

厳密制約を用いたインタラクティブなメッシュ変形手法

Interactive Mesh Deformation with Hard Constraints

吉岡 康博[†], 正会員 増田 宏[†], 古川 慈之^{††}

Yasuhiro Yoshioka[†], Hiroshi Masuda[†] and Yoshiyuki Furukawa^{††}

Abstract

Interactive mesh editing techniques typically encode geometric shapes using discrete differential equations and approximately solve them using the least squares method. However, these techniques fail to precisely preserve user-defined constraints. In our method, user-defined constraints, such as positional constraints, are treated as hard constraints, and then the combination of hard and soft constraints is converted into a typical least-squares system using QR decomposition. Our method stably and efficiently calculates vertex positions that preserve differential properties. A well-known problem with hard constraints is over-constraints. If over-constraints are involved, they may cause the solver to halt the computation. We show how our framework resolves over-constrained situations, which include redundant and conflicting constraints.

キーワード: コンピュータグラフィックス, メッシュモデリング, デフォーメーション

1. ま え が き

近年, インタラクティブなメッシュ変形手法が活発に研究されている. これらの研究は幾何的な詳細形状を保存しながら, 自由曲面を直感的に編集するためのモデリングツールの開発を目的としている.

典型的なメッシュ変形においては, まず初めにメッシュ上で固定する領域と, 操作するためのハンドルとなる領域を選択する. ハンドルとは, パラメトリック曲面の制御点に相当するもので, 曲面を変形するためにユーザが操作する対象のことである. ユーザが, スクリーン上でマウスやスライダをドラッグしてハンドルを移動させることにより, ハンドルを構成する頂点が移動し, それに追従するように曲面全体が変形する.

最近では, インタラクティブなメッシュ変形手法として, 曲面の微分量を保存する手法¹⁾⁻⁴⁾が提案されている. これらの手法では, メッシュの各頂点における微分量が頂点に関する線形式で記述される. メッシュ編集過程において, ハンドルの頂点や固定頂点は境界条件として扱われ, 変形の前後で微分量ができる限り満たされるように曲面の座標

頂点が決められる.

ただし, 各頂点における微分量の制約と頂点の位置制約の個数の和は頂点数よりも多いため, 過剰制約となって厳密解は存在しない. 従来手法では最小 2 乗法で近似的に解かれるため, ハンドル頂点等の位置制約は正確には満たされない. 最小 2 乗法でも制約の重みを変化させて制約の強弱をつけることができるが, 与えられた許容誤差を満たす重みを求めることは一般には難しい.

本研究では, 位置制約の記述に厳密制約を導入することで, 位置制約を正確に満たしながらインタラクティブなメッシュ変形を安定的に行う手法を提案する. 一方で, 厳密制約においては, 過剰制約が与えられた場合に, 数値計算が破綻するという問題が知られている. 本研究では, こうした過剰制約の検出と解消の問題についても扱う.

本論文の貢献は以下のようにまとめられる.

- 位置制約を正確に満たすメッシュ変形手法の提案.
- 過剰制約の検出と解消手法の提案.

2. 厳密制約を用いたメッシュ変形

メッシュモデリングは, その柔軟性から, 意匠デザインの支援などにおいても有望である. そうした応用におけるメッシュモデリングでは, できるだけ満たせばよい「緩い制約」だけでなく, 必ず満たさなければならない「厳密制約」が有用である. しかし, 現状ではこの両方を扱えるインタラクティブな形状変形手法は知られていない.

本研究では, これらを同時に扱える形状変形手法を提案する. ここでは, ユーザが明示的に与えた寸法制約や位置

2006 年 4 月 3 日受付, 2006 年 8 月 22 日再受付, 2006 年 10 月 17 日採録
[†] 東京大学 大学院 工学系研究科

(〒 113-8656 文京区本郷 7-3-1, TEL 03-5841-6511)

^{††} 産業技術総合研究所

(〒 305-8564 茨城県つくば市並木 1-2, TEL 029-861-7059)

[†] School of Engineering, The University of Tokyo

(7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656)

^{††} National Institute of Advanced Industrial Science and Technology

(1-2 Namiki, Tsukuba-shi, Ibaraki 305-8564)

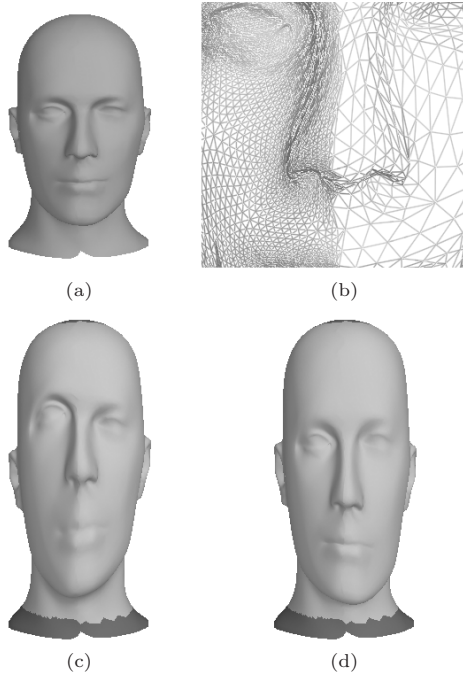


図 1 異なる重みによる変形結果の比較.
Comparison of different weights.

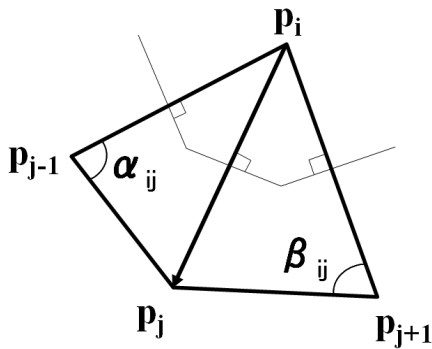


図 2 α_{ij} と β_{ij} の定義
Definition of α_{ij} and β_{ij} .

制約を厳密制約として扱い、元の形状の平均曲率を緩い制約として扱う。また、ここでのインタラクティブとは、数秒以内の前処理によって、ユーザが曲面形状を連続的に変更し表示できることを指す。

2.1 離散ラプラス演算子

n 頂点の三角形メッシュモデル $M = (V, E, F)$ を考える。 V, E, F はそれぞれ頂点、稜線、面の集合である。それぞれの頂点は 3 次元座標 $\mathbf{p}_i \in \mathbb{R}^3$ をもつ。また、各頂点の離散ラプラス演算子は頂点座標を用いて次のように定義される。

$$\delta_i = \sum_{j \in N(i)} w_{ij} (\mathbf{p}_i - \mathbf{p}_j), \quad (1)$$

ここで、 $N(i) = \{j | (i, j) \in E\}$ は頂点 i の隣接頂点の集合である。

重み w_{ij} の決定方法は数値計算の安定性と変形結果の品質にとって重要である。 w_{ij} が不適切に設定されると、形状に歪みが生じたり、計算が破綻してしまう原因となる。

重みを指定する単純な方法は、次に示す均一な重みを設定

することである。ここで、 $|N(i)|$ は隣接頂点の個数を表す。

$$w_{ij} = \frac{1}{|N(i)|} \quad (2)$$

均一な重みによる計算は数値的に安定であるため、この重みを用いている既存研究もある³⁾。しかし、均一な重みはメッシュを構成する三角形の大きさと形がほぼ均一であることを前提としている。図 1 は、メッシュモデルの首の部分を固定し、頭頂部に指定されたハンドル領域を上方に移動させた変形を示しているが、三角形が均一であるという条件が満たされない図 1(b) のような不均一なメッシュを含むため、図 1(c) のような歪んだ形状を生成してしまう。NURBS 曲面などのパラメトリック曲面に対するメッシュ分割では、細長い歪な三角形や、面積が一定でない三角形が生成されることが多く、そうした場合には均一な重みは適していない。

不均一なメッシュに対しては、次のコタンジェントによる重みがよく用いられる⁵⁾。この式では、重みの値は、三角形の大きさと底辺の角度に応じて異なる。

$$w_{ij} = \frac{1}{4\text{Area}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) \quad (3)$$

ここで、 α_{ij} と β_{ij} は図 2 のように稜線 (i, j) を挟む二つの角の角度である。また、 $\text{Area}(i)$ は頂点 $i \in V$ のポロノイ面積である。頂点 i のポロノイ面積とは、最も近い頂点が頂点 i となる領域の面積である。なお、式 (3) は、頂点 i における平均曲率法線の近似となっていることが知られている⁵⁾。

式 (3) は頂点周囲のポロノイ面積が異なる場合や、隣接する頂点が均等に並んでいない場合でも式 (2) を用いた時のように結果が歪むことがない。図 1(a)(b) のように非対称にメッシュ分割されたモデルに対しても、式 (3) を用いた場合では図 1(d) のようにメッシュの位相の影響の少ないよりよい結果が得られている。

2.2 従来手法による緩い制約のみを用いた変形

変形の前後において離散ラプラス演算子の式 (1) を保存することで、形状の凹凸を保持した変形を行うことができる。ここでは、緩い制約のみを用いた従来手法による形状変形について説明する。

従来手法においては、変形後の頂点座標 $\mathbf{p} \in (\mathbb{R}^3)^n$ を次式のような最小 2 乗法によって求めている³⁾。

$$\min_{\mathbf{p}} \left(\sum_i \omega_i \left\| \sum_{j \in N(i)} w_{ij} (\mathbf{p}_i - \mathbf{p}_j) - \hat{\delta}_i \right\|^2 + \sum_{i \in P} \lambda_i \|\mathbf{p}_i - \mathbf{u}_i\|^2 \right) \quad (4)$$

ここで、 P は位置制約を与える頂点の集合、 $\hat{\delta}_i$ は初期状態のラプラス演算、 $\mathbf{u}_i \in \mathbb{R}^3$ は頂点位置を指定するためのベクトルで、元のメッシュの座標値もしくはユーザが指定した移動後の座標値が与えられる。

式 (4) は \mathbf{p} に関する線形の式から構成されるため、行列表現により次のように記述することができる。

$$\min_{\mathbf{p}} \|\mathbf{A}\mathbf{p} - \mathbf{c}\|^2 \quad (5)$$

さらに、線形式に関する最小 2 乗法は連立 1 次方程式に帰着されるため、式 (5) は以下のように解くことができる。

$$\mathbf{A}^T \mathbf{A} \mathbf{p} = \mathbf{A}^T \mathbf{c} \quad (6)$$

$\mathbf{A}^T \mathbf{A}$ は正定値対称行列であるので、式 (6) は効率的な数値解法が知られている⁶⁾。

しかしながら、式 (4) を用いて変形した場合、ハンドル頂点の位置制約は近似的にしか満たされない。 λ_i を大きくすることで位置制約はより厳密に満たされるが、大きな λ_i は数値計算を不安定にするという問題がある。また、メッシュが不均一な場合には、式 (3) で定義される重みの値は各頂点によって大きく異なるため、位置制約を許容誤差内で満たし、かつ数値計算を過度に不安定にしない λ_i の値を求めることは容易ではない。

2.3 制約付き最小 2 乗法による厳密制約の導入

本研究では厳密制約を完全に満たしながら、緩い制約を最小 2 乗法により近似的に解く。ここで、緩い制約と同様に、厳密制約を以下のような線形式で記述する。

$$\mathbf{B}\mathbf{p} = \mathbf{d} \quad (7)$$

本研究における厳密制約を用いたメッシュ変形フレームワークは、制約付き最小 2 乗法によって式 (8) のように定式化される。

$$\min_{\mathbf{B}\mathbf{p}=\mathbf{d}} \|\mathbf{A}\mathbf{p} - \mathbf{c}\|^2 \quad (8)$$

ただし $\mathbf{A} \in \mathbb{R}^{l \times n}$, $\mathbf{B} \in \mathbb{R}^{m \times n}$, $\mathbf{c} \in \mathbb{R}^l$, $\mathbf{d} \in \mathbb{R}^m$, $m \leq n \leq l$ である。また、 n は頂点数、 l は緩い制約の式の個数、 m は厳密制約の式の個数である。

ここで、 \mathbf{B} の階数が m であると仮定すると、 \mathbf{B}^T は QR 分解⁷⁾ により以下のような行列の積として表すことができる。

$$\mathbf{B}^T = \mathbf{Q}\mathbf{R}, \quad (9)$$

ここで $\mathbf{Q} \in \mathbb{R}^{n \times n}$ は直交行列、 $\mathbf{R} \in \mathbb{R}^{n \times m}$ は上三角行列である。

ここから、次のように $\mathbf{p}_1, \mathbf{p}_2$ を定義する。

$$\mathbf{Q}^T \mathbf{p} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} \begin{matrix} m \\ n - m \end{matrix} \quad (10)$$

以上より、式 (8) は次のように表現される。

$$\min_{\mathbf{R}_1^T \mathbf{p}_1 = \mathbf{d}} \|\mathbf{A}\mathbf{Q}_1 \mathbf{p}_1 + \mathbf{A}\mathbf{Q}_2 \mathbf{p}_2 - \mathbf{c}\|^2 \quad (11)$$

\mathbf{p}_1 は $\mathbf{R}_1^T \mathbf{p}_1 = \mathbf{d}$ によりあらかじめ求められるので、 \mathbf{p}_2 は次の制約のない最小 2 乗問題として求められる。

$$\min_{\mathbf{p}_2} \|\mathbf{A}\mathbf{Q}_2 \mathbf{p}_2 - (\mathbf{c} - \mathbf{A}\mathbf{Q}_1 \mathbf{p}_1)\|^2 \quad (12)$$

これは正定値対称行列の解法問題に帰着するため、効率的に解くことができる。

3. 過剰制約の解消

本研究では、他の制約と同時に成立することがありえない制約を矛盾制約、論理的には矛盾しないが他の制約から導出できる制約を冗長制約と呼ぶ。また、矛盾制約と冗長制約を総称して過剰制約と呼ぶことにする。

厳密制約の導入は、制約を厳密に満たすために有効である反面、厳密制約に過剰制約が存在した場合に、計算の破綻を招く。通常のインタラクティブなメッシュ変形では、ユーザは任意の箇所に制約を与えることができるため、制約の整合性を管理するための仕組みが重要となる。そこで、過剰制約を QR 分解の過程でランク落ち (rank deficiency) として検出し、取り除くことを考える。

本研究では、QR 分解 $\mathbf{B}^T = \mathbf{Q}\mathbf{R}$ をハウスホルダー変換⁷⁾ により求める。 \mathbf{B}^T の各列はそれぞれ一つの厳密制約の式を表す。ハウスホルダー変換の過程で、各列は逐次的に処理される。QR 分解を行う前、メッシュモデル中の n 頂点は n 自由度を持ち、 i 番目の列が処理された後、 $n - i$ の自由度を持つ。 \mathbf{B}^T が冗長性や矛盾を含有していない場合、QR 分解によって自由度は $n - m$ となる。

\mathbf{B}^T の j 番目の列が過剰制約であった場合、第 j 列を処理しても自由度を減らすことができない。この時、行列は第 j 列の対角成分以下の成分がすべて 0 と等しくなるため、これを監視することで過剰な制約を検出することができる。

過剰な制約が検出された場合、その列は無視され、次の列が処理される。すべての過剰な制約を除去することで、式 (8) の解が一意に決まる。

すべての過剰制約が除去された後、算出された解が除去された過剰制約を満たすかどうかを検査する。解がその制約を満たす場合は、冗長な制約としてそれを単に無視する。そうでない場合は、矛盾した制約であるため、アプリケーションは制約の修正を促す警告を表示する。

図 3 は過剰制約を与えた場合の結果である。濃い色の部分が位置制約を与えられた領域で、図 3(a) では丸で囲った領域に二重の位置制約を与えている。図 3(c) では二つの位置制約が異なるため、これらは矛盾制約となる。

図 3(b) は既存手法による変形結果で、すべての制約は緩い制約として記述されている。この場合、計算が破綻することはないが、どの制約が競合しているかを検出することは困難である。図 3(c) は本研究の手法により矛盾制約を自動的に検出して、後から指定した制約を除去することで解決した結果である。この結果が望ましいものでない場合、もう一方の先に指定された制約を除去する必要があるが、その場合には、最初に除去された制約を行列 \mathbf{B}^T の第一列目に挿入してから再度ハウスホルダー変換を施すことで、他

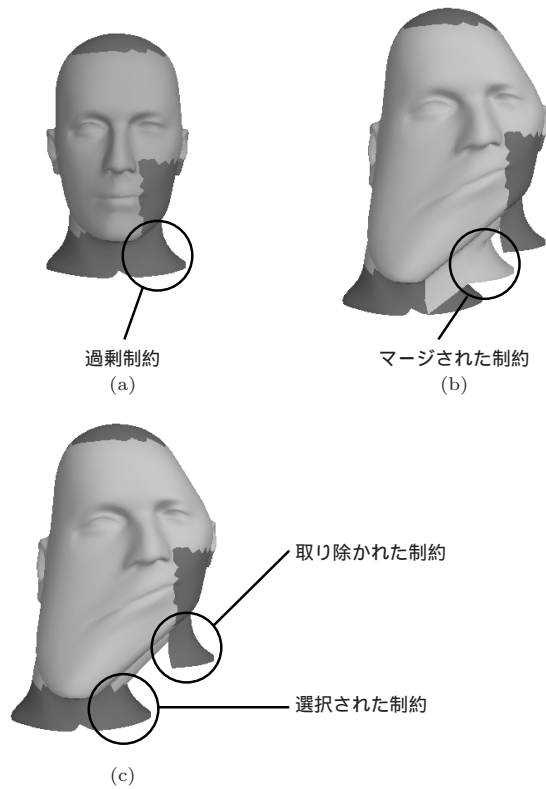


図 3 過剰制約の処理の例
An example of the conflict management.

方の矛盾制約を検出・除去することができる。厳密制約に過剰制約が含まれる場合には、ランク落ちで計算が破綻することが問題となるが、本手法を用いることで計算の破綻が回避でき、変形結果を出力して確認、修正することが可能となる。

4. 実験と評価

制約付き最小 2 乗法に基づく形状変形を評価した結果を示す。ここでは、緩い制約のみを用いた従来手法と比較をして、計算の安定性と計算時間を評価する。

計算時間は CPU: 2.0 GHz Pentium M, メモリ: 1.0 GB, Windows OS の PC を用いて計測した。実装は C++ を用いて行った。本研究では最小 2 乗法の計算に、数値計算ソルバ TAUCS⁶⁾ を用いて線形システムをコレスキー分解により解いている。数値計算は倍精度で行った。

4.1 計算の安定性

本研究では図 4, 図 5 に示すモデルを用いて評価を行った。色の濃い部分は位置制約を与えた領域である。この実験では、従来手法の計算において、式 (4) で $\omega_i = 1.0$, $\lambda_i = 5.0$ を用いて評価した。図 6 は図 4 のモデルの一部を拡大したものである。この図に示すようにここで用いるモデルは不均一なメッシュで構成されているため式 (3) の重みを用いなければ正しい結果を得ることができない。

表 1 は数値計算の安定性の評価結果である。本研究で提案した手法を用いた場合、すべてのモデルに関して正しく

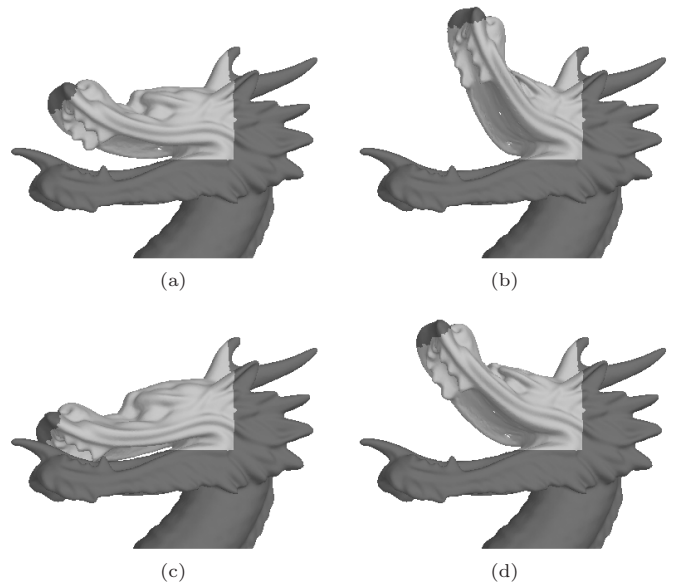


図 4 提案手法による変形の例 (1)
An example of deformation. (1)

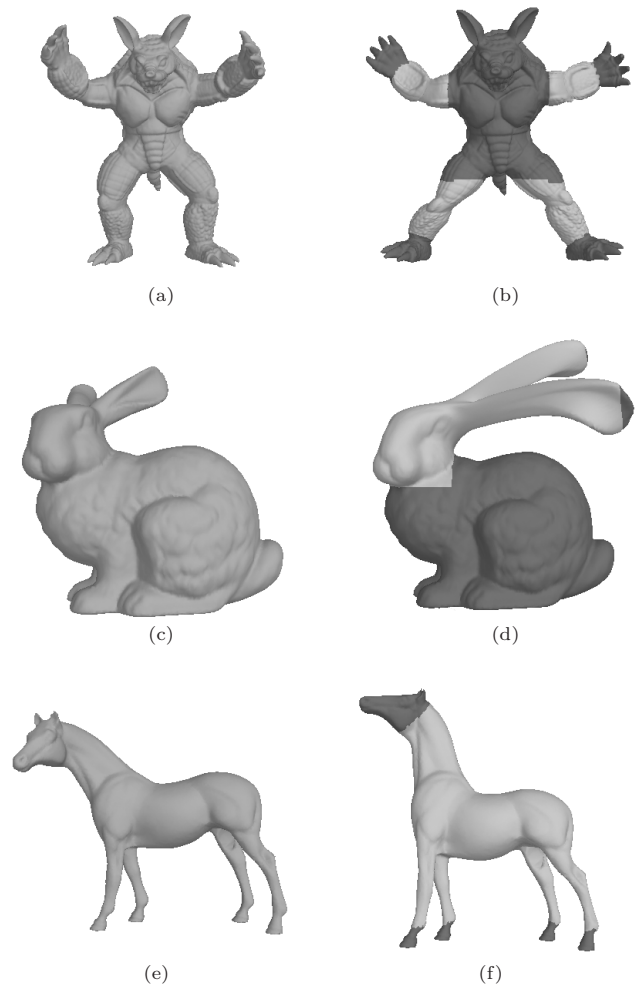


図 5 提案手法による変形の例 (2)
An example of deformation. (2)

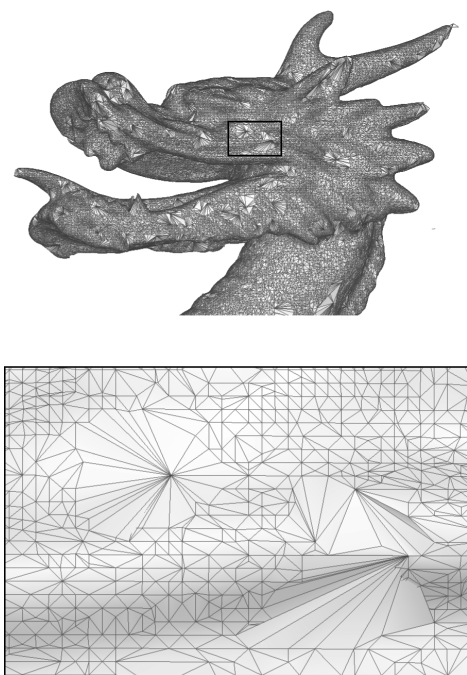


図 6 歪な三角形を含むメッシュの例
An example of the mesh which has bad shaped triangles.

表 1 オーバフローに対する頑健性の比較
(LS: 緩い制約のみ. QR: 緩い制約と厳密制約.)

Comparison of stability against overflow.
LS: Only soft constraints. QR: Both soft and hard constraints.

| Weighting | #vertices | LS | | QR | |
|------------------------|-----------|-----|-----|-----|-----|
| | | Uni | Cot | Uni | Cot |
| (a) Armadillo (図 5(b)) | 172970 | ○ | ○ | ○ | ○ |
| (c) Bunny (図 5(d)) | 34835 | ○ | × | ○ | ○ |
| (b) Dragon (図 4) | 437645 | ○ | × | ○ | ○ |
| (d) Horse (図 5(f)) | 48485 | ○ | ○ | ○ | ○ |

計算が行われた。しかしながら、最小 2 乗法を用いた従来手法による計算では Bunny と Dragon において式 (3) の重みを用いた場合、行列計算が破綻してしまい結果を出力することができなかった。

以上の結果から、本手法は従来手法と同等以上に安定であるということが言える。従来手法の最小 2 乗法による形状変形においては、位置制約に相対的に大きな最小 2 乗法の重みを与えることで、位置制約の誤差を小さくする必要があった。しかし、これには十分な数値の桁数が必要となるため、実装上、数値計算が破綻しやすくなる。一方、本研究の手法では、厳密制約を緩い制約の最小 2 乗式から分離しているために大きな重みを用いる必要がなく、計算を安定化することができたと考えられる。

4.2 計算時間

本研究の手法は、従来手法¹⁾³⁾と同様に 3 段階の処理で構成される。最初の準備段階では、緩い制約と厳密制約を記述した行列を組立てる。次の前処理段階では、QR 分解とコレスキー分解によって、制約を表現した行列を二つの行列の積に分解する。最後に、ユーザが指定したハンドル位置を基に、変形後の頂点座標をインタラクティブに計算する。前処理段階で既にコレスキー分解が行われているた

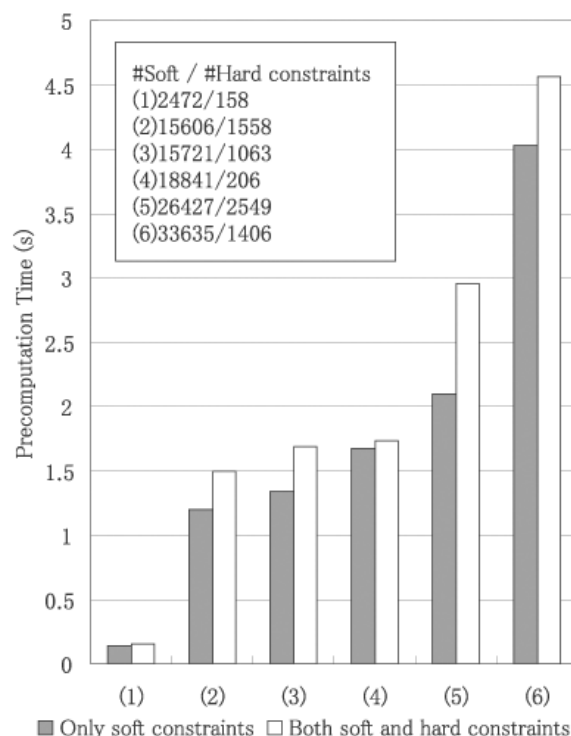


図 7 前処理の計算時間比較
Comparison of precomputation times.

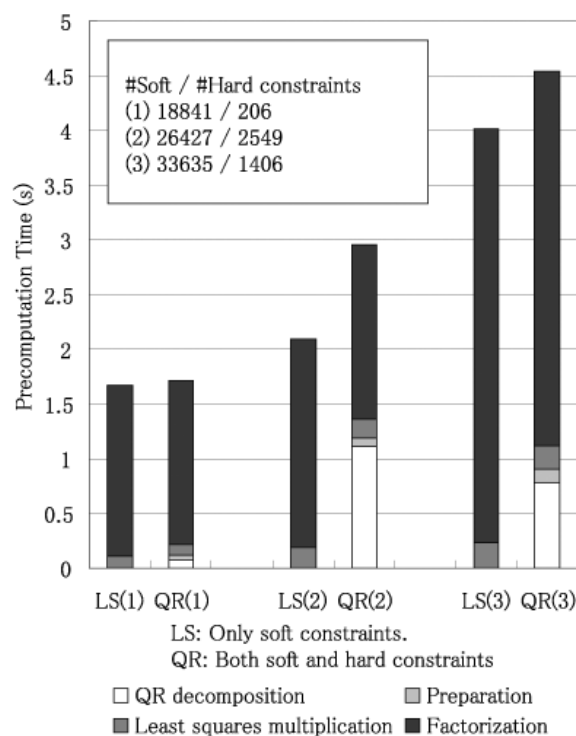


図 8 前処理時間の詳細
Details of precomputation times.

め、この段階での計算は非常に高速に行われる。本手法、及び従来手法においてボトルネックとなるのは前処理段階である。

図7は前処理段階の計算時間を頂点数の異なる6通りの例で比較した結果である。グラフの左側は緩い制約のみを用いた場合の計算時間で、右側は本手法を用いた場合の計算時間である。厳密制約と緩い制約を用いた本研究の手法は従来の緩い制約のみの手法と同程度に高速であるという結果が得られた。

図8は前処理段階の計算時間の内訳を頂点数の異なる3通りの例で比較した結果である。グラフの左側は緩い制約のみを用いた場合の計算時間で、右側は本手法を用いた場合の計算時間である。また、Preparationは $R_1^T y = d$ と $c - AQ_1 y$ および AQ_2 を、Least squares multiplicationは $(AQ_2)^T$ と AQ_2 の積を、Factorizationは $(AQ_2)^T AQ_2$ のコレスキー分解を計算するのに要した時間である。厳密制約を用いた場合、緩い制約のみの場合よりもQR分解の計算時間を要するが、厳密制約を分離した分最小2乗法の計算が高速化されるため、全体ではそれほど計算時間の増加は起こらないことがわかる。

5. む す び

本研究の結果、曲面の平均曲率を保存した形状変形に関して、以下の結論が得られた。

- QR分解を用いた制約付き最小2乗法によって、厳密制約と緩い制約の両立を実現した。
- QR分解を用いることで、過剰制約と計算不安定性を解消した。

以上のことから、厳密制約と緩い制約を両立するメッシュ変形フレームワークを実現できたといえる。

今後、本手法を発展させ、より複雑な制約の表現や、より直感的なインタフェースによる変形を実現する可能性を検討していく予定である。また、疎行列の解法についても、計算の安定性と高速性を両立させる方法について検討していきたいと考えている。

〔文 献〕

- 1) M. Botsch, and L. Kobbelt : "An intuitive framework for real-time freeform modeling," ACM Trans. Graphics, 23(3), pp. 630-634 (2004)
- 2) M. Botsch, and L. Kobbelt : "A remeshing approach to multiresolution modeling," Symposium on Geometry Processing, pp.189-196 (2004)
- 3) O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H. Seidel : "Laplacian surface editing," Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 175-184 (2004)
- 4) Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H. Shum : "Mesh editing with poisson-based gradient field manipulation," ACM Trans. Graphics, 23(3), pp.644-651 (2004)
- 5) M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr : "Discrete differential-geometry operators for triangulated 2-manifolds," Visualization and Mathematics III, pp.35-57 (2003)
- 6) S. Toledo, D. Chen, and V. Rotkin : "Taucs: A library of sparse linear solvers," <http://www.tau.ac.il/stoledo/taucs/>, (2003)
- 7) G. H. Golub and C. F. V. Loan : "Matrix Computations", 3rd



よしおか やすひる
吉岡 康博 2004年、東京大学工学部システム創成学科知能社会システムコース卒業。2006年、同大学院工学系研究科環境海洋工学専攻修士課程修了。メッシュモデリング、コンピュータグラフィックスに関する研究に従事。



ますだ ひろし
増田 宏 1985年、東京大学工学部精密機械工学科卒業。1987年、同大学院工学系研究科修士課程修了。同年日本アイ・ピー・エム(株)東京基礎研究所。1998年、東京大学人工物工学研究センター助教授、2003年より東京大学大学院工学系研究科助教授。形状モデリング、コンピュータグラフィックス、CADに関する研究に従事。工学博士。正会員。



ふるかわ よしゆき
古川 慈之 1998年、東京大学工学部船舶海洋工学科卒業。2003年、同大学院工学系研究科環境海洋工学専攻博士課程修了。同年(独)産業技術総合研究所ものづくり先端技術研究センターに入所、現在に至る。3次元形状処理、設計・製造支援アプリケーション用ソフトウェアプラットフォームに関する研究に従事。博士(工学)。