Vol.2011-CG-142 No.17
2011/2/9

対話型操作におけるメッシュ分割アルゴリズムの高速化

今 井 良 太[†1]　今 井 桂 子[†2]

3 次元の物体の形状を計算機上で表現する手法の 1 つとして，メッシュがある．特に表面メッシュは，物体の表面の形状を多角形の組合せで表現する．本論文で提案する手法は，この表面メッシュを前景と背景のような 2 つ以上の部分に分割するためのものである．多くの頂点からなるメッシュを容易に操作するために，スケッチと呼ばれる操作を導入する．これを対話的に繰り返すことで，複雑な形状をもつメッシュをユーザーの思い通りに分割することができる．本論文の目的は，Ji らによって提案された手法[1)] を改善し，1 回の入力に対する処理を高速化することで，より効率的な操作を実現することである。

A Fast Method on Interactive Mesh Segmentation

Ryota Imai[†1] and Keiko Imai[†2]

We present a fast method on interactive mesh segmentation. Ji et al. suggested a mesh segmentation method[1)] that enables users to draw “sketches” on the mesh to indicate the parts they want. Their method also allows users to input sketches interactively, so that they can see the output and add more sketches to improve the results. In this paper, we propose a speed-up algorithm for mesh segmentation based on the existing method.

†1 中央大学大学院 理工学研究科 情報工学専攻
Information and System Engineering Course, Graduate School of Science and Engineering, Chuo University
†2 中央大学 理工学部 情報工学科
Department of Information System and Engineering, Chuo University

1. はじめに

メッシュは，計算機上で 3 次元の物体の形状を表現する方法の 1 つである．本論文では，特に「表面メッシュ」と呼ばれるメッシュを扱う．これは，物体の表面の形状を多角形の組合せで表現するものであり，物体の内部の情報は持たない．表面メッシュの作成方法としては，計算機上のソフトウェアによって何もない状態から作り出したり，3D スキャナーを用いて現実に存在する物体の形状を読み取ったりすることで作成される．また，加工方法としては，分割や変形，および形状を維持しながらメッシュを再構築するリメッシングなどが挙げられる．

一般的に，メッシュを用いて物体の形状をより細かく表現するためには，大まかな形状を表現する場合に比べてより多くの頂点や面が必要となる．既に存在するメッシュを加工することを考えると，加工対象の頂点数が多くなるほど，処理時間や操作の手間は増加する傾向がある．例えば，メッシュ全体をリメッシングしたり，比較的単純な変形を行なう場合には，処理時間は増加するがユーザーの手間はほとんど変わらない．一方で，メッシュのある特定の部分のみを抽出したり，特定の凹凸の前後で分割するといった加工では，ユーザーが自身の思い通りの結果を得るために，何らかの操作によって加工に必要な情報を入力しなければならない．このような例では，メッシュの規模が大きくなるほどユーザーの手間は増加すると考えられる．

そこで本論文では，メッシュに対する加工方法の中でも，ユーザーの思い通りにメッシュを分割するための操作方法に着目し，容易な操作で分割すべき領域を指定するための手法を提案する．このような手法に対する既存研究として，Ji らによってスケッチを用いる手法が提案されている[1)]．本論文では，Ji らの手法を高速化することで，より効率的な操作を実現することを目的とする．

2. 関連研究

メッシュ分割に対して対話的な処理を行なう研究の 1 つとして，Ji らの研究[1)] が挙げられる．これは，マウスなどの入力デバイスとディスプレイを用いて，メッシュの領域分割を対話的に行なうものである．ユーザは，ディスプレイに表示された表面メッシュに対して，分割したい領域を大まかに示すような「スケッチ」を描画する．スケッチには 2 種類あり，それぞれ「前景」と「背景」を示すために用いられる．例えば，ティーカップを取っ手と本体に分割したい場合，取っ手に沿うように前景のスケッチを描き，次に本体を横切るように

背景のスケッチを描く．もし，分割結果がユーザーの意図と異なる場合は，適宜スケッチを追加して再実行することで，所望の結果に近づけることができる．

スケッチを用いるという観点では，Li らの研究[2] が先行している．これは，写真や絵画などの画像処理において，特定の領域とそれ以外の領域を分離するような操作を容易に行なう手法を提案したものである．例えば，人物を撮影した写真に対して，被写体の人物と背景を分離するような場合に有効である．この手法では，前景と背景を大まかなスケッチで指定することで領域を指定することができ，必要であれば，スケッチを繰り返すことでその結果を対話的に改善することができる．1) の手法はこの画像処理における手法をメッシュに応用したものである．

メッシュ分割にスケッチのような入力を用いる手法としては，3) も挙げられる．3) では，メッシュに対して単なる曲線ではなく，筆のような幅をもった曲線を入力することで，その曲線によって塗りつぶされた範囲内で分割する領域の境界を決定することができる．出力結果が意図と異なる場合は，曲線を部分的に書き換えることで結果を修正することができる．1) の手法とこの手法の違いは，1) が領域を指定するのに対し，3) は境界線の候補を指定するという点である．

3. 問題定義

対象とするメッシュは，以下の条件を満たすものとする．

- 3 角形メッシュである．
- 1 つの辺は高々 2 つの面と共有される．

入力と出力は以下のとおりである．

入力　前景と背景のスケッチに対応する頂点集合 S_F, S_B

出力　分割すべき領域を表す頂点集合 F, B

本論文では，ユーザーの入力から分割する領域を決定し，その結果を出力するまでの過程を問題として定義する．実際には，この入力と出力が対話型操作の 1 回に相当し，ユーザーが意図した結果が得られたと判断するまで複数回繰り返される．

ユーザーは，メッシュを「前景」と「背景」の 2 つの領域に分割するためにスケッチを入力する．このスケッチはマウスなどの入力装置によって画面に表示されたメッシュ上に描かれる軌跡であり，視点を移動しながら複数本描くことができる．スケッチには前景用と背景用の 2 種類があり，ユーザーはこれらを任意に切り替えながら入力することができる．

これらのスケッチから，画面上で対応する頂点を選び出し，頂点集合 S_F, S_B を得る．次に，メッシュ分割アルゴリズムによって各々の頂点が前景または背景のどちらに属するかを判定し，頂点集合 F, B を得る．各集合に含まれる頂点を色分けして画面上に表示することにより，ユーザーは自身の意図と分割結果を比較することができる．このとき，例えば前景とすべき部分が背景と判定されていれば，ユーザーはその付近に追加の前景用スケッチを描き，再度アルゴリズムを実行することができる．

4. 既存手法

ここでは，Ji ら[1] が提案したメッシュ分割アルゴリズムについて述べる．

Ji らのアルゴリズムは，スケッチの頂点集合を入力として region growing アルゴリズムを適用することで，メッシュを 2 種類の領域に分割するというものである．ここで重要なのは，2 点間の距離として feature sensitive metric（FSM と略す）を用いている点である．

この FSM は，2 点間の isophotic metric[4] を考慮することで，平坦なところでは小さく，湾曲しているところでは大きくなる特性をもつ．さらに，凹んでいる部分，すなわち曲率が負である部分では，正の曲率の場合と比べてより大きな値となるように定義されている．このことは，[5),6)] で示された minima rule に対応している．

図 1，**図 2** は，既存手法によってメッシュを分割した例である．どちらの図も，メッシュが 2 つの領域で色分けされていることがわかる．

次に，FSM と region growing アルゴリズムについて詳しく述べる．

4.1 Feature Sensitive Metric

Region growing でメッシュの形状的な特徴を考慮するために，feature sensitive metric を導入する．メッシュ上の 2 頂点 p, q を結ぶパス Γ について，FSM d_Γ は次のように定義される．

$$d_\Gamma(p,q) = \int_\Gamma ds + w \int_{\Gamma^*} ds^* + w^* \int_\Gamma f(k_{pq}) ds \tag{1}$$

第 1 項は，パスのユークリッド距離を表す．第 2 項は，p, q の法線ベクトルのガウス写像上の距離を表し，これら 2 項が isophotic metric に対応する．

第 3 項は，p から q への有向曲率 k_{pq} に基づく値である．有向曲率は文献 7) に基づいて式（2）で表すことができる．

$$k_{pq} = k_1 \cos^2\theta + k_2 \sin^2\theta \tag{2}$$

k_1, k_2 は主方向 T_1, T_2 に対応する主曲率を表す．θ は，q を p の接平面に射影した点を q' としたときの，T_1 と $\overrightarrow{pq'}$ の間の角度を表す．式（1）の第 3 項の関数 f は，minima rule[5),6)]

Vol.2011-CG-142 No.17
2011/2/9

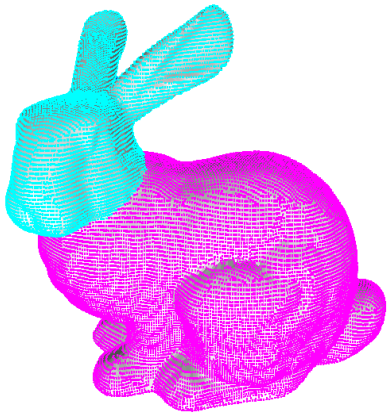

図 1 Stanford bunny モデル
Fig. 1 Stanford bunny model

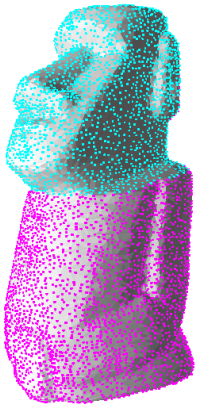

図 2 Moai モデル
Fig. 2 Moai model

に基づき，k_{pq} が負の場合により大きな値となるように定義する．

$$f(x) = \begin{cases} x & (x \geq 0) \\ g(|e^x - 1|) & (x < 0) \end{cases} \tag{3}$$

式（1）の w, w^* はそれぞれガウス写像と有向曲率の重みを表す．

4.2 Region Growing

Q を (v, m, d) という 3 つの変数からなる要素をもつ集合とする．各変数は次のような意味をもつ．

v ：N に含める候補となる頂点

m ：v が含まれるべき領域（F または B）

d ：m から v までの距離（feature sensitive metric）

スケッチの頂点集合 S_F, S_B を既知の頂点集合 F, B として，以下のようなアルゴリズムを実行する．

Region Growing アルゴリズム

Step 1 $N = F \cup B, Q = \phi$ とする．

Step 2 $^\forall v \in N$ について，最も近い頂点 $v' | v' \notin N$ を選び，(v', m, d) を Q に追加する．
ただし，m は v の領域を表し，d は v から v' への距離を表す．

Step 3 Q を d でソートし，d が最も小さい要素 $(v_{\min}, m_{\min}, d_{\min})$ を Q から取り出す．

Step 4 $v_{\min}$ の領域を $m_{\min}$ とし，$N = N \cup \{v_{\min}\}$ とする．

Step 5 すべての頂点が F または B に含まれるまで，Step 2 から Step 4 を繰り返す．

5. 提 案 手 法

既存手法では，2 回目以降の操作において，前回までの分割の結果を考慮していない．そこで提案手法では，メッシュ分割アルゴリズムを実行する際に，次回以降の分割でも同じ結果が得られる可能性の高い部分を検出し，分割結果とは別に記憶しておくことで，不必要な計算を省くことを考える．具体的には，通常のスケッチに加えて「拡張スケッチ」を導入することでこれを実現する．

本稿における拡張スケッチとは，ユーザーが入力したスケッチをより広範囲に拡張した頂点集合である．1 本のスケッチに対して 1 つの拡張スケッチが対応する．**図 3**，**図 4** に，入力したスケッチとその拡張スケッチの例を示す．ウサギの耳と胴の部分に描いた 2 本のスケッチから，それと同じ色の拡張スケッチが広がっていることがわかる．

実際には，拡張スケッチの作成は region growing アルゴリズムの実行中に行なわれるため，図 4 のように表示する必要はない．したがって，ユーザーはこれを意識することなく，既存手法と同様に操作することができる．

5.1 拡張スケッチの作成

拡張スケッチは，region growing アルゴリズムで選んだ頂点を同時に拡張スケッチとして記憶することで作成する．アルゴリズムの開始時から一定回数までの反復で選ばれた頂点を拡張スケッチに追加していくことで，スケッチから近い頂点の集合を得ることができる．例えば，図 4 は 3594 回の反復から得られたものである．頂点を記憶する回数は，メッシュの頂点数とスケッチの頂点数から決定する．

n 回目の分割時に作成された拡張スケッチは，5.3 で述べる削除条件を満たさない限り，$n+1$ 回目以降の分割で，ユーザーが入力したスケッチと同様に扱われる．すなわち，region growing アルゴリズムの初期条件として，S_F, S_B に加えて，前回までに作成された拡張スケッチの頂点を F, B に含めることで，アルゴリズムの反復回数を抑えることができる．

5.2 拡張スケッチを考慮した Region Growing アルゴリズム

拡張スケッチの集合を $\mathcal{E} = \{E_1, E_2, ..., E_s\}$ とする．ただし，s はスケッチの本数を表し，E_i は i 番目に描かれたスケッチに対応する拡張スケッチである．1 回目の分割では，各 E_i はすべて空集合であり，2 回目以降では，前回までの分割で追加された頂点をもつ．

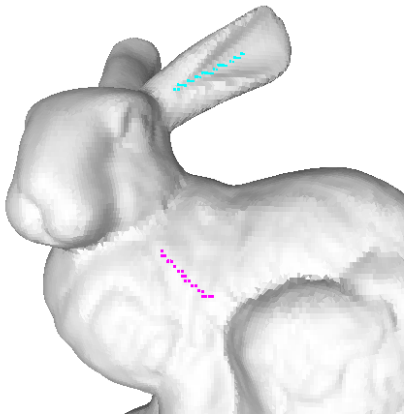

図 3 入力したスケッチ
Fig. 3 Input sketches

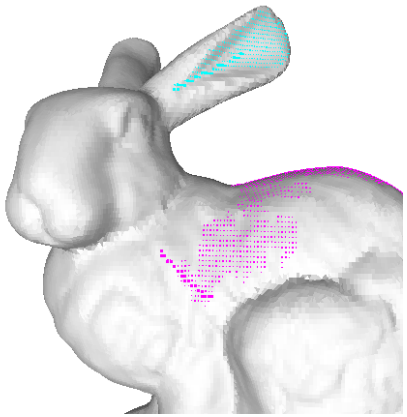

図 4 拡張スケッチ
Fig. 4 Extended sketches

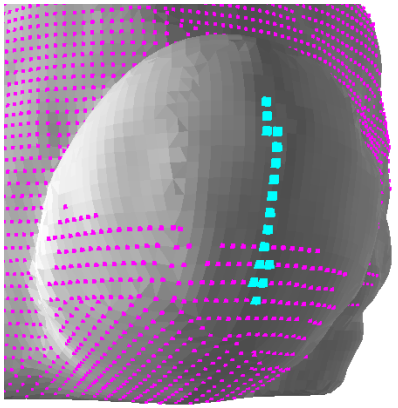

図 5 共有される頂点
Fig. 5 Shared Vertices

まず，既存手法と同様に，スケッチの頂点集合 S_F, S_B を F, B とする．次に，$^\forall E_i \in \mathcal{E}$ について，E_i が 5.3 の削除条件を満たすとき，$E_i = \phi$ とする．さらに，拡張スケッチとして追加する頂点の数を決定するために，式 (4) を用いて反復回数を決める定数 l を求める．ここで，$|V|$ はメッシュの頂点数を表し，α は正の実数である．本稿の計算機実験では，$\alpha = 2.0 \times 10^{-3}$ とした．

$$l = \alpha \cdot |S_F \cup S_B| \cdot |V| - \sum_{i=1}^{s} |E_i| \tag{4}$$

最後に，各 E_i について，対応するスケッチが前景である場合，$F = F \cup E_i$ とし，背景である場合，$B = B \cup E_i$ とする．

以上の処理をした上で，以下の改良した region growing アルゴリズムを実行する．Step 4 の派生元のスケッチとは，Step 2 で $v_{\min}$ を最近点とした頂点 v を見つけ，さらに v を最近点とした頂点を見つけるというように再帰的に探索していったとき，最後に到達する頂点が属するスケッチを表す．これは，動的計画法を用いることで容易に求めることができる．

改良 Region Growing アルゴリズム

Step 1 $N = F \cup B, Q = \phi$ とする．

Step 2 $^\forall v \in N$ について，最も近い頂点 $v' | v' \notin N$ を選び，(v', m, d) を Q に追加する．ただし，m は v の領域を表し，d は v から v' への距離を表す．

Step 3 Q を d でソートし，d が最も小さい要素 $(v_{\min}, m_{\min}, d_{\min})$ を Q から取り出す．

Step 4 反復回数が l 以下であれば，$v_{\min}$ を派生元のスケッチに対応する拡張スケッチに加える．

Step 5 $v_{\min}$ の領域を $m_{\min}$ とし，$N = N \cup \{v_{\min}\}$ とする．

Step 6 すべての頂点が F または B に含まれるまで，Step 2 から Step 4 を繰り返す．

5.3 拡張スケッチの削除条件

ある回の分割で追加したスケッチが，前回までに作成した拡張スケッチと頂点を共有し，かつ互いの種類が異なる場合，region growing アルゴリズムが正しい挙動にならない可能性がある．**図 5** は，赤で示した背景の拡張スケッチと，青で示した前景のスケッチが頂点を共有している例である．この場合，すべての拡張スケッチを無視して既存手法どおりのアルゴリズムを実行すれば正しい挙動を得られるが，処理時間を短縮することはできない．そこで，$^\forall E_i \in \mathcal{E}$ について，追加したスケッチと頂点を共有し，その種類が異なる場合のみ，$E_i = \phi$ として拡張スケッチを削除する．

6. 計算機実験

3 つのインスタンス A，B，C について，既存手法と提案手法の処理時間の比較を行なっ

Vol.2011-CG-142 No.17
2011/2/9

表 1 処理時間の比較
Table 1 Comparison of Processing Time

回数	A	B	C
1	1.004	1.000	1.003
2	0.959	0.953	0.960
3	0.921	0.899	0.916
4	0.850	0.879	0.838
5	0.765	0.821	0.764
6	0.684	0.961	0.651

た．各インスタンスは，Stanford bunny モデルに対して 6 回の分割を行なう際の入力スケッチを記録したものである．**表 1** は，各回の分割における提案手法の処理時間を既存手法の処理時間で割った値であり，値が小さいほど提案手法が高速であることを示す．

どのインスタンスの場合も，2 回目以降の処理時間が減少し，回を重ねるごとにさらに減少していく傾向が見られる．ただし，インスタンス B では 6 回目で処理時間が増加しており，A や C とは異なる傾向を示していることがわかる．これは，6 回目の操作で入力したスケッチが，1 回目に入力したスケッチに対応する拡張スケッチと頂点を共有し，この拡張スケッチが削除されたためであると考えられる．しかし，それ以外の拡張スケッチは削除されなかったため，既存手法より高速に実行できたと思われる．

一方，A と B では拡張スケッチの削除は発生していないが，両者の値の違いは入力するスケッチによって拡張スケッチの効果に差があることを示していると考えられる．

7. 結　　論

本論文では，対話型のメッシュ分割手法において，前回までの処理で得た情報を効率的に利用し，処理時間を減少させる手法を提案した．計算機実験では，既存手法と比較して，操作を繰り返すごとに処理時間が減少していく傾向があることを確認した．

しかし，一部の結果では処理時間が期待通りに減少しない場合もあり，より詳細な実験によってその原因や発生条件を検証する必要がある．また，拡張スケッチの作成方法についても，パラメータやアルゴリズムに検討の余地が残っている．今後は，多くのモデルやインスタンスを用いて実験を行い，問題点を洗い出していくことが必要である．

参 考 文 献

1) Ji, Z., Liu, L., Chen, Z. and Wang, G.: Easy Mesh Cutting, *Computer Graphic Forum (Proceedings of Eurographics)*, Vol.25, No.3, pp.283–291 (2006).
2) Li, Y., Sun, J., Tang, C. and Shum, H.: Lazy snapping, *ACM Transactions on Graphics*, Vol.23, No.3, pp.303–308 (2004).
3) Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S. and Dobkin, D.: Modeling by example, *ACM Transactions on Graphics*, Vol.23, No.3, pp.652–663 (2004).
4) Pottmann, H., Steiner, T., Hofer, M., Haider, C. and Hanbury, A.: The isophotic metric and its application to feature sensitive morphology on surfaces, *Computer Vision-ECCV 2004*, pp.18–23 (2004).
5) Hoffman, D. and Richards, W.: Parts of recognition, *Cognition*, Vol.18, No.1-3, pp.65–96 (1984).
6) Hoffman, D. and Singh, M.: Salience of visual parts, *Cognition*, Vol.63, No.1, pp. 29–78 (1997).
7) Taubin, G.: Estimating the tensor of curvature of a surface from a polyhedral approximation, *Proceedings of the Fifth International Conference on Computer Vision*, IEEE Computer Society, pp.902–907 (1995).