グラフィクスとCAD 83－6
（1996. 12. 20）

多方向画像からの立体再構成

藤本 直樹　小堀 研一　久津輪 敏郎
大阪工業大学

多方向画像から撮影した二次元画像を用いて、三次元立体を生成する方法がある。この方法は比較的簡単に三次元立体を構築することができるという利点がある。したがって、この方法は実用的ではあるが、処理に時間を要するという問題点も持っている。そこで、本稿では複数枚の二次元画像を使って高速に三次元立体を再構成する手法について報告する。この方法では二次元画像を多数の矩形に分割して、Octree データとすべての矩形で作られる投影角推との交差を判定して、目的の立体を得るものである。いくつかの実験を行って本手法の高速性を評価した。

Three-dimensional reconstruction using multiple two dimensional images

Naoki Fujimoto　Ken-ichi Kobori　Toshiro Kutsuwa
Osaka Institute of Technology

There is a method that reconstructs a three-dimensional object using several two-dimensional images taken from multiple viewpoints. This method has the advantage of easy generation of three-dimensional objects. However the method is practical, it has much cost for reconstructing the objects. This paper proposes a fast reconstruction method using two-dimensional images. At first, the two-dimensional images are divided into many rectangles. Secondly, the system examines intersections between Octree data and the pyramids by the projection of all the rectangles. As a result, we can get the three-dimensional object. Several experimental results show that this method reduces the computational time.

1. はじめに

立体を生成する方法の一つに多方向から物体を撮影した画像を用いる方法がある. この方法では求める立体の凹部に生成されない箇所が出てくる可能性があるなどの問題があるものの, 比較的簡単なシステムによって実現されるメリットがある. 例えば, 医用画像処理の分野においては, 人体のレントゲン写真など既存の画像データを用いてコンピュータ上に臓器や血管の立体を生成し, 診断を容易に行うことができる. また, カメラによって多方向から撮影した写真を用いることによってデザインの分野においては簡単なシステムによって三次元形状をコンピュータに構築することができる. そのため, 実用的な立体生成の手法であると言える. このときのデータの保持の仕方や計算手法には多くの方法が提案されている[1~4]. 求められる立体に境界表現を用いるものやOctree データ構造を用いて最終的な形状を求めるもの等がその例である. しかし, 2次元画像に複雑な処理する必要がある[1], 一定方向の画像にしか対応しない[2], また計算コストが高いため, 多くの画像を用いて立体を生成する場合やデーターベース化する際に何度もこの処理を繰り返す場合には多大な時間がかかるもの[3),4)]であった. そこで本稿では, 元となる2値画像を矩形領域に分割し, かつ得られる立体に Octree データ構造を用いることによって, 画像の形状による制約を受けず, あらゆる方向の画像を用いることができることを前提に, 高速に多方向画像から立体を生成する手法について提案する.

2. 立体生成の処理の流れ

本手法の処理の流れを図1に示す. 本手法では最初に画像を矩形に分割する. 次に, 矩形と視点で構成される四角錐と Octree の間で交差判定を行い, 交差していればOctant の分割を繰り返すことによって形状の内部に当たる領域を生成する. また, 複数画像によってできる複数の立体を合成することによって最終的な形状を得る立体合成処理を行う.

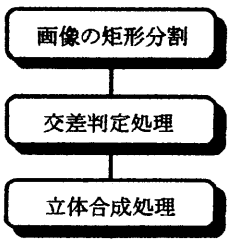


図1 処理の流れ

3. 画像の矩形領域分割

本研究では前処理として物体を撮影した2値画像である図2(a)を図2(b)のような矩形領域に分割する. これは, Octree データ構造との交差判定を高速に行うことを目的としており, 分割された矩形領域の数はできる限り少なくなるようにする. この時, 画像の凹部分は必ず分割されるため, 凹の部分からx軸に沿う方向, またはy軸に沿う方向に分割する[5]. 例えば, 図2(a)の①と②の凹部分からは図2(b)のaの矩形が得られる. 本研究では, 最初に縦方向に並ぶ凹部分の数と, 横方向に並ぶ凹部分の数を調べ, 生成される矩形領域の数が少なくなる方向を選択している. また, 付加情報として, 矩形と矩形が隣り合う場合には

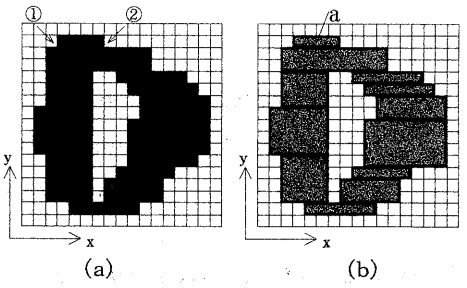


図2 2値画像と矩形領域分割

各々の矩形の稜線のどの範囲が隣接している部分かを調べて保持しておく. 例えば, 図3の①のような矩形の場合には, 矩形の下側の稜線の一部がもう一つの②の矩形と隣接しているので, ①の矩形の下側の稜線には隣接している範囲の x 値を付加して保持する. この情報は後述の交差

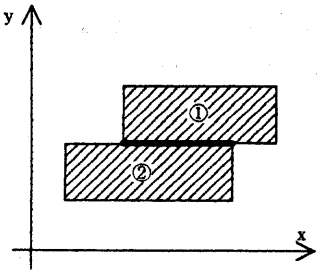


図3 隣接情報の取得

判定処理において矩形錐体と矩形錐体との間に存在するOctantの分割を防ぐために用いる.

4. Octreeデータ構造

Octreeデータ構造は基準となるCubeを再帰的に8分割することによって形状を表現するデータ構造である. この時の分割されていない Cube を Root Cube, また一つ一つの Cube を Octant と呼ぶ. Octant には 3 種類あり, 形状の内部を BLACK Octant, 形状の輪郭に交差するものを GRAY Octant, 形状の外部を WHITE Octant と呼ぶ. このデータ構造は空間分割モデルのデータを少ないメモリで持つことができることや形状の生成時には, 親 Octant の情報を子 Octant に継承できる等の利点をもつ. 本研究では, 親 Octant での交差判定に必要な情報を子 Octant に継承していくことによって, 交差判定の高速化を行った.

5. 矩形錐体とOctreeの交差判定

ここでは, 簡単のため一方向からの画像の矩形錐体を用いて立体を生成する方法について説明する. 交差判定手順の大略を図4に示し, 以下に各手順について詳述する.

・手順1

図5のようにコンピュータ上の三次元空間に 3 章で求めた複数の矩形と生成されるOctreeデータの元となる Root Cube を配置する. この時, 矩形データは 3 章より求めた2次元の頂点座標データをそのまま用い, z 値を0とすることで三次元上に配置できる.

・手順2

最初に視点からの Octant の可視面を判定し, 4角形又は6角形の輪郭多角形を構成する. 視点から見える面は, 子 Octant においても同一であるため分割が起こった際には, この情報は次々に子 Octant に継承していく. 次にその輪郭多角形の頂点を図6のように視点の方向を参照して xy 平面に投影する. 投影位置 $(xe, ye, 0)$ の計算は次式より得る.

$$xp = ze \times xo / (ze - zo)$$
$$yp = ze \times yo / (ze - zo)$$

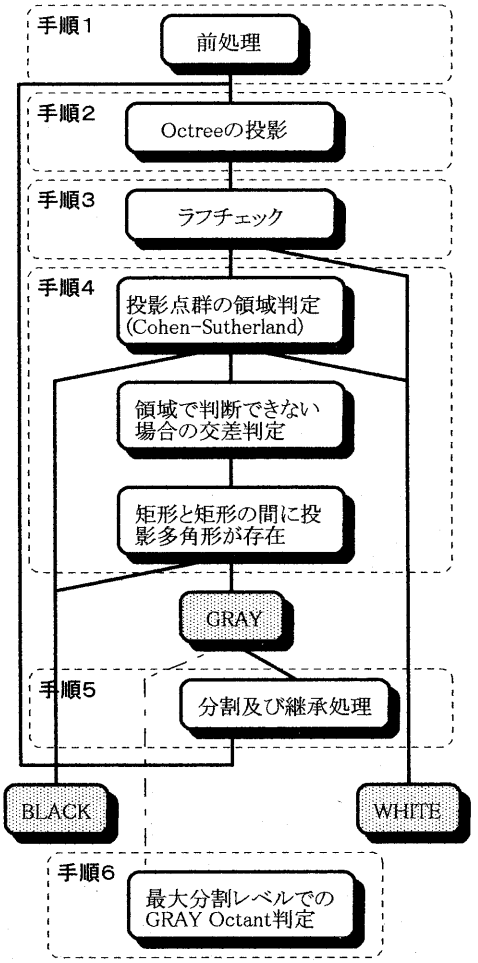


図4 交差判定のフローチャート

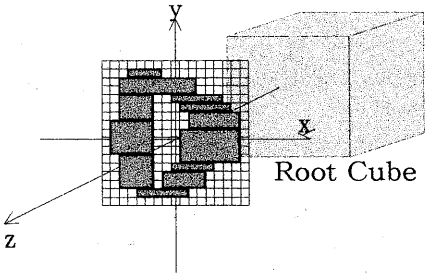


図5 画像矩形とRoot Cubeの配置

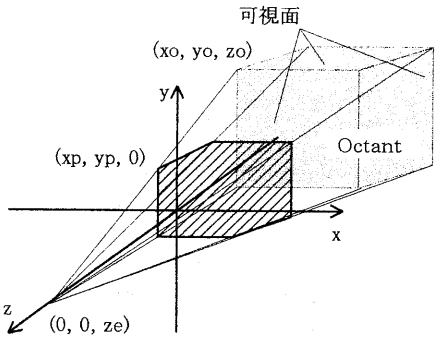


図6 Octree 頂点の xy 平面投影

・手順3

手順2で求めた Octant の投影多角形と画像を構成する矩形との交差を xy 平面上で2次元的に判定していく. ここでは最初にラフチェックを行うことで, 交差判定の高速化を行う. Octant を投影した投影多角形を長方形で近似し, 画像を構成する矩形の各々との交差を大小判定のみによって行う. 交差の可能性がある場合は交差判定手順4へ進み, 交差の可能性がない場合には Octant を WHITE として判定する.

・手順4

手順3のラフチェックによって交差の可能性があると判断された場合は手順2より求めた投影点群と矩形との位置関係を判定する. この時, Cohen-Sutherland[6]のアルゴリズムを応用する. 1つの矩形領域の4つの辺を延長して分けられる9個の領域のうち, どの領域に Octree の投影点が存在するかを判定する. なおこの判定は2次元上の大小関係だけから判断でき, 次の3通りに決定される.

1)図7(a)のように Octree の投影点がすべて内部であると判断された場合, Octant を BLACK と判定する.

2)図7(b)のように交差が一切なく, 投影多角形が完全に矩形の外部であると判断されれば WHITE Octant と判定する.

3)図7(c)のように Octree 投影多角形の頂点と頂点が矩形の内部と外側にまたがって存在するような場合で, 明らかに矩形と投影多角形の辺が交差するパターンは GRAY と判断する. さらに, 矩形と投影多角形の辺が交差する可能性のあるパターンであると判断される場合には, その交差可能性のある線分について矩形の1辺との交差判定を行う. ここで投影多角形の1辺でも交差であれば Octant は GRAY と判定される. また, 図 8(a)のような画像矩形群に対して投影多角形は本来は BLACK であるが, 先の判定によっては①②③の各々の矩形からGRAYと判定される. そこで, このような判定によって無駄な分割が行われないようにする. ここでは, 図 8(b)のように矩形の隣接情報と投影多角形の外包矩形を用いて判定を行う. 例えば①の矩形と投影多角形の判定を行う際は, ①の矩形の下側の辺の隣接情報の範囲に外包矩形が存在しているので BLACK と判定できる.

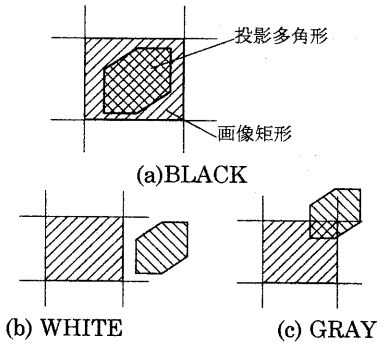


図7 存在領域による判定

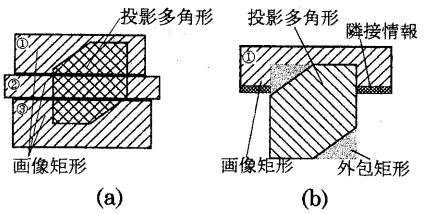


図8 矩形間の分割回避

・手順5

手順4についてある方向のすべての画像矩形に対して繰り返す. その方向の画像矩形群から GRAY と判断された Octant についてはあらかじめ設定した最大分割レベルを超えないように分割する. そして, 分割後の Octant の頂点位置を計算し, 手順2から4を再帰的に繰り返していく. 手順2での可視面の情報は分割された Octant に継承していく. また, 注目する Octant に交差する画像矩形のみ, 子 Octant に継承して交差を調べる.

・手順6

最大分割レベルまでの再帰計算の結果，GRAY Octant と判定された Octant については最終的な判定を行う．GRAY Octant について中心点を計算してこの点を xy 平面上に投影し，その点がいずれかの矩形の内部に入っているかどうかを座標値の大小によって判定する．入っていれば BLACK と判定し，そうでなければ WHITE と判定する．その結果，すべての Octant が BLACK と WHITE の2種類に分類される．

6. 立体の合成

5章では，1方向からの画像を用いた場合に限って説明したが，多方向画像によって生成される複数の Octree について論理積をとることによって立体を合成し形状を求める．図9では方向Aの画像より求められた①の物体と方向 B の画像より求められた②の物体の2つを合成し，最終的に③の立体形状内部を求める様子を表す．

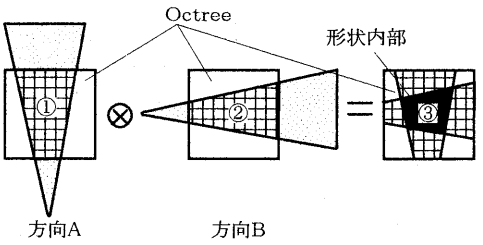


図 9　立体合成

7. 実験結果と考察

図10(a)のような椅子形状の多方向画像を5枚用意し，実験を行った．図10(b)は図10(a)のうちの1枚の画像から作成した矩形領域データの例である．立体を再構成した結果を図10(c)に示す．また図11(a)は同様の手順により生成したカメラ形状，図11(b)はテーブル形状である．次に，前処理として画像を凸多角形に分割し，凸錐体の境界表現を用いて Octree との交差判定を行う従来法との比較実験を行った．図12(a)は椅子形状，(b)はカメラ形状，(c)はテーブル形状の立体について交差判定にかかった時間を示す．実験に用いたのは Pentium-Pro200Mhz 搭載の DOS/V パソコンである．

従来法では前処理が高速に行えるために，最大分割レベルの低い4および5の範囲では処理に大きな差は見られなかった．しかし，最大分割レベルを上げていくにつれ，本手法の立体生成処理時間は短くなっている．最大分割レベルを上げると，立体生成処理時間に占める交差判定処理時間の割合が大きくなる．よって，交差判定の高速な本手法の立体生成時間が短くなった．最も時間を要すると思われる交差判定の処理の大部分を座標値の大小判定に帰着させ，乗算回数を減らした結果であると考えられる．

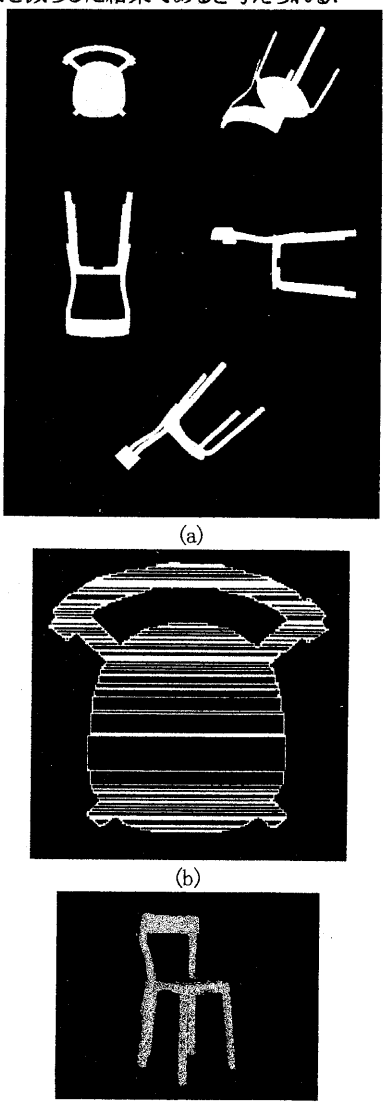
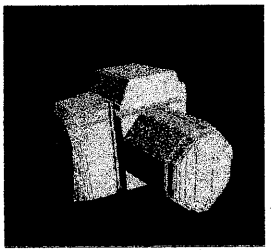
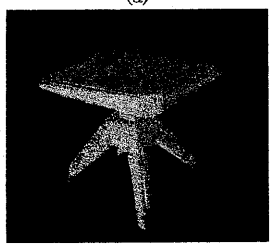


図10 実験に用いた物体画像と実験結果

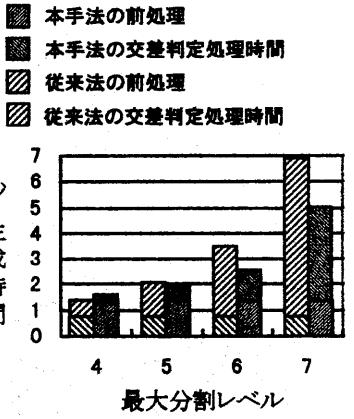


(a)

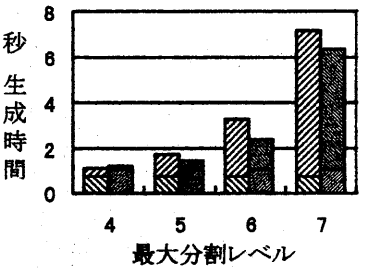


(b)

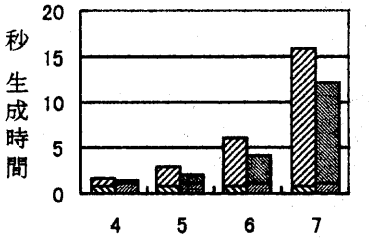
図11 立体再構成結果



(a)



(b)



(c)

図12 立体生成時間

8. おわりに

多方向からの2次元2値画像から矩形領域への分割という簡単な処理で，三次元立体を構成することができた．前処理の処理時間については従来法の処理時間の方が高速であった．しかし，交差判定部分に関しては，多くの処理を2次元の座標値の大小比較に帰着させることができ，立体生成を高速に行うことができた．

本手法では，画像の形状によって画像矩形の数が変わる．今後は画像矩形の数と生成時間の関係を明確にしてさらなる高速化を図りたい．

参考文献

1) 登尾,福田,有本:複数枚画像を用いて3次元物体を近似したOctreeを生成する一手法,情報処理学会論文誌,Vol.29,No.5,pp.178-189(1988).

2) Kim, Y. C. and Aggarwal, J. K. :Rectangular Parallelepiped Coding:A Volumetric Representation of Three-Dimensional Objects, IEEEJ. of Robotics and Automation, Vol. RA-2, No. 3, pp. 127-134(1986)

3) Hong, T-H. and Shneier, M. O:Describing a Robot's Workspace Using a Sequence of View from a Moving Camera, IEEE Trans, on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 6, pp. 721-726(1985)

4) 三宅,土井:立体形状の多面体近似システム,情報処理学会論誌,Vol.25,No.5,pp.745-754(1984).

5) 伊理,腰塚:計算幾何学と地理情報処理,共立出版,pp.100-105(1986)

6) 中前,西田:3次元コンピュータグラフィックス,昭晃堂,pp.104-106(1986)