社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－CG－119 (3)
2005／5／20

ストローク入力によるメッシュモデル分割操作のための簡易実装法

三谷 純[†] 鈴木宏正[††]

コンピュータグラフィックス及びCADの分野において，近年ではユーザーインターフェースの改善の観点から，ユーザーのストローク入力によるモデル構築の手法が多く研究されている．この手法では，ストロークによる3Dモデルの切断は無くてはならない機能の一つである．ところでメッシュモデルの切断を従来のブール演算の手法で厳密に実装するには，演算誤差の考慮が必要で多大な労力を要する．本研究では，ストローク入力によるメッシュ分割操作を対象とし，得られる形状の精度よりも実装の容易さと切断後のメッシュの品質に注目した，新しい実装方法を提案する．提案する手法では，ストローク入力後にメッシュ頂点をストローク上に乗るように移動させ，3通りの三角形分割方法の組み合わせでメッシュモデルの切断を実現させる．

A method for cutting a mesh model by hand drawn stroke achieved by a simple implementation

JUN MITANI[†] and HIROMASA SUZUKI[††]

In the field of CG and CAD, the way to design 3D models using hand drawn strokes is well studied on view of improving the user interfaces in recent years. When using strokes to design 3D models, the function to cut a model by a stroke is necessary. For cutting a model, we can use an approach of using boolean operation for polygonal models that is well studied in past years. But implementing the boolean operation that works well with handling unavoidable numerical errors is very difficult. In this paper, we propose a new method for simple implementation of cut operation. We place more value on simplicity and quality of the mesh generated by a cutting than the accuracy. Firstly, we move vertices of a mesh that lay on near from a stroke to new place so that they ride on the stroke. Then we simplify the stroke so that all triangles are divided into less than two.

1. はじめに

計算機が広く普及した今日では，計算機を用いて意図した作業を効率的に行うためのインターフェースに関する研究が盛んに行われている．コンピュータグラフィックスやCADソフトのような3次元形状を計算機内に構築するアプリケーションにおいても，インターフェースの改善は大きな課題である．

近年では，ユーザがマウスまたはタブレットなどを用いて入力するストローク情報を元に，3次元形状を編集するためのインターフェースが多く提案されている．ストロークの形状を直接3次元形状に反映させるインターフェースは直感的であり，スケッチインターフェースとして広く研究されている[1)2)]．このようなインターフェースにおいて，ストロークの入力によって3次元形状を切断する操作は無くてはならない機能の一つである[3)]．ストロークによる3次元形状の切断は図1(a)(b)に示すように，ストローク曲線を投影面に垂直にスイープして得られる曲面によって3次元形状を切断することで実現できる．一般にストロークはマウスカーソルの軌跡を離散的にサンプリングした2次元平面上の点列で表現されるため，ストロークによる3次元形状の切断は，点列を結んで得られる折れ線をスイープした面によって3次元モデルを切断することになる．3次元モデルが三角形メッシュで表現されている場合，多面体のブール演算の手法によってこのような切断を実現できるが，最終的には個々の三角形をストロークによって切断する問題(図1(c))に帰着される．

三角形をストロークで分割することは難しい問題ではないが，演算時に発生する誤差により厳密な実装には困難が

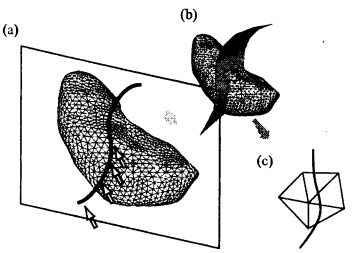

図1 ストロークによるモデルの切断
Fig. 1 Cutting operation using a hand drawn stroke.

† 筑波大学大学院 システム情報工学研究科
Department of Computer Science, Univ. of Tsukuba
†† 東京大学東京大学先端科学技術研究センター
Fine Digital Engineering, RCAST, The Univ. of Tokyo

伴う．切断処理を行ったモデルに対しても継続して変形処理を行うためには，位相情報の保持が必要なため，演算誤差によって位相が破綻しない頑健なアルゴリズムが要求される．また，切断後の三角形メッシュの品質を維持することも重要である．

本稿では，対象をストロークによる三角形メッシュモデルの切断に絞り，「そもそもストロークによる入力は厳密な形状構築には用いられないため，ストロークやモデルの形状を多少変更しても問題ない」という前提に立つことで，頑健で，切断後の三角形メッシュの品質が高く，さらに実装が容易なメッシュ切断アルゴリズムを提案する．

2. 対象とする形状と従来の手法の問題点

2.1 対象とする形状

本研究の動機は，人体の3Dデータを元に服の型紙を作成する際に，ユーザが意図したストロークで切開線を入力したいというものである．具体的には図2(a)に示すような三角形メッシュモデルに対し，ユーザーの入力したストロークを元に図2(b)に示すようなメッシュの切開を行うことを対象としている．ストロークによってメッシュモデル全体が切断される(モデルの裏側の面も切断される)インターフェースが必要とされることもあるが，図2ではスクリーンに投影したときに可視な面だけが切断の対象となっている．しかし，このことは本研究において本質的な差異は無い．

本研究で対象とする3次元形状は次のようなものである．

- 三角形の集合で表現される
- 境界を持つことを許した二多様体モデルとして表現される位相構造を内部に持つ

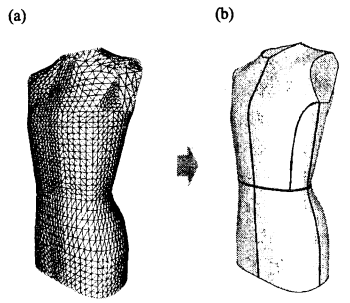

図2 対象とするストロークによる三角形メッシュモデルの切断
Fig. 2 Our target of cutting operation for trianglated model.

2.2 従来の手法の問題点

三角形メッシュモデルをストロークによって切断するには，ストロークを投影面に垂直にスイープして得られる曲面によって3次元形状を切断すればよい．これは，投影面上の個々の三角形をストロークによって切断したものを集約することで実現できる．

ストロークによって三角形メッシュを切断する場合，切断線にストロークの形状が忠実に反映されるようにするには，図3のように，ストロークが通る三角形それぞれについて，ストロークを構成する線分が稜線となるような三角形分割を行う必要がある．

図3 ストロークによる三角形の分割（○印はストロークと辺の交点）
Fig. 3 Dividing triangles by a stroke.

この方法では，まずストロークと三角形の交点を求める必要があるが，ストロークがメッシュの頂点近くを通る場合など演算誤差によって厳密に交点が求まらず，切断後の位相の構築に失敗することがある．また，特定の線分が稜線となるような三角形分割は，制約付きドロネー三角形分割（CDT: Constraint Delaunay Triangulation）によって実現できるが，この方法を用いて切断を行うと，元の三角形と比較して小さな三角形が多数発生する．ストロークによっては極端に扁平な三角形が生成されることもある．

つまり従来の手法には次の3つの問題点がある．

- 演算誤差による破綻の懸念がある
- 制約付きドロネー三角形分割を実装するコストが発生する
- 生成される三角形メッシュの品質がよくない

ここでの「メッシュの品質」とは，モデルを構成する三角形の大きさのばらつきと，極端に扁平な三角形の割合を指す．切断操作を行った後も継続して他の変形操作を行う際には，三角形の大きさは均一で，扁平な三角形は少ない方が好ましい．文献4)では，スケッチインターフェースで生成，変形したモデルのメッシュ構造を再度構築し直すことで，メッシュの品質を高める手法が提案されている．

また演算誤差による位相の破綻については，幾何情報よりも位相を優先して切断を行う手法が提案されている[5]．いずれも実装するには，制約付きドロネー三角形分割の実装以上に大変な困難が伴う．

ところで，ユーザーがマウスなどで入力するストロークは人の手によって生成されたマウスカーソルの軌跡であるため，再現性に乏しく，厳密な形状操作に適用することはできない．つまり，ストローク入力によるインターフェースが採られている操作には，厳密性は必要とされていないと言うことができる．たとえばIgarashiらのTeddy[1]に代表されるような，ぬいぐるみ形状においては，ストロークによって形状の概観が定まればよいのであって，ストロークの軌跡を忠実に再現する必要は無い．

さらに，三角形メッシュによるモデルの表現自体が，対象とする形状を近似したものであり，この三角形メッシュの細かさが，ユーザーから要求される近似の精度であると考えてよいものとすると，元のモデルを構成する三角形よ

りも細かい三角形を用いてストローク形状を反映させる必要性は小さい．そこで本研究では，ストロークによる切断操作によって発生する三角形分割は高々4 分割に収まるようにする．また，対象とする三角形メッシュの頂点位置を微小距離移動させ，さらにストロークの変形を行うことで，実装の簡略化と得られるメッシュの品質の向上を図る．

ここで，ユーザーが入力するストロークは以下の前提に基づくものとする

- 元の三角形メッシュによって表現されている程度の精度しか要求されない（ストロークが 1 つの三角形を複数回横切らなければ表現できないような形状の精度は要求されない）
- ストロークが交差しない

上記の前提のもと，本研究では高々3 通りの三角形の分割方法の組み合わせでストロークによるメッシュの切断を実現する手法を提案する．実装は極めて容易であり，また三角形の分割は 4 分割以内に収まるため，1 回の切断操作で細かい三角形が生成されにくいという利点がある．

3. 手法の詳細

本章では，メッシュのストロークによる切断を簡易に実装する手法の詳細を述べる．

3.1 用語と表記の定義

以降の説明で用いる用語と表記を次のように定義する．

ストローク ユーザーの入力するマウスカーソルの軌跡で，スクリーン上に配置された点列で表現される．

ストローク頂点 ストロークを構成する点．

ストローク線分 2 つのストローク頂点を端点に持つ線分．

モデル頂点 切断対象とする 3D モデルを構成する頂点．

モデル辺 切断対象とする 3D モデルを構成する辺．

モデル面 切断対象とする 3D モデルを構成する三角形．

E_i 識別子が i のモデル辺 (モデル辺には一意の識別子を割り当てる)．

V_j 識別子が j のモデル頂点 (モデル頂点には一意の識別子を割り当てる)．

3.2 モデル頂点の移動

予めストロークに近い位置にあるモデル頂点をストローク上に移動させ，ストローク上に乗っていることを示すフラグを立てることを行う．なお，頂点を切断線上に移動させる方法は，Biermann ら[6] が再分割曲面のブール演算の実装においても採用している方法である．ストローク頂点とモデル頂点が近い位置にある場合，交差判定が演算誤差により不安定となることがあるが，このように頂点を移動させて予めフラグを立てておくことで交差判定の誤りを防ぐことができる．このことの副次的な効果として，扁平な三角形が生成されることを防ぐこともできる．

ストローク頂点への移動

スクリーン座標での各モデル頂点から最も近いストローク頂点を探索し，その距離が閾値（メッシュを構成する辺の平均長に対する割合で指定）よりも小さい場合，モデル頂点をスクリーンに平行な平面上でストローク頂点に一致するように移動させる．また，ストローク頂点にはメッシュ頂点と一致していることを表すフラグを立てておく．

ストローク辺への移動

スクリーン座標での各モデル頂点から最も近いストローク辺を探索し，その距離が閾値（メッシュを構成する辺の平均長に対する割合で指定）よりも小さい場合，モデル頂点をスクリーンに平行な平面上で，ストローク辺に乗るように移動させる (図 4)．また，ストロークには新しいストローク頂点を追加し，そのストローク頂点にメッシュ頂点と一致していることを表すフラグを立てておく．

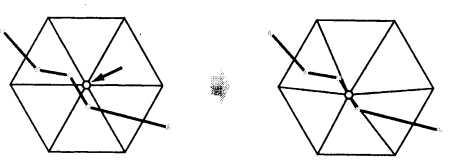

図 4 ストロークに近いモデル頂点の移動
Fig. 4 Moving a vertex of a model near from a stroke.

3.3 ストロークの正規化

本手法では，入力されたストロークの簡略化を行う．後で述べる 3 パターンの三角形分割の組み合わせでメッシュモデルの切断が実現するように簡略化することを「ストロークの正規化」と呼ぶこととする．

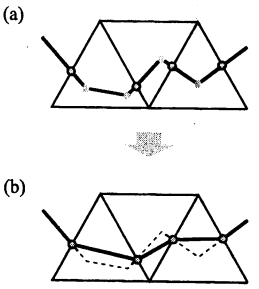

図 5 モデル面上の頂点を除いたストロークによる切断
Fig. 5 Cutting by a stroke that does not contain on-face-vertices.

メッシュモデルの三角形とストロークが図 5(a) のような関係にあるとき，図 5(b) のように，ストローク辺とモデル辺の交点に新しいストローク頂点を生成し，モデル頂点またはモデル辺上のストローク頂点だけを結ぶことで，ストロークの形状を簡略化できる．この場合，モデル面に乗るストロークの形状は考慮していないため，ストロークに忠実な切断線は実現できないが，モデル面の内部に新しい頂点が発生しないため，三角形分割を容易に実装することができる．個別の三角形に着目すれば，三角形を横切る

直線で分割されることになる．このようなストロークの変換をストロークの**第一正規化**と呼ぶこととする．第一正規化後のストローク頂点は必ずモデル頂点かモデル辺の上に乗っていることとなる（モデル頂点に乗っているストローク頂点は前節のモデル頂点の移動によってフラグが立てられている）．以降，ストローク頂点が乗っている要素（E_i または V_j）を順に格納したものを配列 A で表すものとする（例:$A = \{E_8, E_7, E_6, V_3, E_6, E_5\}$）．

ところで，第一正規化されたストロークでも図 6 のように，1 つの三角形を複数回横切ることを許容すると，三角形の分割方法が無数に存在し，全てのケースに対応できるアルゴリズムを実装するのは手間である．

図 6 1 つの三角形を複数回横切るストローク
Fig. 6 A stroke crossing a triangle multitime.

そこで，本手法では図 5(b) の例のように，三角形が 2 分割以上されないようにストロークを変更することを考える．これを**第二正規化**と呼ぶこととする．第一正規化されたストロークによって三角形を 2 分割する方法は図 7 に示す 2 通り（モデル頂点とモデル辺を通る場合と，異なる 2 つのモデル辺を通る場合）だけであり，どちらもストロークと三角形の交点が 2 つだけ存在する．また，2 つの交点が乗る要素は配列 A 内で隣接する．

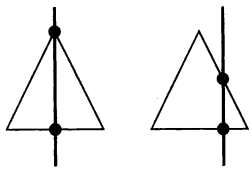

図 7 三角形と 2 点で交差するストローク
Fig. 7 A stroke crossing a triangle at two points.

そこで，同一のモデル面に含まれる要素（頂点または辺）が A の中に複数存在する場合，A 内で最初に現れる要素と A 内で最後に現れる要素の間に挟まれる要素列を削除する．(図 8)

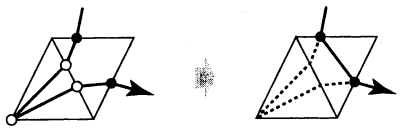

図 8 ストロークが 1 つの三角形と複数交差する場合
Fig. 8 A stroke crosses a triangle multitime.

このアルゴリズムを擬似コードで記したものが図 9 である．図中 A_i は，配列 A の $i-1$ 番目の要素を表し，`A.size` は A に含まれる要素の数，`A.remove(a, b)` は A に含まれる要素の A_a から A_b までを削除する操作を表す．なお，得られる結果は `foreach` ブロック内の処理を適用する三角形の順番に依存して異なったものになる．

```
foreach(Triangles t) {
    for(int i = 0; i < A.size; i++) {
        if(A_i ∈ t) break;
    }

    for(int j = A.size - 1; j >= 0; j--) {
        if(A_j ∈ t) break;
    }

    if(j - i > 1) {
        A.remove(i + 1, j - 1);
    }
}
```

図 9 ストロークの正規化を行うアルゴリズム
Fig. 9 Algorithm for stroke nomalization.

図 10 は，上記のアルゴリズムを適用することで，初期状態のストローク (a) が (b)〜(g) のステップを経て (h) に変換される様子を示したものである．図中の●は正規化後に A に残されることが確定した要素を，○は正規化処理によって削除される要素を表す．色の付いた三角形は，その時点で判定を行っている三角形を表す．正規化によって，ストロークが簡略化され，ストロークによる三角形の分割は 2 分割だけになることが確認できる．

3.4 鋭角を含むストロークへの対応

前節までに述べた方法により，メッシュモデルの切断を図 7 に示す 2 つのパターンで実現することができる．この方法は，ストロークが滑らかな曲線に近い場合は問題ないが，図 11 のように鋭角を含むストロークの場合，鋭角となる点が切断線に反映されないという問題がある．

この問題を解決するために，新たにモデル面上にストロークの端点を（多くても 1 つ）配置することを許容するものとする．この場合，端点を含む三角形とストロークの交点は図 12 に示す 2 通り存在する．ストロークの端点を用いてこの三角形を 3 分割すると，図 12(a) は図 13(c) に示すように，これ以上三角形分割をする必要がない．図 12(b) の場合は，3 分割した後で図 13(a) に示す 2 分割を行うことで，ストロークによる三角形分割が実現する．

つまり，新しく 3 分割の方法を 1 つ追加することで，端点が面上にあるストロークにも対応することができる．従ってストロークが鋭角を含む場合，鋭角となる点でストロークを分割し，それぞれのストロークに対して正規化を行えば，鋭角点を切断線に反映することができる．

以上をまとめると，本研究で提案する手法ではストロークによる三角形メッシュの切断を次のような手順で行う．

(1) ストローク近傍のモデル頂点を移動させる
(2) 鋭角点でストロークを分割する

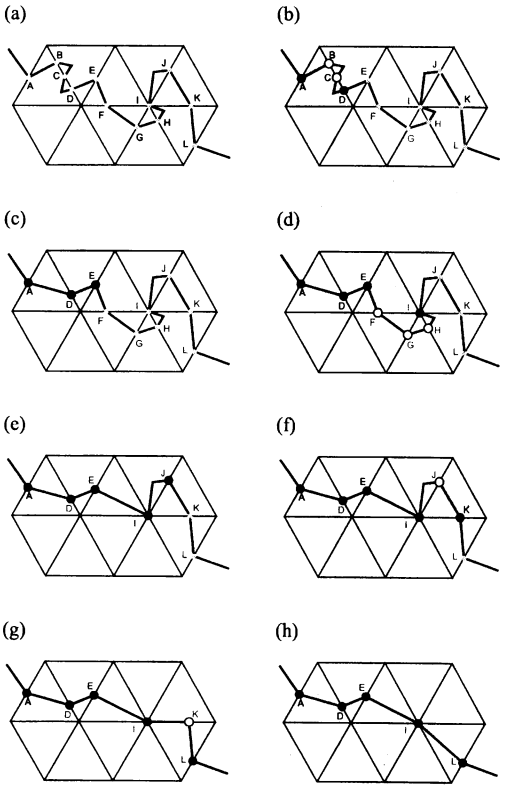

図 10　ストロークの正規化の例
Fig. 10　An example of stroke nomalization.

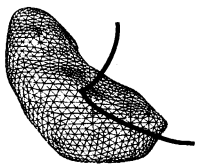

図 11　鋭角を含むストローク
Fig. 11　A stroke that contains a sharp angle.

(3)　分割された各ストロークについて次の処理を行う
　(a)　ストロークの端点を含む三角形を図 13(c) の方法で 3 分割する
　(b)　ストロークの正規化を行う
　(c)　ストロークと三角形の交わり方に応じて図 13 に示す分割を行う

4. 結　　果

本稿で提案する手法を C++言語を用いて WindowsXP が動作する PC 上に実装した．ユーザーの入力したストロークによってメッシュの切断を行った結果は図 14 および図 15 のようになった．

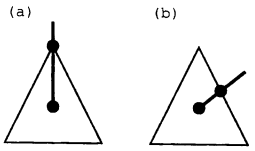

図 12　面上に端点を持つストロークと三角形の交差
Fig. 12　A stroke that has an end point on a triangle.

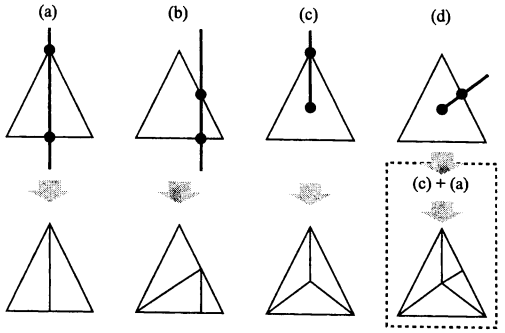

図 13　三角形の分割パターン
Fig. 13　Patterns of triangle division.

図 14(a) はユーザーが入力したストロークであり，(b) はストロークに近いメッシュ頂点を移動させたものである．(c) は第一正規化を (d) は第二正規化を行った様子であり，最終的には (e) に示す切断線を得ることができた．元のメッシュを構成する三角形の稜線を使用することで，少ない分割で切断が実現できている．

図 15 はストロークに鋭角な点がある場合の例である．(a) に示すユーザーが入力したストロークに対し，(b) の切断結果を得た．

5. 結　　論

本稿では，ストロークによるメッシュモデルの切断を容易に実装する方法を提案した．実装すべき三角形分割のパターンが 3 通りだけで済むため，切断後も位相情報を保つのが容易である．また，予め互いに近い位置にあるストロークとモデル頂点を移動させることを行うので，ストロークとメッシュ辺の交点の算出も頑健に行うことができた．実装した結果より，少ない分割数でストロークの形状に近い切断線の生成を実現することができた．

6. 展　　望

本研究で実装した手法は，メッシュモデルの表面に切開線を入れることを行ったが，モデルを 2 つに切断し，切断した後も閉じた物体を構成するようにする場合には，切り口が穴埋めされる必要があるため，用途に応じてはさらなる実装が必要な場合がある．また，今後は既存の手法との

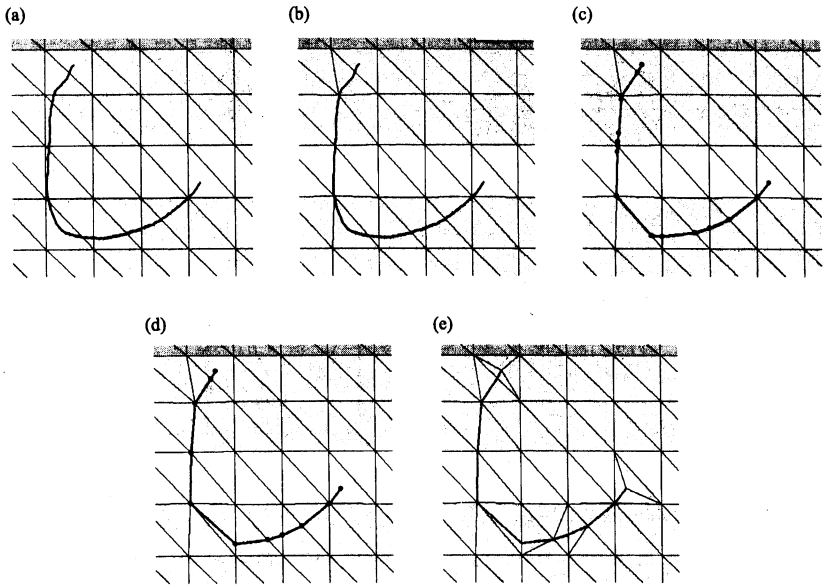

図 14 本手法を適用した結果
Fig. 14 Result of our implementation.

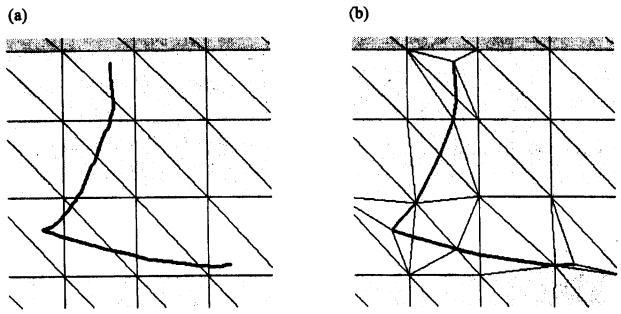

図 15 鋭角な点を含むストロークによる切断結果
Fig. 15 Result with a stroke that contains a sharp angle.

比較や，本手法の頑健性について，定量的な評価を行う必要がある．

参 考 文 献

1) Igarashi, T., Matsuoka, S. and Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design, *CM SIGGRAPH'99*, pp. 409–416 (1999).
2) 松田浩一，近藤邦雄: 3 次元形状入力のためのスケッチインタプリタシステム，第 14 回 NICOGRAPH/MULTIMEDIA 論文コンテスト，マルチメディアコンテンツ復興協会, pp. 17–25 (1998).
3) 大和田茂, 赤保谷鮎美, Nielsen, F., 楠房子, 五十嵐健夫: 切る, WISS 第 12 回インタラクティブシステムとソフトウェアに関するワークショップ, pp. 1–4 (2004).
4) Igarashi, T. and Hughes, J. F.: Smooth Meshes for Sketch-based Freeform Modeling, *ACM Symposium on Interactive 3D Graphics*, pp. 139–142 (2003).
5) 乾正知, 寺門宏明: 多面体の頑健な切断演算アルゴリズム, 情報処理学会論文誌, Vol. 38, No. 8, pp. 283–291 (1997).
6) Biermann, H., Kristjansson, D. and Zorin, D.: Approximate Boolean operations on free-form solids, *SIGGRAPH*, pp. 185–194 (2001).