組み合わせを見つけることである. ここで 3 角形化の基準となる稜線を

$E\{p_i, p_k\}$, 稜線 E の片側の 3 角形を $F_1\{p_i, p_j, p_k\}$ とする. また, 面 F_1 の法線ベクトル $\mathbf{n}_a$ に平行で, 稜線 E を含み面 F_1 側を裏とする平面 P を定義し, その平面 P の法線ベクトルを $\mathbf{n}_p$ とする. これを図 4.5 に示す. このとき,

- 角度 $\angle p_i, p_j, p_k > \angle p_i, p_m, p_k \quad (\forall m \neq i, j, k, l)$ である.
- 平面 F_1 の表側に存在する. すなわち $\mathbf{n}_p \cdot (\mathbf{p}_l - \mathbf{p}_i) > 0.0$ である.

を満たす p_i, p_k, p_l で 3 角形を作る.

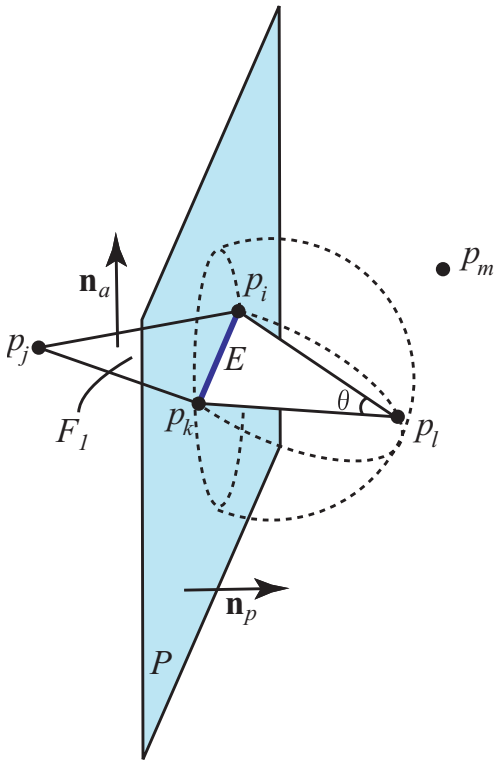


図 4.5: 最適な 3 角形の探索

まず任意の境界稜線を選択し, 稜線の両端の頂点とのベクトルが成す角度が最大になる頂点を見つけ 3 角形を生成する. その 3 角形の生成には次の 3 つのパターンが存在する.

- 1 つの境界稜線と 1 つの境界上の頂点の組み合わせで新しく 2 本の稜線を生成する
- 2 つの境界稜線の組み合わせで新しく 1 本の稜線を生成する

- 3 つの境界稜線の組み合わせで新たに稜線は生成しない

これらの操作で新しく生成された稜線を基準として 3 角形生成を再帰的に繰り返し, 接続すべき境界稜線が無くなった時点で処理を終了する. 図 4.6 に境界を持つ 3 角形メッシュに対して接続操作を適用した例を示す.

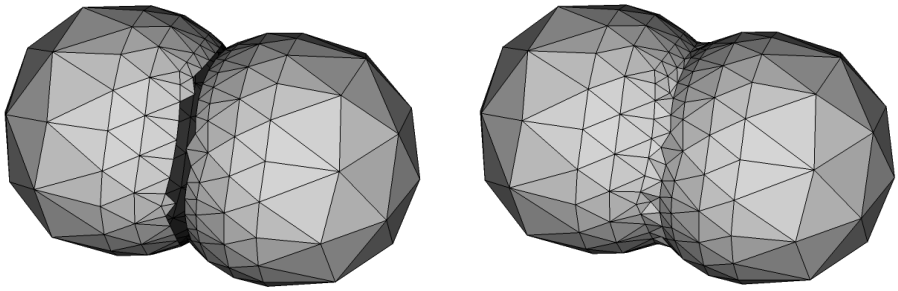


図 4.6: メッシュ境界接続

4.5 メッシュのフィッティング操作

制御メッシュの細分割と接続によって生成されたメッシュを元のメッシュが定義する細分割曲面にフィットするよう制御点の座標を修正する方法について述べる. 本操作によって接続されたメッシュ形状は, 境界付近で接続性が変化する頂点を持ち, そのためメッシュ上に凹凸が現れるケースがある. そのような箇所を修正するために, このメッシュのフィッティング操作を用いる.

接続した 3 角形メッシュの頂点は次の 2 つに分類できる. 1 つは接続領域から遠い領域に存在する頂点であり, 元の 3 角形メッシュと同じ頂点座標と隣接関係を持っている. この頂点については元の制御メッシュと同じ極限点を持つので制御点の座標を修正する必要はない. もう 1 つは接続部分から近い領域に存在する頂点であり, メッシュの面の削除と境界接続操作により頂点座標、隣接関係がともに変化した制御点である. この頂点については, 制御メッシュの詳細化により頂点座標と隣接関係が元の制御メッシュから変化しているので, 元の細分割曲面上に対応する極限点を求める必要がある.

本操作では, メッシュの面の削除の際に影響をうける頂点に対して細分割極限点を求めておき, 接続したメッシュの頂点が定義する極限点とのずれを修正する方法を用いる. 制御メッシュのフィッティングには前述した鈴木ら [28] の手法を用いる. しかし, 鈴木らの手法

では全ての頂点の座標を修正しているが, 本手法では上述のとおり座標, 隣接関係が変化した頂点についてのみフィッティング操作を行う. 具体的には, 局所的に細分割された 3 角形を構成する頂点とそれらの頂点に隣接している頂点をフィッティングの対象とする. 図 4.7 に対象となる頂点を図示する.

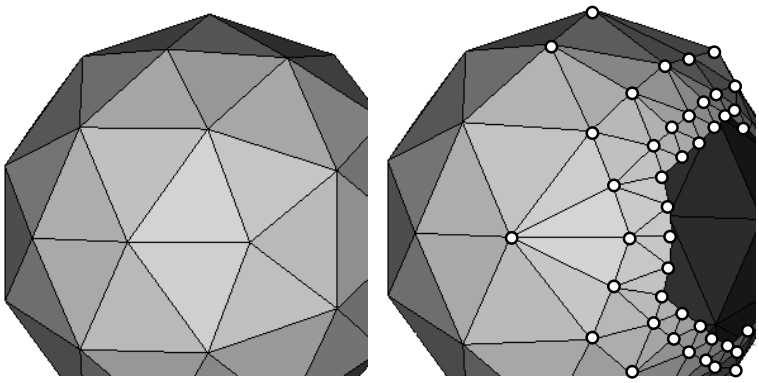


図 4.7: フィッティングの対象となる頂点

図 4.7 で左側は元の制御メッシュ, 右側は局所的に詳細化された制御メッシュである. 白丸の印が付いている制御点がフィッティングの対象となる. なお、接続の対象となる他方の制御メッシュは表示していない. このフィッティング操作を行うことで、入力細分割曲面を近似した接続形状を得ることができる.

また, 本接続操作で接続したメッシュに対してフィッティングを適用した例を図 4.8 に示す. エネルギ関数の収束パラメータ $\lambda = 0.5$, 許容誤差 $\varepsilon = 1.0 \times 10^{-4}$ で操作を適用している. フィッティング操作により, 境界接続領域のメッシュの振動が補正されているのが分かる.

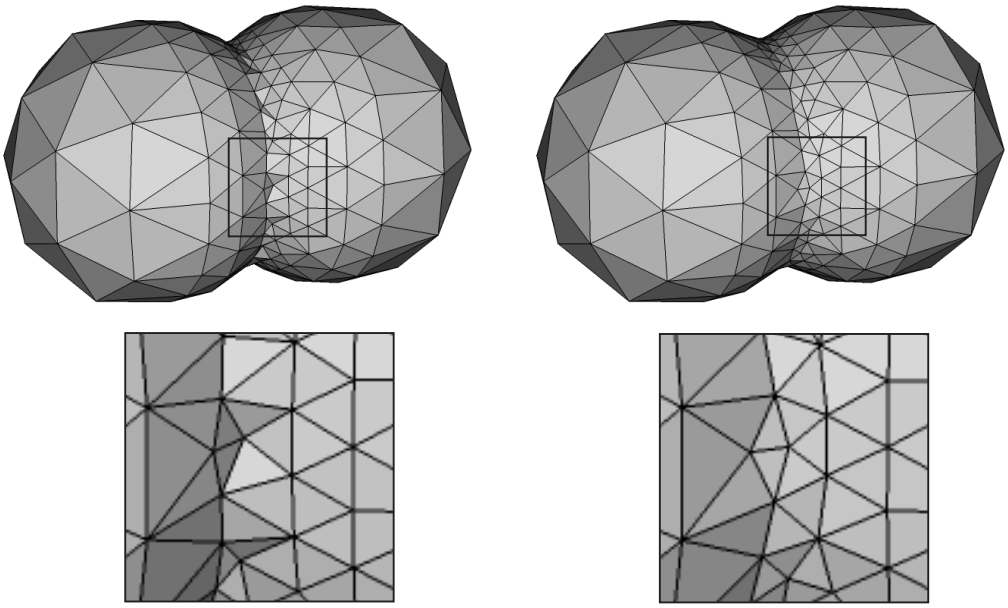


図 4.8: フィッティング操作適用例

第 5 章

システムへのコマンド実装

本章では, 本論文が提案するメッシュ接続操作の実装について述べる. 始めに本操作を実装したはにわモデラーについて述べ, 次にはにわモデラーのモデリング機能について説明する. 続いて, 本接続操作の実装と実行例を示す. 最後に, はにわモデラーで作成したデータをインターネット上で閲覧するためのソフトウェアである XVL Viewer について説明する.

5.1 はにわモデラーとは

3 次元 CG 技術はエンターテイメント, 設計, 医療などのさまざまな分野で利用され, その有用性が広く認識されている技術である. 一方で, 3 次元 CG の学習環境に目を向けた場合, インターネットの普及に伴い多くの CG 学習環境がインターネット上に構築されるようになってきた. しかし, これらの CG 学習環境の多くは紙の教科書に含まれる文章をコンピュータ上で電子化しただけのものであり, コンピュータやインターネットを利用したメリットが十分に活かせていないのが現状である.

3 次元 CG の学習をより効果的に行うためには, 理論を説明するための文章に加え, 学習者がその効果を体験できるようにするための手段を提供しておくことが不可欠である. この時, CG に関する技術は形状を作成するための技術と形状を表示するための技術に分けられることを考えると, その両方を体験できるようになっていることが望ましい.

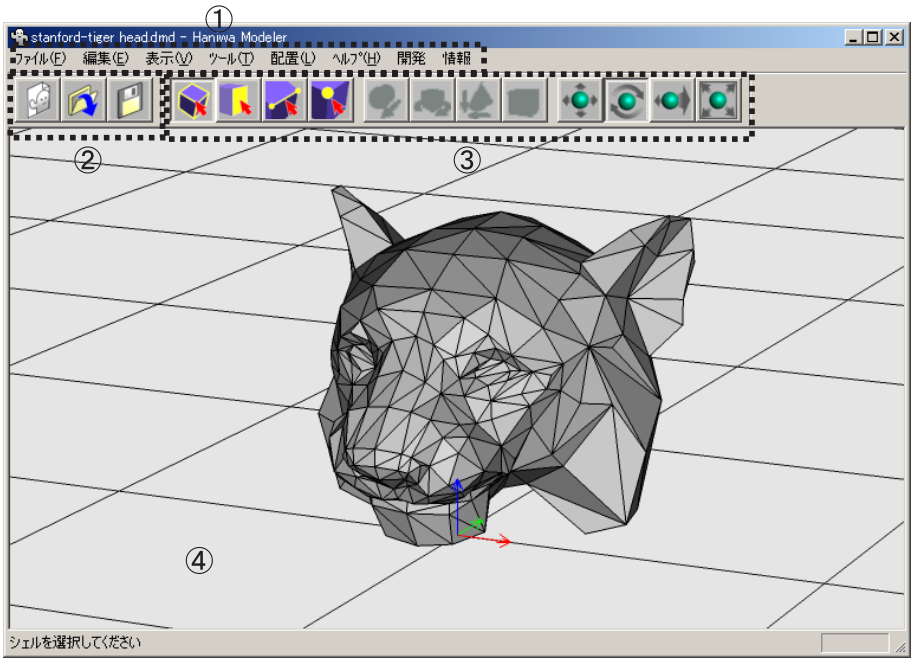
そこで教育利用目的のための 3 次元形状モデラーである「はにわモデラー」, および得られる形状を効率的に表示するための「はにわレンダラー」を開発している.

これらを利用することにより, より魅力的な CG 学習環境を提供することができるようになる. 学習者は教科書で学習した内容をコンピュータ上で実際に体験することができるようになり, CG 技術に関するより深い理解を得ることが期待できる.

はにわモデラーの動作環境には, パーソナルコンピュータ上で動作し, 高度な機能と信頼性を実現している Microsoft 社の「Windows NT 4.0」を採用している. 開発環境には, Windows 上で動作するアプリケーションの作成を補助するためのクラスライブラリである MFC(Microsoft Foundation Class) を含み, Windows 上におけるソフトウェア開発において広く利用されている Microsoft 社の「Visual C++ 6.0」を利用している. 本モデリングシステムは拡張性に優れたオブジェクト構造を採るため, 機能修正, 機能追加, 機能削除を容易に行うことができる. また, 3 次元形状管理システムとしては, ラティス・テクノロジー株式会社が提供している「Lattice Kernel Version 1」 [34] を利用している. なお, はにわモデラーを含むシステムは, 情報処理振興協会の協力のもとリコー社, ラティス・テクノロジー社の各社とのコラボレーションを行うことにより制作された.

本接続操作は, このはにわモデラーのコマンドとして実装している. 他のモデリングコ

マンドと併用して用いることで, 一連の高度なモデリング環境を提供している. 図 5.1 にはにわモデラーの実行画面を示す. 図中の丸数字 1 はメニューバーであり, ここから様々なコマンドを実行する. また丸数字 2 は標準ツールバー, 丸数字 3 は操作ツールバーでありコマンドの中でも使用頻度の高いものをボタンとして持つ. 丸数字 4 は作業スペースで, 形状表示, 要素選択などを行う領域である.



① メニューバー
② 標準ツールバー
③ 操作ツールバー
④ 作業スペース

図 5.1: はにわモデラーの実行画面

5.2 はにわモデラーのモデリング機能

はにわモデラーのモデリング機能について説明する. はにわモデラーのモデリング手順は, 始めにマウスのピックにより形状要素を選択し, その要素に対して編集操作をメニュー, またはツールバーから選択し実行するというように行う. 形状要素にはシェル, 面, 稜線, 頂点の 4 つを持つ. 以下に, はにわモデラーのモデリング機能と実行例を示す.

基本形状生成

正 n 角柱を生成する機能である. 図 5.2 に基本形状生成例を示す.

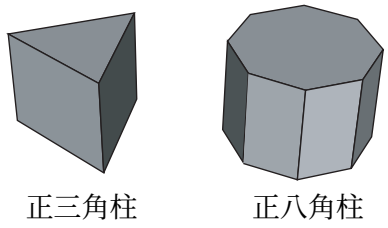


図 5.2: 基本形状生成

幾何変形操作

各要素の移動, 回転, 拡大縮小を行う. 移動量, 回転量, 拡大縮小率はマウスのドラッグ操作によって決定する. 図 5.3 に幾何変形操作の例を示す.

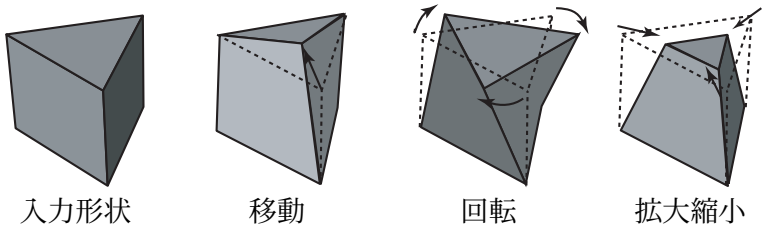


図 5.3: 幾何変形操作

位相操作

2 つの頂点を結ぶ頂点連結, その逆の操作の稜線削除を行う. また, 稜線分割, その逆の操作の頂点削除を行う. さらに 3 つのパターンの面分割, 面の持ち上げ操作を持つ. 図 5.4, 図 5.5, 図 5.6 にそれぞれの操作の実行例を示す.

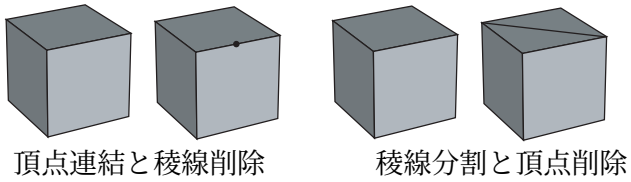


図 5.4: 分割操作と削除操作

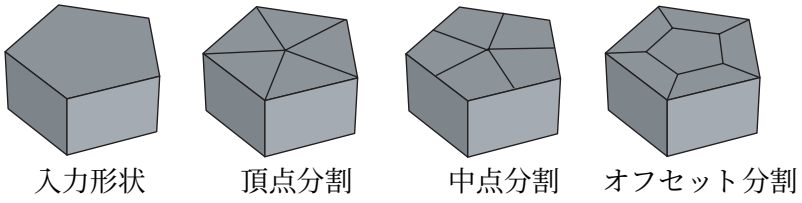


図 5.5: 面分割操作

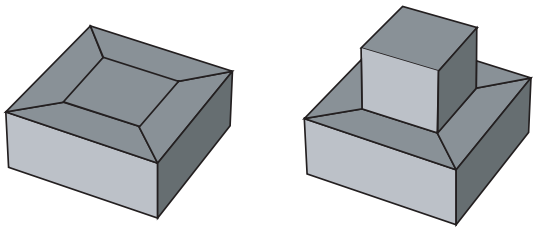


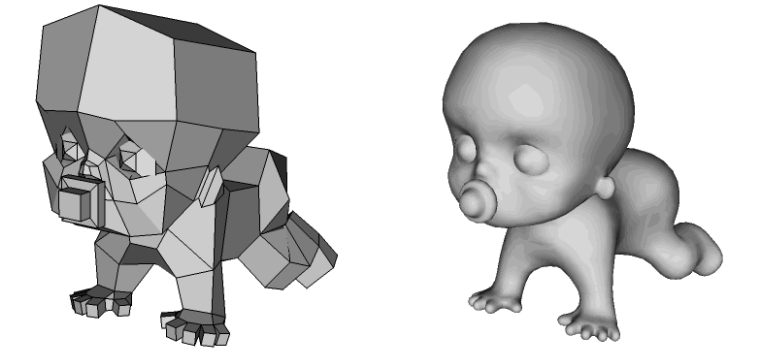
図 5.6: 面持ち上げ操作

丸め操作

丸め操作とは, 多角形面によって構成されるシェルを曲面化する機能である. 図 5.7 に丸め操作実行例を示す.

5.3 3 角形メッシュ接続コマンド

本論文で提案する接続操作は, はにわモデラーの 1 つのコマンドとして実装している. ユーザーにより自由に 3 次元空間内に配置された 2 つのシェルを入力メッシュとして操作を



入力形状　　丸め形状

図 5.7: 丸め操作

実行する．操作オプションによりそれぞれのシェルに対する局所的細分割の回数, メッシュのフィッティングの許容誤差, 収束率を指定できる．デフォルト値は, 細分割回数がそれぞれ 2 回, 許容誤差 ε が 0.0001, 収束率 λ を 0.5 としている．図 5.8 に接続操作パラメータのダイアログ示す.

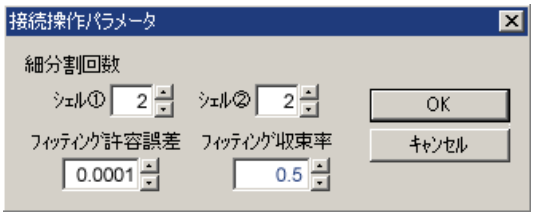


図 5.8: 接続操作パラメータのダイアログ

5.4 XVL Viewer

はにわモデラーは, 作成した形状モデルを XVL(eXtensible VRML with Lattice) と呼ばれるフォーマットで出力することができる．この XVL フォーマットは, 3 次元曲面形状を非常に小さいデータサイズで表現し, ラティス・テクノロジー株式会社が提供する XVL Viewer というブラウザ上で動作する 3 次元モデルビューアで閲覧することができる．この XVL Viewer を利用することで, はにわモデラーを用いて作成した 3 次元形状モデルをイン

ターネット上に配信することが可能である．図 5.9 に，XVL Viewer の実行画面を示す．

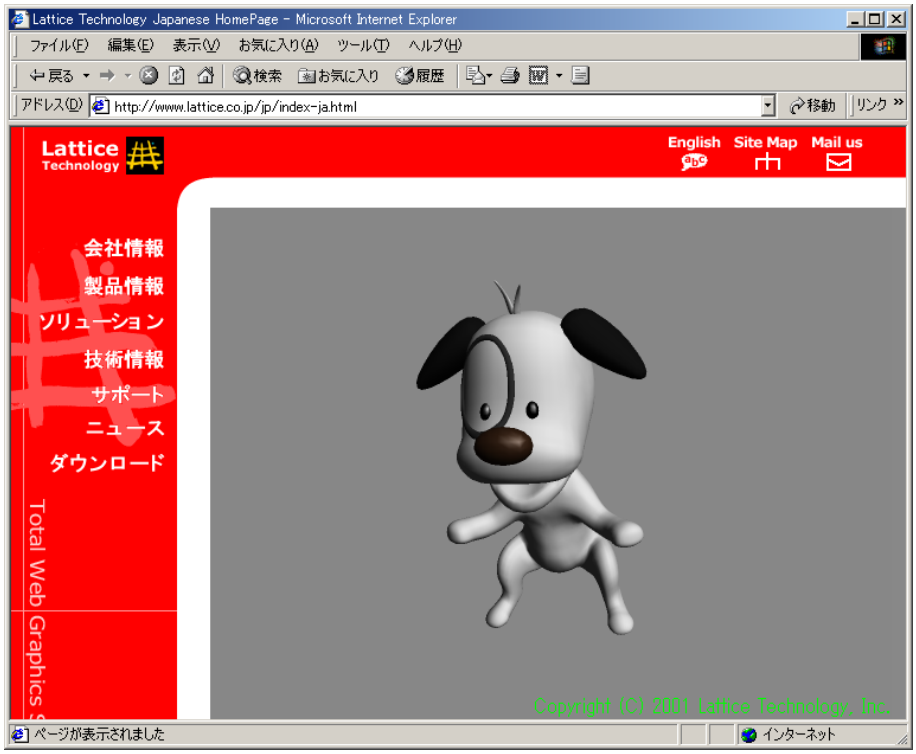


図 5.9: XVL Viewer による 3 次元形状モデルの表示

第 6 章

結果と考察

本章では, まず本論文が提案する接続操作をサンプルメッシュデータに適用した実行例を示す. 次にそれぞれの実行例を評価し考察を行う.

6.1 実行結果

本節では, 本論文で提案した接続操作をメッシュに対して適用した実行例を示す. 実行例として, 一方のメッシュの一部を切り取り他方のメッシュへ貼り付けた実行例, 細分割回数を変えて接続した実行例, *sharp edge* 属性を持ったメッシュの接続操作の実行例を示す.

兎の背中へのアヒルの羽の接続

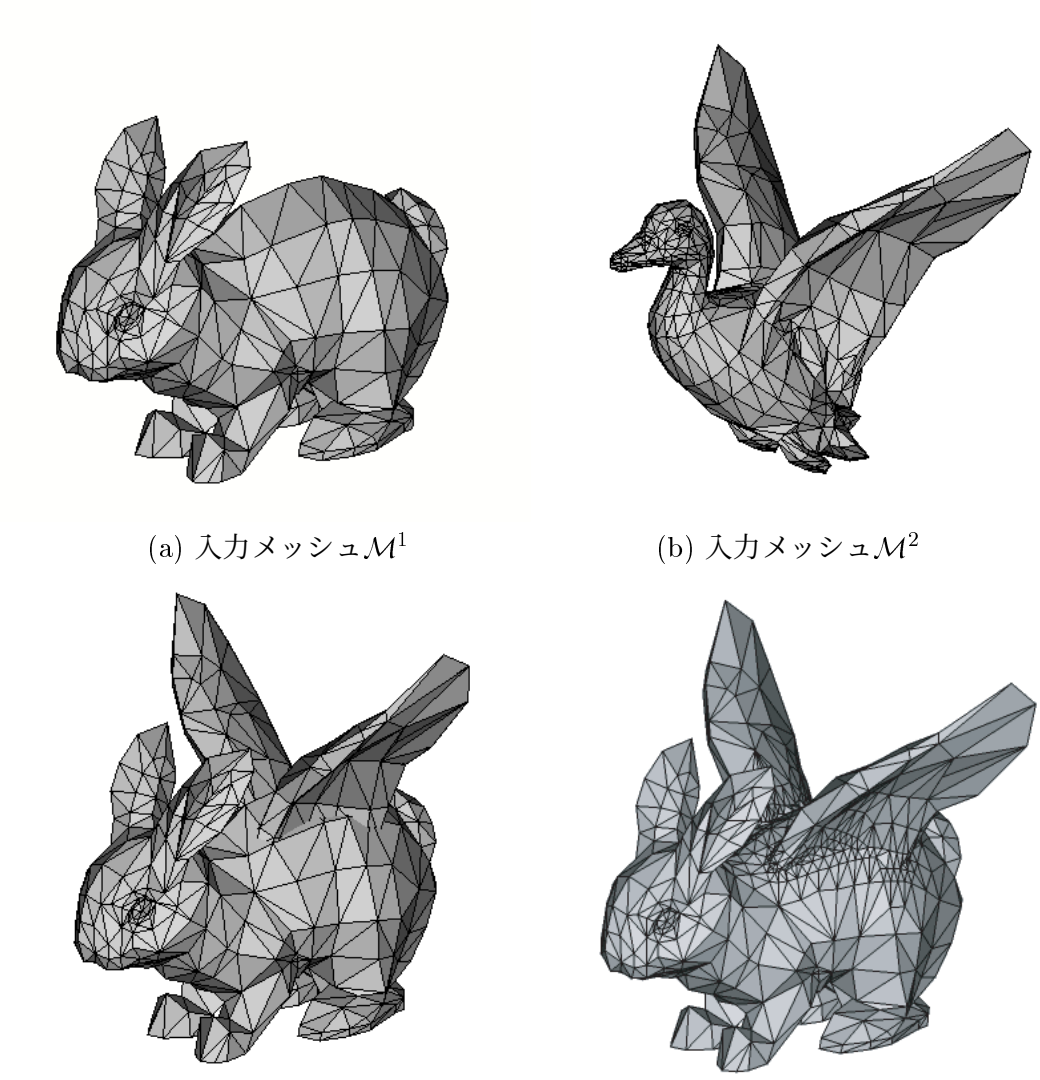
図 6.1 に兎の背中にアヒルの羽を接続した実行例を示す. アヒルの羽は事前にアヒルのメッシュから切り取り, 兎の背中へ接続操作を行っている. 細分割回数は兎, アヒルの羽それぞれ 2 回を指定し, フィッティング許容誤差に 1.0×10^{-4}, 収束率に 0.5 を指定した. 本接続手法は一方のメッシュの一部を切り取り, 他方のメッシュに貼り付けるという操作に有効であることが分かる.

細分割回数を変えて接続

図 6.2 に, 単純な球のメッシュを局所的細分割の回数を変えて接続した例を示す. ここでは, 分かりやすさのために単純な球のメッシュを用いた. 局所的細分割の回数を上げるにしたがって, 接続境界付近のメッシュが細かく分割され, 2 つの入力形状の干渉稜線がより正確に表現されていくのが分かる.

角を持つメッシュの接続

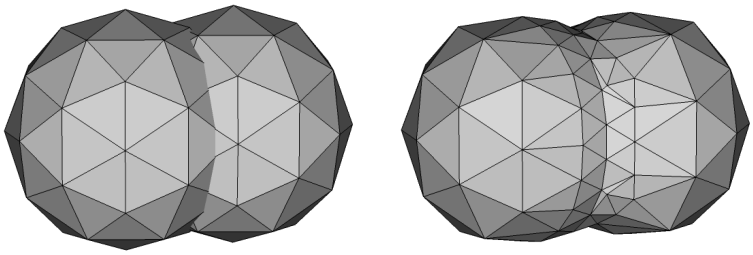
図 6.3 にトライセラトプスの背中に大砲を接続した実行例を示す. 大砲の土台部分は事前に切り取って接続操作を行っている. また大砲には *sharp edge* 属性を付加してある. 細分割回数はトライセラトプス, 大砲それぞれ 2 回を指定し, フィッティング許容誤差に 1.0×10^{-4}, 収束率に 0.5 を指定した. 大砲の支え部分の尖った形状特徴が接続形状にも残っているのが分かる.



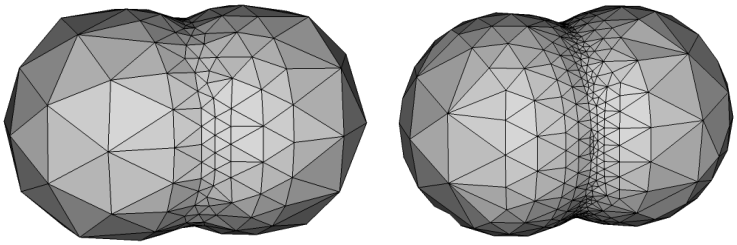
(a) 入力メッシュ$\mathcal{M}^1$　(b) 入力メッシュ$\mathcal{M}^2$

(c) メッシュの配置　(d) 接続メッシュ$\mathcal{M}_C$

図 6.1: 兎とアヒルの羽の接続メッシュ



(a) メッシュの配置　　(b) 局所的細分割回数 1

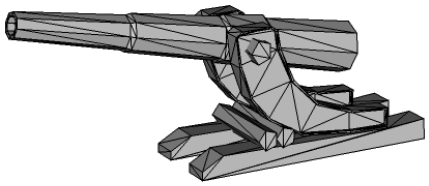


(c) 局所的細分割回数 2　　(b) 局所的細分割回数 3

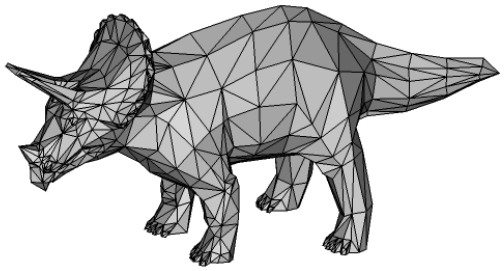
図 6.2: 異なる細分割回数指定による接続メッシュ

グリフォン

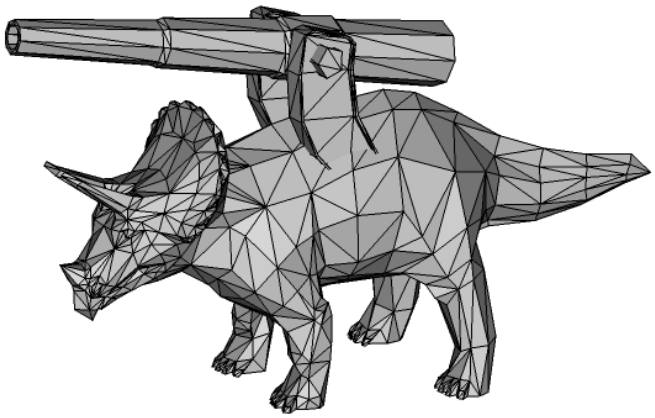
図 6.4 に, 鷲と虎のメッシュを接続してグリフォンを生成した実行例を示す. 鷲のメッシュから頭と羽を切り取り, 虎のメッシュと接続操作を行った. 細分割回数は鷲の頭と羽は 1 回, 虎は 2 回とし, フィッティング許容誤差に 1.0×10^{-4}, 収束率に 0.5 を指定した. ユーザーがそれぞれのメッシュに対して適用する細分割回数を自由に指定できるので, メッシュの精度が異なるデータ間でも意図したとおりに本接続操作を適用できる.



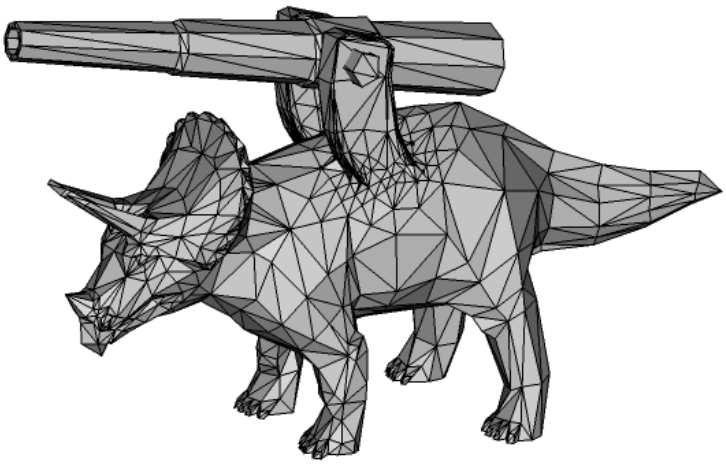
(a) 入力メッシュ$\mathcal{M}^1$



(b) 入力メッシュ$\mathcal{M}^2$

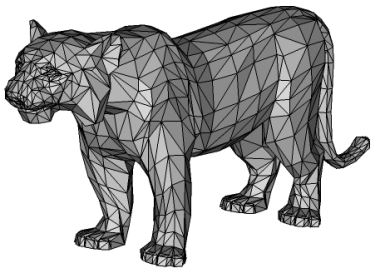


(c) メッシュの配置

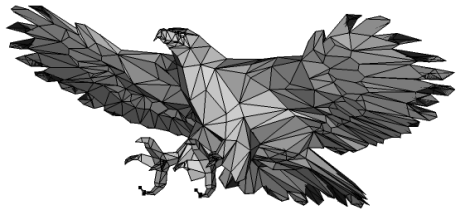


(d) 接続メッシュ$\mathcal{M}_C$

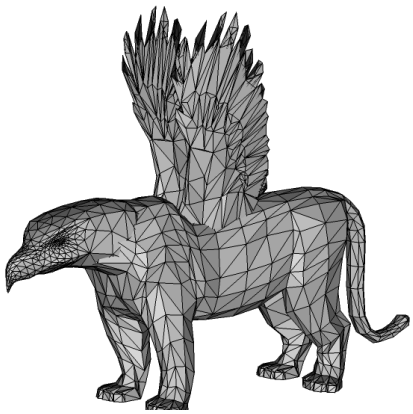
図 6.3: トライセラトプスと大砲の接続メッシュ



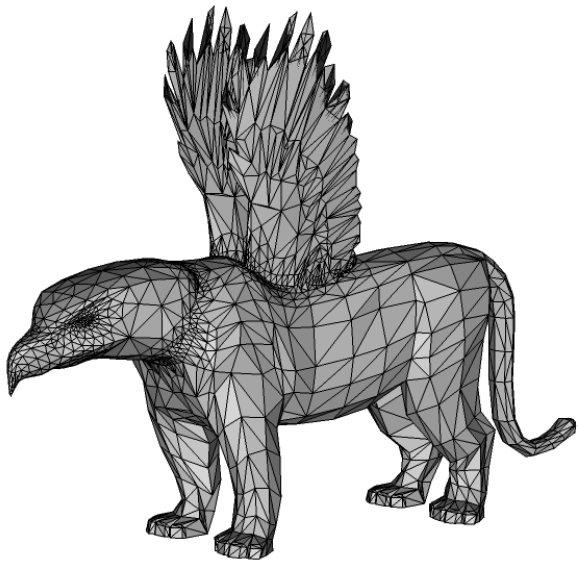
(a) 入力メッシュ$\mathcal{M}^1$



(b) 入力メッシュ$\mathcal{M}^2$



(c) メッシュの配置



(d) 接続メッシュ$\mathcal{M}_C$

図 6.4: 鷲と虎の接続メッシュ

6.2 考察

本章では, 本接続操作を実行速度, メッシュ数によって評価する. 実行速度の計測には, PentiumIII-800MHz, メモリ 256MB, グラフィック・ボードに GeForce GTS を搭載した PC-AT 互換機を用いた. 実行例で用いたデータはインターネットのサイトから入手したフリーデータである. そのため, 本手法の一般性, 実用性を評価するのに適したデータと言える. 表 2 には, 本実行例において,「局所的細分割と面の削除」,「境界接続」,「フィッティング操作」の各ステップに費やした時間を表示している. ただし, 図 6.2 については, 3 回の局所的細分割操作を適用したメッシュに関する実行時間を示している. また, 図 6.4 については, 鷲の頭と虎の胴体を (1), 鷲の羽と虎の胴体を (2) として実行にかかった時間を示している.

	局所的細分割と面の削除	境界接続	フィッティング	合計時間
図 6.1	0.495	0.120	0.081	0.696
図 6.2	0.397	0.132	0.076	0.605
図 6.3	0.427	0.130	0.080	0.637
図 6.4 (1)	0.632	0.342	0.180	0.774
図 6.4 (2)	0.824	0.499	0.220	1.443

表 2: 各実行例における実行時間 (秒)

各ステップごとの実行時間を比べると,「境界接続」,「フィッティング操作」にかかった時間に比べ,「局所的細分割と面の削除」が全実行時間の大半を占めていることが分かる. また「局所的細分割と面の削除」ステップで処理時間を多く必要とするのは, 局所変形操作を多数回適用していることと, その局所変形操作前にメッシュの位相の整合性を保つための複雑な位相チェックを行っていることが挙げられる. この結果から考察されることは, 本手法は境界を接続しフィッティング操作を行うことに関しては高速な処理が可能だということである. 局所的細分割と面の削除方法を改良すれば処理全体に費やす時間が短縮され, 高速な接続操作が実現されるはずである.

また表 3 は, 本実行例で示したそれぞれのメッシュの面 f と頂点 v の数を示している.

表 2 を見ても分かるように, 接続されたメッシュの面数は入力した 2 つのメッシュの面数と比較して大幅な増加は見られない. これは本手法が入力メッシュの干渉領域を局所的に細分割し, その境界を接続して接続メッシュを得るといった方法を採用していることによる. 本操作は, 接続操作によるメッシュの変形を局所化し, メッシュの面数の増加を抑えるという目的を実現しており, 本操作の有用性を確認できた.

	$\mathcal{M}^1$		$\mathcal{M}^2$		$\mathcal{M}_S^1$		$\mathcal{M}_S^2$		$\mathcal{M}_c$	
	f	v	f	v	f	v	f	v	f	v
図 6.1	902	453	418	213	1346	675	1259	699	1760	882
図 6.2	80	42	80	42	880	442	874	438	1052	532
図 6.3	135	72	1370	687	1246	632	1566	785	2270	1174
図 6.4	1850	925	2076	1040	2338	1313	2901	1512	4165	2086

表 3: 各実行例における頂点と面の数

本操作の利点として, ユーザーはメッシュを貼り付けたい位置に配置し, 本操作を適用するだけで接続形状を得られるという操作の簡易さが挙げられる. 本操作は, 操作の取り消し, やり直しといった履歴操作の対象となっており, メッシュの接続位置を変えたいのであれば, 操作を取り消してメッシュの位置や向きを変え, 接続操作を再実行する, といったことがストレスなく行うことができる. また, それぞれの入力メッシュに対して細分割回数を自由に指定できるので, 入力メッシュの細かさやサイズの異なるメッシュ同士でも本接続操作を適用することができる. これは, インターネット上にある様々なメッシュデータを利用可能であることを表し, 本操作の実用性を高めている. さらに, 図 6.3 の例のように, 尖った部分を持つメッシュの特徴を保存して, 接続形状を生成することができる点も本操作の大きな利点の一つである. これは, 従来の接続操作では実現されていない利点であり, 本操作の有用性を表している.

第 7 章

結論と展望

本論文では, 新しい形状モデリング手法である局所的細分割に基づいた任意の 3 角形メッシュの接続操作を示した. 従来のメッシュ融合演算手法では, 不要な部分のメッシュ数が増加すること, ユーザーによるメッシュの境界の指定が必要であることが問題であった. 本操作は局所的細分割操作によって, 接続操作による変形部分を局所化することでこれらの問題を解決した. ユーザーによるメッシュを貼り付ける境界の指定は不要であり, メッシュを配置する操作だけで接続形状を生成できる.

本手法では, メッシュの干渉部分の局所的細分割を行い, 生成されたメッシュの境界を接続することで接続形状を得る. そのため, 干渉領域のメッシュだけが細かい 3 角形で表現され, その他の部分ではメッシュ形状は入力形状から変化しない. さらに, メッシュのフィッティング操作を行うことで接続形状を入力形状に近似する. そのため, 視覚的に違和感の少ない接続形状を得ることができる. また, メッシュの稜線にタグを付加することで, 角張った形状をその特徴を保持したまま接続することができる. これは, 陰関数表現形式の形状間の融合演算では実現できない操作であり, 本操作の有用性を示している.

本手法の今後の課題として, 以下の 2 つの点を挙げる.

第 1 に, 効率的なメッシュの切り取り操作の開発が挙げられる. 3 次元メッシュの領域指定は, ユーザーが境界を表す稜線や頂点を順に選択していくことで行うが, 複雑なメッシュ形状の場合は大変な労力を必要とする。2 次元グラフィックス・ソフトウェアでは, 貼り付け操作とともに切り取り操作 (cutting) は重要な操作として用いられる. 2 次元の切り取り操作では, 淀型領域指定や色情報の差による領域指定など多数の切り取り領域指定の手段が提供されている. 金井ら [33] は, 3 次元メッシュ上の最短経路を用いた領域指定手法を提案している. この方法は, ユーザーが指定した頂点間の最短経路を離散グラフの選択的詳細化に基づく方法で算出する. この方法で用いられるアルゴリズムは, 近似手法であるが近似精度が高く, 高速であるため, ユーザーはインタラクティブに領域指定の境界の入力・修正が可能である. この他にも, メッシュの連続性や入力点からの距離を利用して領域を選択する方法などが考えられる. この切り取り操作を効率的に行うことができるようにすることで, 本接続操作をモデリングの一連の操作として効果的に適用できるようになる.

第 2 に, メッシュの局所的細分割操作と削除の高速化が挙げられる. 実行結果から分かるように, 本接続操作の実行時間のほとんどは, メッシュの局所的細分割操作と削除に費やされる. これは, メッシュの局所変形操作を多数回用いるために起こり, この操作の回数を減少させることで解決できる. 本論文で提案した方法では, メッシュの干渉領域に対して一様に局所的細分割操作を適用している. しかし, メッシュの接続性や 3 角形のサイズなどに注目し, 視覚的に影響のない部分について細分割を行わなければ, 局所変形操作の適用回数を

減少させ実行時間を短くすることができると考えられる. 計算負荷が低く, 効率よく局所的細分割操作を適用する面を選出する手法が望まれる.

本手法の利点に, 簡単にメッシュの貼り付け操作を行うことができることが挙げられる. この簡易さを生かした本手法の拡張, 改良を行い, 本操作をより実用的なものにしていきたい.

謝辞

本論文を締めくくるにあたり, 日頃から御指導頂き, また本研究のきっかけと適切な助言を頂きました, 本塾環境情報学部の千代倉弘明教授に心より感謝申し上げます.

本塾環境情報学部の専任講師の金井崇氏には, 本論文の副査をお引き受け頂き, 研究初期の段階から研究に関する様々な情報や技術を多数ご教授して頂きました.

本塾環境情報学部の武藤佳恭教授には, 本論文の副査を快くお引き受け下さいました. さらに本稿を作成するうえで, 大変貴重な助言と新しい視点を頂きました.

本稿の執筆にあたり, 東京工科大学メディア学部の渡邊大地専任講師, 本塾政策・メディア研究科の脇田玲氏、同修士課程 2 年の道川隆士氏には惜しみない助言と協力をして頂きました. 特に, 同修士課程 2 年の道川隆士氏には、様々な形で本研究に関する御協力を頂きました。また, 慶應義塾大学千代倉研究室において, 未熟な時分から現在に至るまで共に研究を進めてきた, 修士課程 2 年の斉藤佳奈子氏, 同 1 年の武部佳文氏には熱心に議論をして頂きました. ラティス・テクノロジー社の田中浩司氏には, プログラム実装における技術的なアドバイスを多数頂きました.

例題で用いた兎, アヒルは Viewpoint Data Labs Inc. のメッシュデータである. またトライセラトプス, 大砲, 鷲, 虎のメッシュデータは PLATINUM Technology Inc. のメッシュデータを用いた. はにわモデラー, XVL Viewer の実行画面で用いた赤ちゃんとアールのデータはラティス・テクノロジー株式会社の多角形モデルを用いた.

また本研究の一部は慶應義塾大学森泰吉郎研究振興基金, 日本育英会奨学金, 慶應義塾奨学制度の支援を受けた. ここに感謝の意を表する.

最後に、本研究に御協力頂いた全ての皆様に、厚く御礼申し上げます。

参考文献

[1] Paulo Roma Cavalcanti and Ulisses T. Mello. Three-dimensional constrained delaunay triangulation: A minimalist approach. *Proceedings, 8th International Meshing Roundtable, South lake Tahoe, CA, U.S.A*, pp. 119–129, October 1999.

[2] Keith Kin Yip Chan, Stephen Mann, and Richard Bartels. World space surface pasting. *Graphics Interface '97*, pp. 146–154, May 1997. ISBN 0-9695338-6-1 ISSN 0713-5424.

[3] Philippe Decaudin. Geometric deformation by merging a 3D-object with a simple shape. *Graphics Interface '96*, pp. 55–60, May 1996. ISBN 0-9695338-5-3.

[4] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. *Proceedings of SIGGRAPH 98*, pp. 85–94, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[5] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of SIGGRAPH 99*, pp. 317–324, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[6] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. *Proceedings of SIGGRAPH 95*, pp. 173–182, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[7] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. *Proceedings of SIGGRAPH 96*, pp. 171–180, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[8] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. *Proceedings of SIGGRAPH 93*, pp. 35–44, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California.

[9] T. L. Hilton and P. K. Egbert. Vector fields: an interactive tool for animation, modeling and simulation with physically based 3d particle systems and soft objects. *Computer Graphics Forum*, Vol. 13, No. 3, pp. 329–338, 1994.

[10] G. Hirota, R. Maheshwari, and M. C. Lin. Fast volume-preserving free-form deformation using multi-level optimization. *Computer-Aided Design*, Vol. 32, No. 8-9, pp. 499–512, August 2000. ISSN 0010-4485.

[11] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Proceedings of SIGGRAPH 94*, pp. 295–302, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[12] J. Hoschek and D. lasser. *Fundamentals of Computer Aided Design.* A K Peters, Nattick, Massachusetts, 1993.

[13] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No. 3, pp. 218–230, September 1995. ISSN 1077-2626.

[14] Xiaogang Jin, Youfu Li, and Qunsheng Peng. General constrained deformations based on generalized metaballs. *Computers & Graphics*, Vol. 24, No. 2, pp. 219–231, April 2000. ISSN 0097-8493.

[15] Takashi Kanai, Hiromasa Suzuki, and Fumihiko Kimura. Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics & Applications*, Vol. 20, No. 2, pp. 62–75, March/April 2000. ISSN 0272-1716.

[16] Takashi Kanai, Hiromasa Suzuki, Jun Mitani, and Fumihiko Kimura. Interactive mesh fusion based on local 3D metamorphosis. *Graphics Interface '99*, pp. 148–156, June 1999. ISBN 1-55860-632-7.

[17] Leif P. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer*, Vol. 16, No. 3-4, pp. 142–150, 2000. ISSN 0178-2789.

[18] U. Labsik, L. Kobbelt, R. Schneider, and H.-P.Seidel. Progressive transmission of subdivision surfaces. *Proceedings of Computaional Geometry*, Vol. 15, pp. 25–39, 2000.

[19] Francis Lazarus and Anne Verroust. Metamorphosis of cylinder-like objects. *The Journal of Visualization and Computer Animation*, Vol. 8, No. 3, pp. 131–146, 1997. ISSN 1049-8907.

[20] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98*, pp. 95–104, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[21] C. Loop. Smooth spline surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[22] D. Terzopulus M. Vasilescu. Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarchical subdivision. *Proceedings of Computer Vision and Pattern Recognition Conference*, pp. 829–832, 1992.

[23] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. *Proceedings of SIGGRAPH 96*, pp. 181–188, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[24] M. Mäntylä. *Introduction to Solid Moderling.* Computer Science Press, Rockville, Maryland, 1988.

[25] Joseph OŔourke. *Computational Geometry in C.* Cambridge University Press, New York, 1994.

[26] Mervi Ranta, Masatomo Inui, Fumihiko Kimura, and Martti Mäntylä. Cut and paste based modeling with boundary features. *SMA '93: Proceedings of the Second Symposium on Solid Modeling and Applications*, pp. 303–312, May 1993. Held in held May 19-21, 1993 in Montreal, Quebec, Canada.

[27] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *Proceedings of SIGGRAPH 97*, pp. 279–286, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[28] Hiromasa Suzuki, Shingo Takenchi, Takashi Kanai, and Fumihiko Kimura. Subdivision surface fitting to a range of points. *Pacific Graphics '99*, October 1999. Held in Seoul, Korea.

[29] T.W.Sederberg and S.R.Parry. Free-form deformation of solid geometric models. *Computer Graphics*, Vol. 20, No. 4, August 1986.

[30] Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools*, Vol. 2, No. 4, pp. 1–14, 1997. ISSN 1086-7651.

[31] Malte Zöckler, Detlev Stalling, and Hans-Christian Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, Vol. 16, No. 5, pp. 241–253, 2000. ISSN 0178-2789.

[32] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97*, pp. 259–268, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[33] 金井崇, 鈴木宏正. 離散グラフの選択的詳細化に基づく多面体上の近似最短経路算出とその応用. 情報処理学会研究報告, pp. 13–18, 1999. 99-CG-97.

[34] 鳥谷浩志, 千代倉宏明. 「特集:21 世紀の三次元 CAD を展望する -第 3 部- ゲームから CAD までカバーする斬新なコンセプトのソリッドカーネル」. 日経 CG1998 年 6 月号, 1998.