3 方向デクセルモデルとポリゴンモデルの相互変換技術*

乾 正 知** 櫻 井 貴*** 梅 津 信 幸†

Data Conversion Technology between Triple Dixel Model and Polygonal Model

Masatomo INUI, Takashi SAKURAI and Nobuyuki UMEZU

In the triple dixel model, an object shape is represented with 3 groups of axis aligned segments named X-dixels, Y-dixels, and Z-dixels. In this paper, the authors propose a data conversion method from a conventional polygonal model such as STL to a triple dixel model, and vice versa. In this method, the following technology is employed; (1) Data conversion from a polygonal model to a triple dixel model is realized by multiple applications of “depth peeling”. This process can be accelerated by using the depth buffer and stencil buffer mechanisms of GPU. (2) Data conversion from a triple dixel model to a polygonal model is realized by gathering and classifying polygons attached to the end point of neighboring dixels. An experimental conversion program is implemented and some computational experiments are performed. Our program can convert a polygonal model with 3,000,000 polygons to a triple dixel model with 1024 x 1024 resolution grids in 8 seconds. Reverse conversion from a triple dixel model to a polygonal model is realized in a second.

Key words: triple dixel model, data conversion, STL, GPU

1. は じ め に

コンピュータメモリの大容量化と低価格化，オペレーティングシステムの 64 ビット化，そして GPU（Graphics Processing Unit）などの高並列なプロセッサの普及により，3 次元立体を微小なポリゴンや立方体群を用いて近似的に表現する離散的なモデリング手法が，図形処理の分野で普及し始めている．著者らは，このようなモデリング手法の一つである 3 方向デクセルモデル¹⁾に注目し，これと GPU による並列処理を組み合わせることで，機械製造に関する様々な処理を高速・高精度に実行する技術を開発している．

3 方向デクセルモデルは，X 軸方向，Y 軸方向，Z 軸方向の 3 つのデクセルモデル²⁾を組み合わせたものであり，切削加工シミュレーションなどで効果を発揮するモデリング手法である^{3),4)}．既に商用 CAD システムが広く普及していること，そしてポリゴンに基づく図形表示や解析技術が一般化している現状を考えると，本技術を本格的に普及させるためには，(1) CAD により作成された形状モデルを 3 方向デクセルモデルへ変換し，加工シミュレーションなどに取り込む機能，(2) 加工シミュレーション後に，加工結果などを表す 3 方向デクセルモデルをポリゴンモデルに変換する機能，の二つが必要となる（図 1 参照）．

ほとんどの CAD システムは，立体モデルの表面を微小な三角形の集合に分解し，STL や OBJ などのポリゴンモデルとして出力する機能を備えている．そこで本研究では，以下に示すような，ポリゴンモデルと 3 方向デクセルモデルの相互変換技術を実現したので報告する；

(1) ポリゴンモデルから 3 方向デクセルモデルへの変換：深度剥離（depth peeling）と呼ばれるコンピュータグラフィックス技術を利用して，GPU により高速に変換する手法を実現した．本手法は，ポリゴンに微小な重なり合いや

Fig. 1 Data conversion between a polygonal model and a triple dixel model for the geometric milling simulation

隙間が生じていても，適切に変換できる．

(2) 3 方向デクセルモデルからポリゴンモデルへの変換：

(1) の逆変換であり，変換結果として微小な三角形ポリゴンの集合が出力される．X 軸方向，Y 軸方向，Z 軸方向の各デクセルモデルを別々にポリゴン化し，それらのうちで十分な精度を有するポリゴンを選択することで変換を実現した．この手法は非常に高速だが，ポリゴン間に多くの重なり合いが生じることがある．

次章では，3 方向デクセルモデルの概要を紹介する．また 3 方向デクセルモデルとポリゴンモデル間のデータ変換に関する過去の研究について簡単に述べる．第 3 章では，ポリゴンモデルから 3 方向デクセルモデルへの変換技術について説明する．第 4 章では，その逆変換である 3 方向デクセルモデルからポリゴンモデルへの変換手法について述べる．第 5 章では，実装したプログラムを用いて，幾つかの複雑な形状についてデータ変換実験を行った結果について説明する．

2. 3 方向デクセルモデル

3 方向デクセルモデルや，これと同等の立体表現手法は，既に多くの研究者から提案されている．製造支援の分野では，藤尾らの研究³⁾や Leu らの研究⁴⁾が知られている．また

* 原稿受付 平成 21 年 5 月 29 日

** 正 会 員 茨城大学（茨城県日立市中成沢 4・12・1）

*** 学生会員 茨城大学

† 茨城大学

Fig. 2 Definition of dixel model and triple dixel model

Fig. 3 Comparison between a dixel model and a triple dixel model

コンピュータグラフィックスの分野では、3方向デクセルモデルと同じ手法を、距離場表現 (Distance field representation) という名称で利用している例がある⁵⁾。

2.1 デクセルと3方向デクセルモデル

立体形状の離散的な表現手法の一つに、デクセルモデルがある²⁾。デクセルでは、図2(a)に示すように、立体形状を垂直な線分 (これをデクセルとよぶ) の集合で表現する。デクセルはXY平面上の直交格子に基づいて定義されており、必要に応じて線分を2本、3本と継ぎ足すことができる。同一格子に基づいて定義されたデクセルは、順にポイントで結ばれている。この仕組みにより、デクセルは横に挟まれた形状も自由に表現できる。

デクセルモデルは、XY平面上の格子に基づいて、立体を細長い直方体状に切り分けたものと見なすことができる (図3(a)参照)。このとき各デクセルの端点は、直方体の縦方向の辺と曲面の交点に対応する。比較的平坦な曲面からなる形状の場合には、隣接する縦方向の辺上の交点間の距離が十分に小さいので、形状を高精度に表現できる。しかし急峻な壁形状の場合には、隣接する辺上の交点間の距離が非常に大きくなるため、デクセルモデルで表現した形状とオリジナルの形状の間に、大きな誤差が生じてしまう。

この問題を解消するために、XY平面上の格子に基づく通

Fig. 4 Conversion from a polygonal model to a dixel model

常のデクセル (これをZ軸方向デクセルとよぶ) の他に、図2(b)に示すように、YZ平面上の格子に基づくX軸方向デクセルや、ZX平面上の格子に基づくY軸方向デクセルも用意する。立体形状をこれら3つのデクセルモデルの組み合わせで表現する手法を、3方向デクセルモデル¹⁾とよぶ。

デクセル定義の元となる3つの直交格子の解像度や配置を揃えると、3つの格子により空間を小さな立方体の集合に分割できる。3方向デクセルモデルは、立体の表面をこれらの立方体で切り分けたものと見なせる。図3(b)に示すように、どのような向きの曲面でも、立方体の辺上の交点間の距離は一辺の長さ程度なので、3方向デクセルモデルでは、どのような形状もほぼ均一の精度で表現できる。

2.2 ポリゴンモデルと3方向デクセルモデルのデータ変換

ポリゴンモデルなどの境界表現 (B-reps) による立体モデルと3方向デクセルモデル間のデータ変換に関する研究は、過去に幾つか行われている。

3方向デクセルモデルは、異なる3方向についてのデクセルモデルなので、ポリゴンモデルからある1方向のデクセルモデルへの変換を考えれば十分である。図4にデータ変換の基本処理を示す¹⁾。用意された格子上的各点から、格子に垂直な方向に半直線をのぼし、これとポリゴンモデルの全交点を得る。次に各交点において半直線を分割し、得られた線分群のうち立体の内側に存在するもの (図中の実線部) を選択する。これらを順にポイントで結ぶと、対応するデクセルモデルが得られる。高精度なデクセルモデルの場合には、100万を超す数の格子点についてこの計算を繰り返すことになるので、処理に膨大な時間を要してしまう。

3方向デクセルモデルでは、図3(b)に示すように空間を立方体で分割することになるので、立体を微小な立方体の集合で表現するボクセル (Voxel) モデルと類似性を有する。ポリゴンモデルのボクセル化については、GPUを利用した高速な手法が幾つか提案されている^{6)~8)}。ボクセル化では、ポリゴンモデルに対する微小な立方体の内外判定が主要な処理となるため、論文^{6),7)}の成果はそのままでは3方向デクセルモデルへの変換に適用できない。論文⁸⁾では、本研究と同様に深度剥離を用いて、ポリゴンモデルのボクセル化を実現している。その手法には、3章で述べるわれわれの技術と類似する点があるが、ポリゴンに重なりや隙間が生じている場合を考慮していない。

3方向デクセルからポリゴンモデルへのデータ変換につ

Fig. 5 Possible inconsistency in the in-out classification of a vertex of a cube in a triple dixel model

いては、マーチングキューブ法⁹⁾を利用した手法が知られている^{1,4,5)}。先に述べたように、3 方向デクセルモデルは立体を微小な立方体で分割したものと等価である。そこで図 5(a)に示すように、各立方体について、その 8 個の頂点が立体の内部か外部かを 3 方向デクセルモデルに基づいて分類する。そして分類結果に基づいて立方体内部にポリゴンを与えれば、3 方向デクセルモデルをポリゴン群へ変換できる。

この手法は、デクセルの端点の座標が、X 軸方向、Y 軸方向、Z 軸方向の全てのデクセルにおいて整合的であることに基いている。しかし実際には、デクセルの端点には、その座標を計算する際の数値誤差に起因する位置ずれが生じやすく、しばしば相互に矛盾を生じてしまう。図 5(b)と(c)にその一例を示す。この例では、赤いデクセルと緑のデクセルの端点に、数値誤差による不整合が生じている。そのため立方体の右端の頂点は、(b)では横向き赤いデクセルの外側なので立体の「外部」と判定されるが、(c)では上向きの緑のデクセルの内側なので立体の「内部」と判定されるため、適切な分類ができない¹⁰⁾。モデルを高精度化するために格子を細かくすればするほど、この種の問題が頻発してしまう。

3. ポリゴンモデルから 3 方向デクセルモデルへの変換

本章では、深度剥離¹¹⁾の利用によるポリゴンモデルから Z 軸方向デクセルモデルへのデータ変換について説明する。他の軸方向のデクセルモデルへの変換も同様の手法で実現できるので、以下の処理を繰り返せば、任意のポリゴンモデルを 3 方向デクセルモデルへ変換できる。

3.1 深度剥離による変換

ポリゴンモデルを真下から Z 軸の正方向へ見上げた画像を、平行投影を用いて生成することを考える。その際に、フレームバッファのピクセルの格子を、デクセルモデルの基本となる XY 平面上の格子と一致するように与える。GPU のデプスバッファ機能を用いると、フレームバッファのピクセルの各格子について、基準面からポリゴンモデルの一番手前側の表面までの距離（これをデプス値とよぶ）が計算され、ピクセルの対応するデプスバッファの各格子に格納される。得られた値に基準面の高さを加えることで、図 6(a)中に 1 と付した点群の Z 座標値を得ることができる。

得られたデプス値を、ステンシルバッファと呼ばれるメモリに格納した上で、再度ポリゴンモデルの画像を平行投影を用い

Fig. 6 Depth peeling of a polygonal model. (a) depth data after the initial drawing, (b) depth data after the first peeling, (c) depth data after the second peeling

Fig. 7 False ordering of intersection points between vertical rays and a polygonal model because of overlapping faces

Fig. 8 False ordering of intersection points between vertical rays and a polygonal model because of small gaps between polygons

て描画する。その際に、ステンシルバッファに記録されているデプス値よりも遠くの画像のみを描画するように命じることで、図中に 1 と付した点を無視した画像がフレームバッファに描画され、同時にデプスバッファに図中に 2 と付した点までの距離が格納される（同図(b)参照）。同じ処理を再度行くと、今度は 3 と付した点までの距離が格子ごとに得られる（同図(c)参照）。このようにして、視点側から 1 枚ずつ皮を剥ぐようにしてポリゴンを一部消去した画像を描く手法を深度剥離とよぶ。

上述の処理を行った後、各格子について；

(1) $i=0$ から順に、奇数 $(2i+1)$ 番目の点と、偶数 $(2i+2)$ 番目の点を線分で結ぶ。

(2) $i=0$ から順に、偶数 $(2i+2)$ 番目の点と、次の奇数 $(2i+3)$ 番目の点をポイントでつなぐ。

という二つの処理を繰り返す行することで、ポリゴンモデルを、対応する Z 軸方向デクセルモデルへ変換できる。

3.2 ポリゴンの重なり合いや隙間への対応

STL や OBJ 形式で記録されたポリゴンモデルには、ポリゴン

Fig. 9 Correct ordering of intersection points between vertical rays and a polygonal model by using culling method

Fig. 10 Incorrect intersection points caused by small gaps between polygons

とポリゴンが微かに重なり合うケースや、ポリゴン間に細い隙間が空いているケースが多数生じている。また深度剥離ではGPUの描画機能を用いるが、小さなポリゴンの描画に失敗して画像にピクセルサイズの穴があくこともある。上述した手法では、このような問題のあるデータや描画処理で希望する結果が得られない。Juの提案する技術¹⁰⁾などを用いて、事前にポリゴンの重なり合いや隙間を全て修復してから変換することも考えられるが、これらの手法では、重なり合いや隙間が一部のポリゴンでのみ生じている場合でも、モデルを構成する全ポリゴンを全く異なるポリゴン群に置き換えてしまう欠点がある。

図7に、一部においてポリゴンがわずかな距離で重なりあっている例を示す。この場合、深度剥離を用いて一番手前の点を無視して画像を描画しても、すぐ上側の次のポリゴンが描画されるので、赤い矢印で示した一部の交点に望ましくない番号付けがされてしまう。図8には、逆にポリゴン間に隙間が空いている場合の問題を示す。隙間を通して奥のポリゴンが描画されてしまうため、やはり赤い矢印で示した交点の番号付けに問題が生じている。

ポリゴンの重なり合いに起因する問題は、以下の手法で解決できる。深度剥離が正しく行われた場合、奇数番目の点が存在するポリゴンは、その法線方向が必ず視点側（下側）を向いている。同様に、偶数番目の点が存在するポリゴンは、その法線方向が視点とは逆側（上側）を向いていなくてはならない。そこでポリゴンを描画する際に、以下のようにカリング (Culling)

Fig. 11 Attachment of polygons at the end of three adjacent dexels

を行う。

(1) 奇数番目の点を取得するときには、ポリゴンが下向きのもので選択して描画する。図9(a)ではこのようなポリゴンを太線で示した。

(2) 偶数番目の点を取得するときには、ポリゴンが上向きのもので選択して描画する。図9(b)ではこのようなポリゴンを太線で示した。

このように選択的に描画すれば、一部にポリゴンの重なり合いが生じていても、重なっているポリゴンの最も視点側のものが点の決定に関与するので、図7に示したような問題は生じない。

カリングを導入しても、ポリゴンに隙間がある場合の問題は解消しない。図10(a)では奇数番目の交点生成の場合、そして(b)には偶数番目の交点生成の場合を示した。いずれの場合も、半直線が隙間をすり抜けてしまい、矢印で示したように、本来あるべき場所とはかけ離れたところに交点が生じられている。これらの交点は、隣接する格子の交点とは明らかに異なるZ座標値を有している。そこで本研究では、得られた交点を格子ごとに調べ、それが隣接する格子の交点とある閾値以上異なるZ座標値を有している場合には、その値は信頼できないものとして破棄し、代わりに隣接する格子の交点のZ座標値の平均値を与えることにした。

この手法を導入しても、穴や隙間が大きい場合には補正が困難だが、多くのポリゴンモデルでは隙間は非常に小さく狭いので、現状では良好な結果を得ている。この手法では、非常に狭い溝を隙間と誤解して埋めてしまう可能性もある。しかしデクセルモデルのような格子構造に基づくモデリング手法では、格子1個分程度の幅の溝形状は正確に表現することが本質的に難しいので、このような場合には、より高解像度の格子に基づくデクセルモデルへの変換を行うべきである。したがって本補正手法は、実用上は十分な性能を有していると考えられる。

4.3 方向デクセルモデルからポリゴンモデルへの変換

次に3方向デクセルモデルからポリゴンモデルへのデータ変換について説明する。図3(a)に示したように、Z軸方向のデクセルとオリジナルの形状がほぼ垂直に交差する場合には、デクセルは形状を正確に表現できる。一方、デクセルの方向とオリジナルな形状が平行に近い場合には、デクセルでは形状を正しく表現できない。しかし3方向デクセルの場合には、他のX軸方向やY軸方向のデクセルが、この部分の形状を正確に表現していると期待できる。

以上を踏まえて、本研究では、以下に示す2ステップからなるアルゴリズムで、3方向デクセルモデルをポリゴンモデルへ変換する。

Fig. 12 Selection of neighboring end points of dexels

Fig. 13 Selecting condition of neighboring end points of adjacent dexels

(1) X 軸方向, Y 軸方向, Z 軸方向の各デクセルモデルを, 別々にポリゴン化する。

(2) 各デクセルモデルについて, 十分高精度に形状を表していると考えられるポリゴンを選択し, それらを全て合わせることで最終的なポリゴンモデルを得る。

この手法により得られたポリゴンモデルには, 微小な重なり合いが生じている。そのため, ポリゴン間の隣接関係などの位相構造を持たない STL 形式の条件は満たすが, 境界表現法(B-reps)の立体モデルとして利用するには問題が多い。以下の節で各ステップでの処理について詳しく述べる。

4.1 デクセルモデルのポリゴン化

デクセルモデルはデクセルの列なので, 隣接する3つの格子についてデクセルの端点を適切に選択し, 図11に示すようにポリゴンを貼り付ければ, デクセルモデルをポリゴン化することができる。デクセルの端点を選ぶ際には, それらが空間的に隣り合っている必要があるが, その判定は容易ではない。この問題を, 図12を例に示す。この図ではデクセルの端点に, 下から順に番号が付けられている。図(a)の場合には, 同じ番号の端点が空間的に隣り合っているため, それらを結ぶことで容易にポリゴン化できる。しかし同図(b)の場合には, 図中に丸で囲んだ部分のように, 空間的に隣り合っているデクセルの端点に, 異なる番号が付けられているケースが発生している。

本研究では, 空間的に隣り合っているデクセルの端点を, 以下の手法で検出する。図13に示した隣接する2つの格子のデクセル列を, A, B とする。それらの和に当たるデクセル列 C と, 積にあたるデクセル列 D を計算し, 次にそれらの差 $C - D$ にあたるデクセル列 E を計算する。デクセル列 E は立体の境界に相当する範囲を表している。デクセル A とデクセル B の空間的に隣り合っている端点は, 同じ境界上の近傍の点のはずなので, デクセル列 E の同じデクセルに含まれていなくてはならない。したがって E の各デクセルに含まれる A と B の端点を調べることで, 空間的に隣り

Fig. 14 Some conversion result from OBJ data to triple dixel models

(a) Attached polygons to each dixel model

(b) Integration result of all polygons given in (a)

Fig. 15 A conversion result from a triple dixel model to a polygonal model

合っている点を容易に判別できる。この例では, E のデクセル列に基づいて, デクセル A の端点 1 と 2, デクセル A の端点 3 とデクセル B の端点 1, A の端点 4 と B の端点 2, A の端点 5 と B の端点 3 が空間的に隣り合っていることが分かる。

4.2 ポリゴンの選択

次に, X 軸方向, Y 軸方向, Z 軸方向の各デクセルモデルに貼り付けられたポリゴンから, 十分に高精度なものを選択する。オリジナルの形状の法線方向と各デクセルの軸方向が平行に近いほど, そのデクセルはオリジナルの形状を高精度に記録していると考えられる。そこで本研究では, 各軸方向のデクセルモデルについて, モデルに貼り付けられたポリゴンの法線方向とデクセルの軸方向のなす角度を調べ, その値が 60 度より小さいとき, 貼り付けられたポリゴンは十分に高精度と考え, それ以

Fig. 16 Another conversion result from a triple dixel model to a polygonal model

Table 1 Required conversion time from a polygonal model to a triple dixel model

Model name	Number of polygons	Required time (msec)
Part1	202,530	797
Part2	1,686,036	4,157
Part3	1,972,196	7,203
Part4	3,057,514	7,578

外のポリゴンを破棄することにした。最後に、3方向デクセルモデルの全てのデクセルについて残ったポリゴンを集め、変換結果のポリゴンモデルとして出力する。

5. 計算実験の結果

第3章と第4章で述べたアルゴリズムを Visual C++を用いてプログラム化し、提案したデータ変換技術の有効性を検証した。以下の実験では、CPU: Intel Core2Duo E8600 2.00GHz、メモリ: 3GB、GPU: NVIDIA GTX-280を搭載したPCを用いた。またX軸方向、Y軸方向、Z軸方向の各デクセルモデルを定義する格子の解像度が、全て1024×1024となるように設定した。最初に、OBJ形式で記録されたポリゴンモデルを、3方向デクセルへ変換する実験を行った。表1に4つのモデルの変換に要した時間を示した。これらの計算結果が示すように、開発したプログラムは十分に高速な変換を実現している。図14には、各ポリゴンモデルを3方向デクセルに変換したのち、第4章で述べた手法で再度ポリゴンモデルに変換した結果を表示した。複雑に入り組んだ形状も正しく変換されていることが分かる。

次に、3方向デクセルで記録されたモデルを、ポリゴンモデルへ変換する実験を行った。実験では、われわれが開発した3方向デクセルに基づく加工シミュレーションプログラムの処理結果をポリゴンに変換した。図15(a)には、Z軸方向、X軸方向、Y軸方向のデクセルモデルをそれぞれ別々にポリゴン化し、さらに精度の低いポリゴンを除去した結果を示した。各デクセルモデルにおいて、軸方向と平行なポリゴンが選択的に除去され黒く抜けていることが分かる。同図(b)には、これらのポリゴンを重ね書きした結果を示した。隙間や段差なくポリゴンが貼られていることが分かる。図16には、金型の仕上げ加工のシミュレーション結果を、ポリゴンモデルに変換したものを示す。通常のデクセル表現に基づくモデルが不得意とする縦壁にも、ポリゴンが滑らかに貼られている。変換に要した時間は0.26秒であった。変換後のポリゴン

数は、図15の例では2,207,568、また図16の例で2,162,698であった。

6. 結 言

本論文では、ポリゴンで表現された立体モデルと3方向デクセルモデル間のデータ変換技術について述べた。具体的には、以下の成果について報告した。

(1) 深度剥離に基づくポリゴンモデルからデクセルモデルへの変換手法を開発した。本手法ではGPUを利用することで高速な変換を実現している。ポリゴンに重なり合いや隙間が生じている場合でも、カリングや座標値の平均化を行うことで安定した変換が可能である。

(2) X軸方向、Y軸方向、Z軸方向の各デクセルモデルについて、境界を構成するデクセルの端点を抽出しポリゴン化する技術を開発した。さらに精度の低いポリゴンを除去し、残ったポリゴンを重ね合わせることで、3方向デクセルモデルをポリゴンモデルへ変換する手法を開発した。

実際に計算実験を行った結果、これらの技術が実用上十分な速度を有していることを確認した。

提案した変換技術には、解決すべき課題が残されている。

(1) 3方向デクセルモデルをポリゴンモデルへ変換すると、ポリゴンが多くの重なり合いを持つため、そのままでは境界表現の立体モデルとして利用することが難しい。

(2) 変換結果のポリゴン数が膨大なものになってしまう。今後はこれらの課題の解決に取り組んでいきたい。

謝 辞

本研究の実施に際して、(株)ソフィックスの宮本大資氏よりSTLデータの読み込み関数をご提供いただいた。また和歌山大学システム工学部の床井浩平准教授より、深度剥離のOpenGLによる実装法についてご教示いただいた。この場を借りて心から感謝する。

参 考 文 献

- 1) M.O.Benouamer and D.Michelucci: Bridging the Gap between CSG and Brep via a Triple Ray Representation, Proc. of Solid Modeling '97 (1997) 68.
- 2) T.Van Hook: Real-Time Shaded Milling Display, Proc. of ACM Siggraph (1986) 15.
- 3) 藤尾三紀夫: Boundary-Map データ構造に基づくNC加工シミュレータの開発, 沼津工業高等専門学校研究報告, 40, (2006) 311.
- 4) W.Zhang and M.C.Leu: NC Machining Simulation Based on Triple-Dixel Representation, Proc. 2008 International Symposium on Flexible Automation (2008).
- 5) L.P.Kobbelt, M.Botsch, U.Schwanecke, and H-P Seidel: Feature Sensitive Surface Extraction from Volume Data, Computer Graphics Proceedings, ACM Siggraph 2001 (2001) 57.
- 6) A.Karabassi, G.Papaioannou, and T.Theoharis: A depth buffer based voxelization algorithm, Journal of Graphics Tools, 4, 4 (1999) 5.
- 7) W.Li, Z.Fan, X.Wei, and A.Kaufman: GPU-Based Flow Simulation with Complex Boundaries, Technical Report, 031105, SUNY at Stony Brook (2003).
- 8) 原田隆宏, 越塚誠一: グラフィックスハードウェアを用いた高速なソリッドボクセイゼーション, Transaction of JSCES, Paper No.20060023 (2006).
- 9) W.E.Lorensen and H.E.Cline: Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm, Computer Graphics 21, 4 (1987) 163.
- 10) T.Ju: Robust Repair of Polygonal Models, ACM Transaction of Graphics, 23, 3 (2004) 888.
- 11) C.Everitt: Interactive order-independent transparency, Technical Report, NVIDIA (2001).