Poisson 方程式を用いたリメッシング手法

Research on Remeshing Using Poisson Equation

情報工学専攻 金子 晃
KANEKO Hikaru

概 要

本研究では, 3 次元サーフェスメッシュのリメッシングに関する手法を提案する. CAD データからの変換や, 3 次元スキャナを用いて現実の物体をスキャンするといった方法で生成されたメッシュは, データ量が膨大である場合や, メッシュを構成する面や辺といった要素がユーザーの要求と異なる場合がある. そのため, ユーザーの要求に合わせてメッシュをリメッシングする必要がある. ユーザーの要求の一つに, 三角形メッシュから四角形を主としたメッシュへの変換がある. この要求を実現する手法として, 円柱へのパラメータ化を用いたリメッシング手法がある.

円柱へのパラメータ化を用いたリメッシングでは, メッシュの頂点に対して調和関数を定義するが, 定義のための制約点をユーザーが手動で選択する必要がある. また, パラメータ空間における horizontal line と vertical line を用いてリメッシングするため辺の長さの制御が難しく, また円柱へのパラメータ化を伴うため計算に時間がかかる.

そこで本研究では, Poisson 方程式を用いて制約点の候補を自動的に検出する手法と, horizontal line の更新および配置間隔の指定によって出力メッシュの辺長を制御する手法を提案する. また, horizontal line と似た性質を持つ等高線を用いることで, パラメータ化を伴わずに辺長を制御しながらリメッシングする手法も提案し, 計算機実験によりこれらの手法が有効であることを示す.

キーワード： リメッシング, 四角形を主とするメッシュ, 調和関数, Poisson 方程式, 等高線

1 序論

メッシュを扱うアプリケーションにおいて, メッシュを構成する面の数はアプリケーションの処理速度に深く関係している. そのため, モデルの形状をできるだけ変化させずに面の数を減らすことが多くの分野で望まれる. また, 四角形の面で構成されたメッシュであれば, 三角形の面で構成されたメッシュよりも少ない面の数で, 似た形状を表現できる可能性がある. そのため, 本研究では三角形メッシュを四角形を主とするメッシュへリメッシングする問題を扱う.

円柱へのパラメータ化を用いたリメッシング手法 [3] では調和関数の定義のために制約点をユーザーが手動で選択する必要があったが, リメッシングに適した制約点をメッシュを構成する数多くの頂点から手動で選ぶことは難しい. そのため, 本研究では Poisson 方程式を用いた制約点の半自動検出手法 [1] を反復的に用いることで, 制約点の候補を自動的に検出する手法を提案する. [1] の手法ではユーザーの入力によってリメッシングに適した一部の制約点が検出されない場合があったが, Poisson 方程式を反復的に解くことでこうした制約点を検出できる. また, [3] ではメッシュをパラメータ化した円柱の側面上で horizontal line と vertical line を求め, それらが作る格子を出力メッシュの面としていたが, 実空間とパラメータ空間における辺の長さや端点の位置関係の違いから, 辺長の制御が難しかった. そこで本研究では horizontal line 同士の間隔をユーザーがパラメータ h として指定し, 実空間における間隔が h に近づくように horizontal line を生成することで, 出力メッシュの辺長を制御する手法を提案する. また, holizontal line の代わりに調和関数値の等高線を用いることで, 円柱へのパラメータ化を伴わずに提案手法 1 と同様のリメッシングを実現する手法を提案する.

2 Poisson 方程式を用いた制約点検出

[1, 3] では, メッシュ上に調和関数と呼ばれるスカラー場を定義し, 調和関数の勾配ベクトルとそれに直交するベクトルを用いてリメッシングをしていた. 本節では [1, 3] のリメッシング手法で用いられる調和関数について述べる. ここでは入力として与えられる三角形メッシュの頂点集合を V, 面集合を F とする. 調和関数は, $\Delta f = \nabla^2 f = 0$ を満たす実数値関数 $f : V \to \mathrm{R}$ である. ただし境界条件として, 制約点の集合 $C \subset V$ と, 頂点 $i \in C$ に対する関数値 f_i を設定しなければならない. 制約点と関数値の設定について, [3] ではすべてユーザーが任意に設定し, [1] ではユーザーが 1 点の極小点を制約として与えて Poisson 方程式を解くことで, リメッシングに適した制約点とスカラー値を半自動的に設定していた. ここで, リメッシングに適した制約点とは, その制約点によって定義された調和関数を用いると, メッシュの出力方向が形状に沿うようにリメッシングできる制約点であり, 主に尖った形状の先端の点である. 本研究では [1] の手法を基に制約点候補の自動検出手法を提案するので, [1] の手法の概要を以下に示す.

STEP 1 ユーザーが 1 つの制約点と, その点に対するスカラー値を設定する.

STEP 2 STEP 1 の境界条件を用いて Poisson 方程式を解き, メッシュの各頂点に対して設定されるスカラー値が極大となる点の集合を得る.

STEP 3 STEP 2 で得た極大点集合をクラスタリングし, 近くにある頂点をまとめた, いくつかの頂点集合を得る.

STEP 4 STEP 3 で得た, いくつかの頂点集合の中からそれぞれスカラー値が極大となる点を選び, それらと STEP 1 で指定した制約点を全体の制約点とする.

STEP 3 と STEP 4 は余分な極大点を切り捨て, 実際に用いる制約条件を自動的に決定する操作であるが, 本研究では制約点候補の検出を目的とするので, 比較のため STEP 3 と STEP 4 については考慮しない. STEP 1 では一般的に, 尖った形状の先端を制約点と

して設定するのが良いとされている.

horse モデルの耳の中腹の尖った箇所に STEP 1 の制約点を設定し, 制約点の候補を検出した例を図 1 に示す. 足の先端や顔の先端といった箇所に制約点の候補が検出されていることがわかる. しかし, STEP 1 で設定した制約点付近の耳の先端には検出されていない. これは, STEP 1 の制約点付近に尖った形状が存在した場合に, その形状の先端ではスカラー値が収束しないために, 制約点候補として検出されなかったと考えられる.

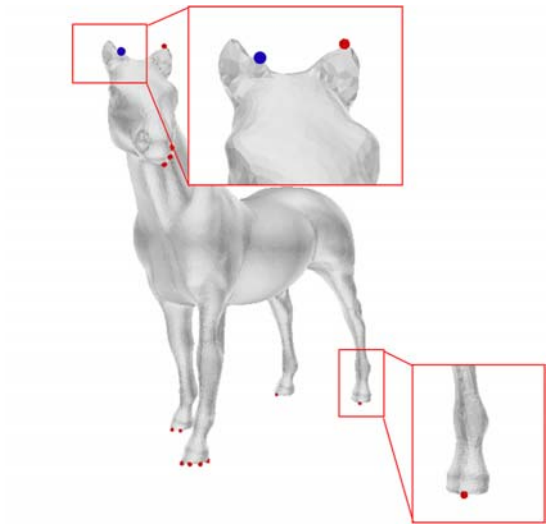


図 1. [1] の手法による極大点検出 (horse モデル)

2.1 制約点候補の検出に関する提案手法

ここでは, [1] の手法の一部を反復的に用いて, 調和関数を定義するために必要な制約点の候補を自動的に検出する手法を提案する. 提案手法の手順を以下に示す. なお, [1] の手法と提案手法の STEP 1 ではユーザーが設定する制約点に対して任意のスカラー値を設定するが, 本研究ではすべて 0 に設定している.

STEP 1 メッシュ上の任意の頂点を制約点として, 任意のスカラー値を設定し, Poisson 方程式を解くことで極大点の集合を求める.

STEP 2 STEP 1 で求めた極大点の集合の中で, スカラー値が最大の点を制約点として任意のスカラー値を設定し, 再び Poisson 方程式を解き, 極大点の集合を求める.

図 1 の例と同じ初期制約点を用いて提案手法を適用した例を図 2 に示す. STEP 2 の制約点には後ろ足の先端の制約点を用いている. 図から, [1] の手法で検出できなかった制約点候補を検出できていることがわかる.

3 円柱へのパラメータ化を用いた手法

三角形サーフェスメッシュから四角形を主としたメッシュへのリメッシング手法の一つとして, 円柱へのパラメータ化を用いた手法がある [3]. 本節ではこの手法の概要を述べる. 円柱へのパラメータ化を用いたリメッシングは以下の流れで行なわれる.

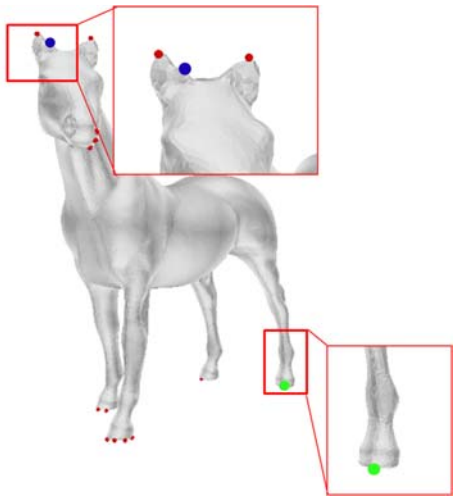


図 2. 提案手法による極大点検出 (horse モデル)

STEP 1 入力メッシュを複数のチューブ型領域へと分割する.

STEP 2 各チューブ型領域を円柱の側面上へパラメータ化する.

STEP 3 各領域を, 対応するパラメータ空間上でリメッシングする.

STEP 1 の操作では, 調和関数の勾配ベクトル $\boldsymbol{g_1}$ と, $\boldsymbol{g_1}$ に直交するベクトル $\boldsymbol{g_2}$ を求め, それらを接線とする曲線である積分曲線と等高線を用いる. 調和関数値に関する鞍点から延ばした積分曲線 (bridge) と, その終点における等高線 (frame) を組み合わせた spectacles をメッシュ上のすべての鞍点について求め, spectacles によってメッシュを分割する (図 3). 分割された領域は, spectacles を構成する bridge を切り離す操作と, 調和関数値の極小 (大) 点の周辺の面からなる極小 (大) 領域を切り離す操作によって, 入力メッシュは極小 (大) 領域以外はすべてチューブ型領域として分割される.

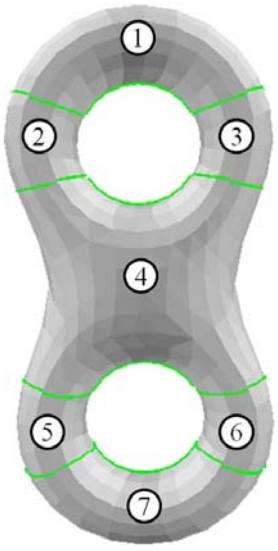
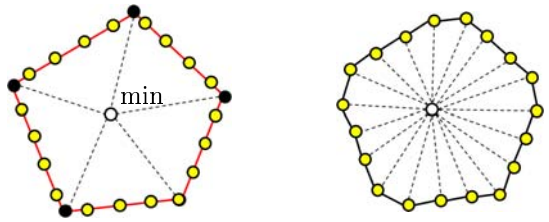


図 3. Spectacles による領域分割

STEP 2 のパラメータ化では, [2] の手法を用いており, 図 5(a) の領域を円柱の側面上へパラメータ化した例が図 5(b) である. STEP 3 では, まず極小領域をリメッシングする. 極小領域のリメッシングは, 領域の境界に頂点を等間隔に配置し, それらの頂点と極小点からなる三角形の面でリメッシングをする (図 4). このとき配置する頂点の数はユーザーが指定した数で, $d_{\min}$ 個とする.

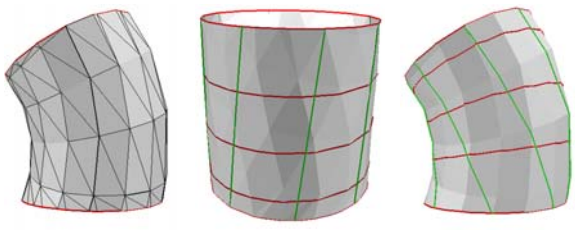


(a) 頂点の配置　　(b) リメッシング

図 4. 極小領域のリメッシング

この操作で極小領域に隣接する領域は一方の隣接領域がリメッシング済みになり, リメッシング後の頂点が境界上に配置された状態となる. これ以降, 片側の隣接する領域がリメッシング済みの領域を順にリメッシングする. ただし, spectacles を境界とする領域については, spectacles を構成する等高線の部分で隣接している領域がリメッシング済みでも, bridge には頂点が配置されていないため, frame におけるリメッシング後の頂点と同程度の配置間隔で頂点を配置する. 以上により, 片側の境界には頂点が配置されている. 次に, もう一方の境界上に, 両境界上の頂点数が等しくなるように頂点を配置する. そして, 異なる境界上に配置された頂点同士をペアにして, 円柱の側面に沿った曲線 (vertical line) で結ぶ. さらに, 円柱の底面に平行な曲線 (horizontal line) を求める. なお, この horizontal line の配置間隔は境界上の頂点の配置間隔から決定する. この 2 種類の曲線で構成される格子を実空間のメッシュ上に射影し, 格子のマスである四角形でリメッシングする. 図 5(a) の領域を円柱の側面上にパラメータ化し, パラメータ空間上で作成した格子 (図 5(b)) を実空間に射影した例が図 5(c) である.

極大領域については, すでに境界上に頂点が配置されているため, 極小領域と同様にリメッシングできる.



(a)　　(b)　　(c)

図 5. 2 つの曲線 (折れ線) で構成される格子

この手法では, horizontal line の配置間隔を境界上の頂点の配置間隔から決めていることや, パラメータ空間における格子を基にリメッシングしていることから, 出力メッシュの実空間における辺の長さを制御することが難しい. また, 円柱へのパラメータ化を行なうため, 計算時間が長い.

4　horizontal line を基にしたリメッシング

本節ではリメッシングに関する 2 つの提案手法について述べる. 1 つは horizontal line 同士の間隔をユーザーがパラメータ h として指定し, 実空間における間隔が h に近づくように horizontal line を生成することで, 出力メッシュの辺長を制御する手法で, これを提案手法 1 とする. もう 1 つは, holizontal line の代わりに調和関数値の等高線を用いることで, 円柱へのパラメータ化を伴わずに提案手法 1 と同様のリメッシングを実現する手法で, これを提案手法 2 とする. なお, どちらの手法も [3] と同様の領域分割を行なう. また, 極小領域へのリメッシングと, 領域の境界への頂点配置方法も [3] と同様に行なうため, 提案手法 1, 2 は片側の隣接する領域がリメッシング済みであるチューブ型領域に対するリメッシング手法である.

4.1　提案手法 1

提案手法 1 では前処理として, [3] と同様にチューブ型領域を円柱の側面上へパラメータ化し, 異なる境界同士の頂点を結ぶ vertical line を求める. ここでは, 実空間において最長の vertical line を l_v とし, リメッシングの対象領域の境界上に配置済みの頂点数を n_{bound} とする. 提案手法 1 は以下の手順で行なう.

STEP 1　l_v 上に, 実空間における長さに基づいて, 一方の境界から h の間隔で頂点を配置する.

STEP 2　l_v 上に配置されたそれぞれの頂点を通る horizontal line を求める.

STEP 3　すべての horizontal line 上に, 実空間で等間隔になるように n_{bound} 個の頂点を配置する.

STEP 4　各 horizontal line 上に配置された頂点と, 境界上に配置された頂点を出力メッシュの頂点として, 四角形の面を生成する.

STEP 1, STEP 2 において実空間上での長さに基づいて h の間隔で horizontal line を生成しているので, 領域の一方の境界からもう一方の境界へ向かう方向に出力される辺の長さが h 程度になる. STEP 4 では隣接する頂点を辺で結ぶことで, 面を生成している.

4.2　提案手法 2

ここでは, 提案手法 1 における horizontal line の代わりに, 調和関数値の等高線を用いることで, メッシュの円柱へのパラメータ化を伴わないリメッシング手法を提案する. 提案手法 2 は一方の境界がリメッシングされたチューブ型領域に対して以下の手順で行なう.

STEP 1　領域内の各頂点の調和関数値を修正する.

STEP 2　領域の片側の境界上に配置されたすべての頂点から, もう一方の境界までの積分曲線を求める.

STEP 3　最長の積分曲線を l'_v として, 提案手法 1 の STEP 1 と同様に l'_v 上に頂点を配置する.

STEP 4　STEP 3 で求めた各頂点の調和関数値に関する等高線を求め, この等高線を horizontal line と見なして提案手法 1 の STEP 3, STEP 4 を適用する.

調和関数値の等高線は, 実空間に射影した horizontal line と似た曲線であるが, spectacles を持つ領域については, 鞍点付近で等高線が分裂してしまうため, そのままでは提案手法 1 と同様のリメッシングはできない. そこで, spectacles が一つの等高線となるように, STEP 1 で調和関数値を修正することで, 等高線の分裂を解消する. そして, STEP 2 では提案手法 1 において前処理で求めてあった vertical line の代わりに, 積分曲線を求める. STEP 3 では l_v の代わりに積分曲線の中で最も長い l'_v を用いている. 次に STEP 3, STEP 4 で horizontal line の代わりに等高線を用いることで, パラメータ化を伴わないリメッシングをする.

4.3 計算機実験

ここでは [3] の円柱へのパラメータ化を用いたリメッシング手法と提案手法 1, 2 を実装し, 計算機実験によって比較を行なった結果を示す. 本研究の実験では入力として, eight モデル（頂点数 766, 面数 1536, $d_{\min} = 30, h = 0.042$）, kitten モデル（図 6, 頂点数 11039, 面数 22078, $d_{\min} = 30, h = 0.042$）, moai モデル（頂点数 10002, 面数 20000, $d_{\min} = 30, h = 0.42$）に対してそれぞれの手法を用いてリメッシングした. 図 7, 8, 9 に kitten モデルに対してそれぞれの手法における出力メッシュの図を示す. 図の比較から, 提案手法 1, 2 は全体的に入力メッシュの形状を保っているが, 耳を表す形状付近では特に, [3] の手法より上手くリメッシングできていることが分かる.

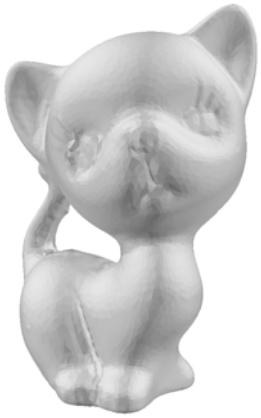


図 6. 入力メッシュ

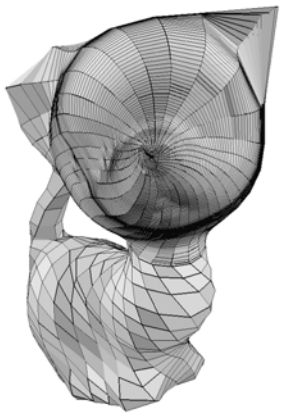


図 7. [3] の手法

次に表 1, 2, 3 で各モデルのリメッシングに関するデータを示す. なお計算時間の単位は秒である. 表 1 のデータと図 7〜9 を合わせて比較すると, 提案手法 1, 2 は少ない頂点と面の数で元モデルの形状を保ち, かつ平均辺長が h に近い値となっていることがわかる. さらに, 提案手法 2 では計算時間も非常に短いことがわかる.

5 結論

本研究では, Poisson 方程式を反復的に解くことで制約点の候補を自動的に検出する手法と, 辺の長さを制御し, かつ三角形メッシュから四角形を主としたメッシュへとリメッシングする手法を提案した. また, 計算機実験によって提案手法の有効性を示した.

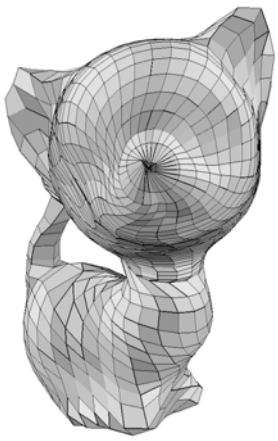


図 8. 提案手法 1

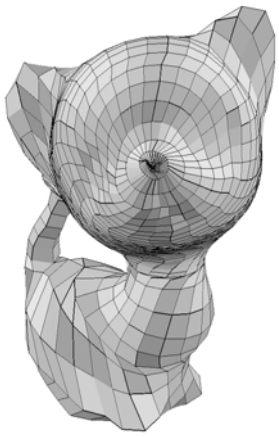


図 9. 提案手法 2

表 1. kitten モデルに対する実験結果

	既存手法	提案手法 1	提案手法 2
頂点数	4172	1523	1334
面数	4195	1546	1357
平均辺長	0.020511	0.0377194	0.0388796
計算時間	39.948	37.938	2.827

表 2. eight モデルに対する実験結果

	既存手法	提案手法 1	提案手法 2
頂点数	1136	786	810
面数	1152	802	826
平均辺長	0.0306735	0.0408875	0.0401201
計算時間	0.465	0.475	0.156

表 3. moai モデルに対する実験結果

	既存手法	提案手法 1	提案手法 2
頂点数	20252	1292	1022
面数	20280	1320	1050
平均辺長	0.198894	0.360464	0.396023
計算時間	65.049	63.875	0.093

リメッシングに関する提案手法 1, 2 は出力メッシュの面の形や法線の方向を考慮せずにリメッシングしているので, これらを考慮することが今後の課題である.

謝辞

本研究を進めるにあたり, 適切な御指導, 御指摘をしていただきました中央大学理工学部情報工学科の今井桂子教授に心から感謝致します. また, 多くの助言をしていただいた今井研究室の学生各位に感謝致します.

参考文献

[1] S. Dong, S. Kircher, and M. Garland, “Harmonic Functions for Quadrilateral Remeshing of Arbitrary Manifolds,” *Computer Aided Geometric Design*, Vol. 22, No. 5, pp. 392-423, 2005.

[2] T. Huysmans, J. Sijbers, and B. Verdonk, “Parameterization of Tubular Surfaces on the Cylinder,” *Journal of WSCG*, Vol. 13, No. 3, pp. 97-104, 2005.

[3] 長野 真之, “調和関数を用いたリメッシング手法,” 中央大学大学院理工学研究科情報工学専攻修士論文, 2006.