調和関数を用いたリメッシング手法

Research on Remeshing through the Use of Harmonic Functions

情報工学専攻 長野 真之
NAGANO Masayuki

概 要

本研究では, 3 次元表面メッシュデータのリメッシングアルゴリズムを提案する. 近年の形状測定技術の発展により, 3 次元スキャナを用いて生成されたメッシュは, 正確に元のモデルの形状を捉えている. そのため, それらのメッシュを様々な分野で応用することが望まれる. しかし, データ量が膨大であることや, メッシュを構成する要素がユーザの要求と異なることから, 応用分野に合わせてメッシュを加工する必要がある. このようにメッシュを加工することをリメッシングという. この要求に応じる手法に, 調和関数を用いたリメッシング手法がある.

調和関数は鞍点の周辺において, 望ましい振る舞いをしない. そこで, 本研究では, 調和関数を spectacles を用いて, よりリメッシングに適した調和関数へと修正するための手法を提案する. また, その手法を実装し, 実際のリメッシングに対して有効であることを示す. また, spectacles を境界とする領域ごとにメッシュを分割し, 各領域ごとにリメッシングすることで特殊な領域を除き四角形で構成された出力メッシュを得る手法を提案し, 実装する.

キーワード： リメッシング, Quad-dominant mesh, 調和関数

1 序論

メッシュをコンピュータグラフィックスや有限要素法, CAD 等に使用したとき, 面の数が処理速度のボトルネックになる. そのため, 元のモデルの形状をできるだけ維持したまま, 面数を減少させることが多くの応用分野で望まれる. さらに, 四角形メッシュを用いたほうが, より少ない面数で曲面を表現できることから, データ量をより少なくできる.

以上より, 三角形を用いた表面メッシュから四角形を主としたメッシュにリメッシングするという考えが生まれる.

[1] の手法では, 調和関数にしばしば鞍点が生じる. 鞍点の周辺における調和関数は四角形を主としたメッシュの生成には適さない. そのため, 本研究では, 鞍点が存在した場合に spectacles [3] を用いて調和関数を修正する手法を提案する. 修正された調和関数を用いてリメッシングを行なうことで, 鞍点周辺における出力メッシュの質を改善することができる. また, spectacles を用いることで, 極大 (小) 点の周辺を除いた領域をチューブ型の部分領域に分割することができる. チューブ型の領域を四角形のみでリメッシングすることは比較的容易である. そこで, 調和関数の極大 (小) 点の周辺を除いた領域を四角形のみでリメッシングする手法を提案する.

本報告の構成は以下の通りである. まず, 2 節で [1] の手法の概要を述べる. 次に, 3 節で [1] の手法によって構築された調和関数を spectacles を用いて修正する手法を提案する. 4 節では, spectacles によって分割された領域ごとにリメッシングを行ない, 特定の領域を除き四角形のみで構成された出力メッシュを得る手法を提案する. 最後に, 5 節で結論を述べる.

2 調和関数を用いた手法

三角形表面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ための手法に [1] の手法がある. 本節では, [1] の手法の概要を記す. ここでは, 入力として与えられる三角形メッシュの頂点の集合を V, 面の集合を F とする. ここで, F の要素はすべて三角形であるものとする. [1] の手法では, 調和関数 u を用いる. 調和関数とは $\Delta u = \nabla^2 u = 0$ を満たす関数 u である. [1] の手法は以下のように 4 つのステップに分かれている.

STEP 1 調和関数 $u : V \to \mathbb{R}$ を定義する.

STEP 2 u から 2 つの直交する接ベクトル場 $\boldsymbol{g}_1, \boldsymbol{g}_2 : F \to \mathbb{R}^3$ を求める.

STEP 3 ベクトル場 $\boldsymbol{g}_1, \boldsymbol{g}_2$ をそれぞれ数値積分し, 積分曲線と等高線を得る.

STEP 4 積分曲線と等高線を基にメッシュ化する.

まず, ステップ 1 では入力メッシュの各頂点にスカラー値を割り当て, 調和関数 u を構築する. u において極大値, 極小値を取る頂点は, ユーザが指定した頂点となる.

次に, ステップ 2 では 2 つのベクトル場 $\boldsymbol{g}_1, \boldsymbol{g}_2$ を求める. $\boldsymbol{g}_1$ は u の勾配ベクトルとし, $\boldsymbol{g}_2$ は $\boldsymbol{g}_1$ に直交するベクトル場とする. ステップ 3 では, $\boldsymbol{g}_1, \boldsymbol{g}_2$ を接線に持つ曲線である積分曲線, 等高線を求める. 最後に, ステップ 4 では積分曲線, 等高線から, 出力となる四角形を主としたメッシュを生成する. まず, 積分曲線と等高線の交点を新たなメッシュの頂点とする. 次に, 新たな頂点間を積分曲線または等高線が接続しているとき, 新たな辺を結ぶ. そして, 辺で囲まれた領域を新たな面とする.

図 1 に [1] の手法を用いてリメッシングした結果を示す. この図より, [1] の手法による出力メッシュは, ほぼ全域において四角形の面で構成されていることがわかる. しかし, 図の黒枠部分を見ると, 一部領域において入力メッシュの形状を保持できていない. これは調和関数の鞍点の周辺において, 非常に大きな面が生成されるためである.

3 Spectacles を用いた調和関数の修正

本研究では, 鞍点の周辺における出力メッシュを改善するために, 鞍点周辺の調和関数を修正する手法を提案する. 提案手法の大まかな流れを以下に記す.

STEP 1 極小点から調和関数の小さい順に鞍点を探索し, 鞍点において spectacles [3] を求める.

表 1. 計算時間.

	各ステップに要した時間 [秒]				
	調和関数	修正	積分曲線, 等高線	メッシュ化	合計
eight モデル	0.047	0.077	0.297	0.109	0.530
kitten モデル	124.156	0.234	5.297	0.313	130.000
rockerArm モデル	72.641	0.562	116.563	2.595	192.361

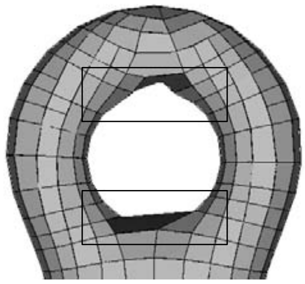

図 1. 既存手法による出力メッシュ.

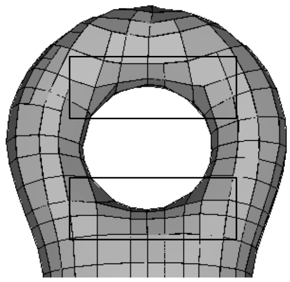

図 3. 提案手法による出力メッシュ.

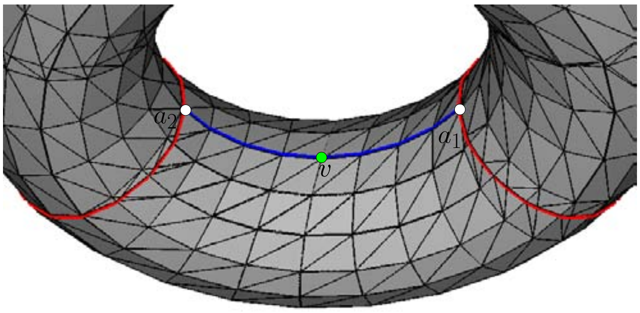

図 2. 頂点 v における spectacles.

STEP 2 Spectacles 上の関数値が等しくなるように, 調和関数 u を再計算する.

上記のように調和関数を修正したのちの処理は [1] の手法と同様に行なう. これ以降本節では, 提案手法の各ステップについて詳述する.

3.1 Spectacles

ステップ 1 では, 頂点 v が鞍点であるか否かを判定する必要がある. 頂点 v に隣接する頂点の関数値を調べることで, 頂点 v が鞍点であることは容易にわかる. 頂点 v が鞍点であった場合, $\boldsymbol{g}_1$ を基に v を始点とする 2 本の積分曲線を求める. 図 2 に鞍点 v から a_1, a_2 へと至る積分曲線を示す. この 2 本の積分曲線を bridge と呼ぶ. bridge の長さは出力メッシュの辺長の数倍とする. bridge の 2 つの終点では, それぞれ等高線を求める. この等高線を frame と呼ぶ. bridge と frame を合わせて頂点 v の spectacles と呼ぶ. Spectacles は [3] において距離関数を用いて定義された概念である. 本研究では, spectacles を調和関数上で構成し, 調和関数の修正に用いる. Spectacles を用いた調和関数の修正については次節で詳述する.

3.2 調和関数の修正

ステップ 2 では spectacles 上の関数値が等しくなるように調和関数を修正する. つまり, spectacles がある値の等高線となるようにする. また, それに合わせて spectacles 周辺の頂点における調和関数値も再計算する. [1] の手法では, 調和関数を計算する際, 頂点数に比例する規模の連立一次方程式を解いている. この手法は頂点数が増加すると非常に計算時間がかかる. そこで本研究では, 調和関数を再計算する際に以下のアルゴリズムを用いる.

Spectacles s 上の頂点の集合を V_s, ユーザの指定した極大 (小) 点の集合を C とする. このとき, s の frame 上の頂点における関数値 u_s を用いて, 任意の $i \in V_s$ に対して $u_i = u_s$ とする. 次に, $u_i^0 = u(i)$ とおき, すべての頂点 $i \in V \setminus \{V_s \cup C\}$ に対して以下の式を反復する.

$$u_i^{k+1} = \frac{u_i^k}{2} + \frac{\sum_{j \in N_i} w_{ij} u_j^k}{2 \sum_{j \in N_i} w_{ij}}$$

u_i^k が収束したら, その値を新たな調和関数値とする. 一般に, 収束した値 u_i^k と初期値 u_i^0 の差は大きくない. そのため, この反復式は短時間で収束する.

3.3 計算機実験

本節では, 提案手法を実装し, 計算機実験を行なった結果を示す.

本実験では, eight モデル (頂点数 766, 面数 1536), kitten モデル (頂点数 11039, 面数 22078), rockerArm モデル (頂点数 40177, 面数 80354) の 3 つのメッシュを入力として用いた. まず eight モデルに対して提案手法を用いてリメッシングした結果を図 3 に示す. 鞍点周辺における面を図 1 と図 3 で比較すると, 提案手法を用いた方が入力メッシュの形状を保持できていることがわかる.

次に, 計算時間について述べる. 提案手法を用いて 3 モデルをリメッシングした際にかかった計算時間を表 2 に示す. 表 2 には,

- 調和関数を求めるために要した時間
- 提案手法を用いて調和関数を修正するために要した時間
- 積分曲線と等高線の計算に要した時間
- メッシュ化に要した時間

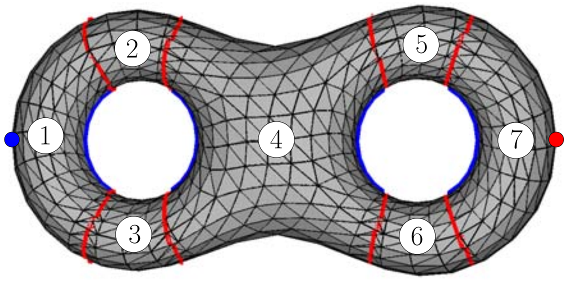

図 4. 領域分割.

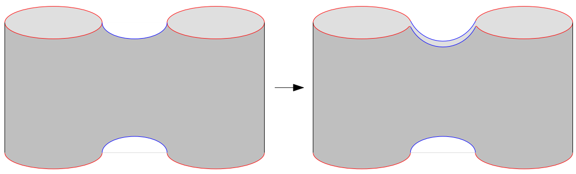

図 5. Bridge の切り離し.

を示す.

表 1 より, 提案手法に要する計算時間はリメッシング全体から見ると無視できる程度の時間であることがわかる.

4 円柱へのパラメータ化を用いたリメッシング

入力されたメッシュを spectacles を境界とする領域に分割すると, 極大 (小) 点を含まない領域はすべてチューブ型にすることができる. ここでいうチューブ型の領域とは, 閉曲線の境界を 2 つ持つ領域である. 図 4 における領域 4 は一見するとチューブ型とはいえない. しかし, spectacles の bridge を図 5 のように切り離すことで, チューブ型としてみなすことができる. また, 領域 1 のように極大 (小) 点を含む領域は, 極大 (小) 点に隣接する面を取り除くことで他の領域と同様にチューブ型の領域として見なせる.

[2] の手法を用いるとチューブ型の各領域は円柱の側面にパラメータ化できる. パラメータ化とは, 3 次元空間上のメッシュを 2 次元平面や球体等に何らかの規則で射影することである. 円柱にパラメータ化された領域を四角形のみの面で構成されるメッシュへとリメッシングすることは比較的容易である. そこで, 本節では spectacles によって分割された領域を円柱上にパラメータ化することで, 入力された三角形メッシュから四角形メッシュへとリメッシングする手法を提案する.

4.1 極小点を含む領域のリメッシング

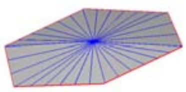

図 6. 極小領域.

提案手法では, 極小点を含む領域からリメッシングを行なう. 極小点を含む領域は極小点に隣接する面を取り除きチューブ型にする. そして, 極小点に隣接する面のみからなる領域 (極小領域) からリメッシングする. まず, 極小点に出力メッシュの頂点を配置する. この頂点の次数は $d_{\min}$ としてユーザから入力されているものとする. 次に, 極小領域の境界に $d_{\min}$ 個の頂点を配置し, 図 6 のように三角形のみを用いてリメッシングする.

極小領域をリメッシングすると, 極小領域の境界には出力メッシュの頂点が配置される. よって, 極小領域と分割されたチューブ型の領域の境界には頂点が配置されていることになる. 極小領域の次にリメッシングを行なうのは, 極小領域に隣接する領域とする. 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していく.

4.2 チューブ型の領域のリメッシング

チューブ型の領域をリメッシングするために, まず, [2] の手法を用いてチューブ型の領域を円柱の側面にパラメータ化する. 図 7 (a) の領域をパラメター化した結果を図 7 (b) に示す.

領域の境界は 1 つの frame のみで構成される場合と 1 つの spectacles そのものが境界となっている場合がある. 1 つの frame のみで構成される境界の場合は, その境界で隣接する 1 つの領域がリメッシングされていればよい. 一方, spectacles そのものが境界となっている場合, frame で隣接している領域がすべてリメッシングされても, bridge 部分に頂点が配置されていない. そこで, リメッシングを行なう前に bridge 上に頂点を配置する. Bridge 上に配置する頂点の配置間隔は frame 上の頂点の配置間隔と同程度にする.

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 (図 7 (b) の黄点). ただし, 図 2 の点 a_1, a_2 のように, spectacles 上の点で次数が 3 以上の点には必ず頂点を配置しなければならない. そして, 図 7 (b) の青線のように異なる境界上の頂点間を線分で結ぶ. さらに, 図 7 (b) の赤線のように円柱の底面に平行な曲線を求める. この 2 種類の曲線をもとのメッシュ上に射影すると, 図 7 (c) のように四角形格子ができる. これ以降は積分曲線と等高線の場合と同様に処理する. 両境界上に配置された頂点数が等しいため, この処理により生成されるメッシュは四角形のみで構成される.

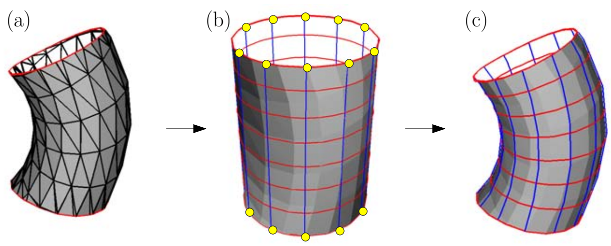

図 7. チューブ型の領域.

4.3 計算機実験

本節で提案した手法を実装し, 計算機実験を行なった結果を示す. 実験に用いた入力メッシュは 3.3 節の実験で用いたものに加え, horse モデル (頂点数 48485, 面数 96966) を用いた.

まず, eight モデルに対して提案手法を用いた結果を示す. $d_{\min} = 30, 60$ としたときの結果を図 8, 9 に示す. 提案手法では $d_{\min}$ により出力メッシュの詳細度を

表 2. 計算時間.

	$d_{\min}$	各ステップに要した時間 [秒]				
		調和関数	領域分割	パラメータ化	メッシュ化	合計
eight モデル	30	0.047	0.078	2.063	0.484	2.678
kitten モデル	30	127.095	0.250	33.655	1.048	162.048
rockerArm モデル	60	78.032	1.187	144.595	4.485	228.299
horse モデル	60	346.547	1.406	178.129	3.530	529.612

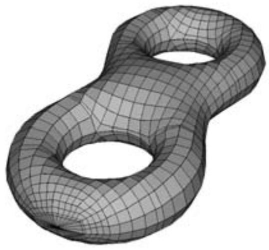

図 8. $d_{\min} = 30$.

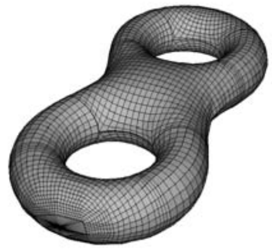

図 9. $d_{\min} = 60$.

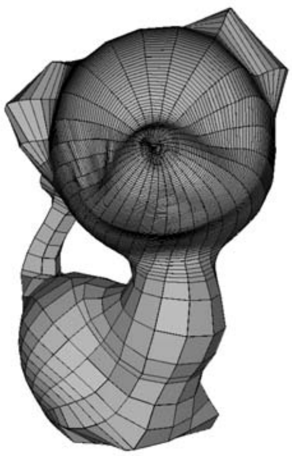

図 10. kitten.

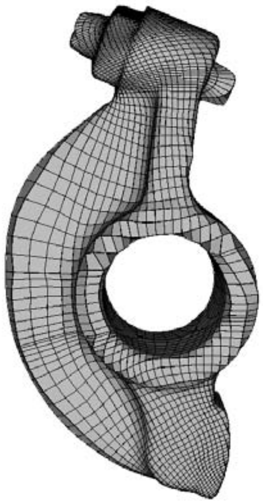

図 11. rockerArm.

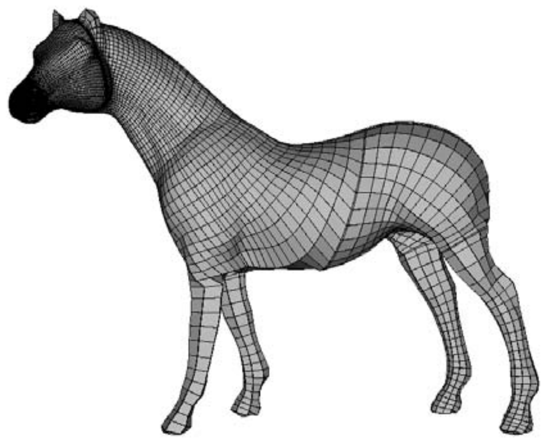

図 12. horse.

コントロールする. 図より, 極大 (小) 点の周辺を除き四角形のみでリメッシングできていることが分かる.

次に, kitten モデル, rockerArm モデルに対する実験結果について述べる. kitten モデルに対して $d_{\min} = 30$ としてリメッシングを行なった結果を図 10 に示す. また, rockerArm モデルに対して $d_{\min} = 60$ でリメッシングした結果を図 11 に示す. また, horse モデルをリメッシングした結果を図 12 に示す. 提案手法は, horse モデルのように細長い形状をしたメッシュに対して非常に有効である.

最後に, 提案手法に要する計算時間について述べる. 各モデルに対して提案手法を実行したときの計算時間を表 2 に示す. 表 2 を見ると, 3 節の提案手法と比較してリメッシングに要する時間は長くなっている. 特に, 各領域をパラメータ化するための計算時間が長い.

5 結論

本研究では, 入力された三角形メッシュからその形状をできるだけ保持したまま, 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. まず, [1] の手法に対して spectacles を用いることで, 鞍点周辺の出力メッシュを改善する手法を提案した. また, 提案手法を実装し調和関数をそのまま用いた場合と提案手法により調和関数を修正した場合を比較した. その結果, 鞍点の周辺における出力メッシュの形状を改善できていることが分かった. 次に, spectacles を用いて分割した領域ごとにリメッシングすることで, 極大 (小) 点の周辺を除き四角形のみでリメッシングする手法を提案し, 実装した. 計算機実験から細長い形状に対する提案手法の有効性が確認できた.

最後に, 今後の課題について述べる. $d_{\min}$ による詳細度のコントロールは感覚的に分かりづらいため, より直感的に出力メッシュの詳細度をコントロールする手法を考える必要がある. さらに, 極小領域に対するリメッシング方法を変更すれば, 四角形のみでリメッシングする手法へと拡張できるかもしれない.

謝辞

本研究を進めるにあたり, 適切な御指導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いたします. また, 多くの助言をしていただいた今井研究室の友人達に感謝いたします.

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] O. Sifri, A. Sheffer, and C. Gotsman, “Geodesic-based Surface Remeshing,” In *Proceedings of 12th International Meshing Roundtable*, pp. 189–199, 2003.