複数枚の画像を用いて3次元物体を近似したオクトツリーを生成する一手法†

登尾啓史†† 福田尚三†† 有本卓††

複数枚の2次元画像を利用して3次元物体を近似したモデルを計算機内に生成することは，コンピュータ・ビジョン，コンピュータ・グラフィックス，ロボティクスなどの分野で必要な処理である．それゆえ多くの方法が提案されているが，多大な計算時間が必要である，画質の善し悪しがモデルの近似精度に大きく影響するといった問題点があり実際的な方法は確立されていない．そこで，本研究では錐体相貫法を利用した高速なモデル生成法を提案する．錐体相貫法では，まず，画像内で最も画質の善し悪しの影響を受けない物体の輪郭を多角形で近似し，それと投影中心で錐体を定義する．次に，すべての視点の錐体の共通領域を作成しそれを物体の近似領域とする．この処理を多面体モデルのうえで行うと，錐体間のすべての面の組み合わせにおいて交差を調べ共通領域を作成しなければならないが，オクトツリーという位置に関する階層構造をもつソリッドモデルのうえで行うと，そのような組み合わせ的な交差判定が不必要になりアルゴリズムが高速化される．また，提案した方法では投影面を自由に設定できるので任意の座標系でモデルを表現できる．そのうえ，透視投影を利用していること，画像の枚数に制限がないことなどから近似精度の良いモデルが作成できる．最後に，共通領域の表面積，物体の表現精度を意味するオクトツリーの作成レベルに関してアルゴリズムの計算量を評価しそれを実験で検討する．

1. まえがき

3次元モデルを用いて現実に存在する物体（以後実物体と記す）の近似領域を計算機内に再現することは，コンピュータ・ビジョン，コンピュータ・グラフィックス，ロボティクス等の広い分野で研究対象となっている．しかしながら，計算時間がかかる，画質の善し悪しがモデルの精度に大きく影響する，といった理由でいまだに実用的な方法は確立されていない．そこで，本研究では複数枚の実物体の投影像（カメラの投影面内の像）からその物体の近似領域を作成する高速な方法を提案する．

この方法では，錐体相貫法という原理（2章）を用いている[1]．これは，実物体の投影像の輪郭を直線近似した多角形（以後画像多角形と記す）と投影中心とで錐体を視点ごとに定義し，複数の視点に対して定義された錐体の共通領域を実物体の近似領域とみなすというものである．

この一連の処理で高速化できるのは，まず，画像多角形の作成であるが，既に高速な方法[2]が提案されているので，本研究では画像多角形は既に得られているものと仮定する．次に，共通領域の作成であるが，3次元モデルとして多面体を採用すると，それには少なくとも，n^2 と N^2 に比例した計算量が必要となる（n：視点数，N：画像多角形の辺数）．そこで，本研究では3次元モデルとしてオクトツリー[3),4)]というソリッドモデルを採用した（3章）．オクトツリーの位置に関する階層構造を錐体の交差判定に利用すると，共通領域の作成は，高々 n と N に比例した計算量となる（5章）．

錐体相貫法を用いたオクトツリーの作成法として，2つの代表的な報告がある．Kim らの方法[5)]は高速であるが，1) 実物体の姿勢に依存して座標系がきまるので，オクトツリーの座標系が自由に一例えば世界座標系で一設定できない，2) 透視投影で得られる実際の投影像を平行投影で扱ううえ，最大でも3投影像からの情報しか利用できないので近似精度がよくないといった問題点を持つ．Hong らの方法[6)]では，1) オクトツリーが任意の座標系で作成できる，2) 投影像を透視投影で扱ううえ投影像の数にも制限がないのでオクトツリーの近似精度が良いというように上記の問題点は解決されている．しかし，そこではオクトツリーと錐体の交差を投影面（2次元）上で判定しており，オクトツリーを投影面に透視変換しなければならない．これには膨大な回数の透視変換の計算が必要でアルゴリズムの処理が遅くなる．さらに，これら2つの方法は，視点ごとに錐体のオクトツリーを作成し，それらを重ね合わせて最終的なオクトツリーを作成するという考え方に基づいている．これでは，実物体に関して不必要な領域の処理が多くなり，計算時間の点で

† Construction of the Octree Approximating a Three-Dimensional Object by Using Multiple Views by HIROSHI NOBORIO, SHOZO FUKUDA and SUGURU ARIMOTO (Faculty of Engineering Science, Osaka University).

†† 大阪大学基礎工学部機械工学科

効率が悪くなる.

そこで我々は，錐体の交差判定をオクトツリー（3次元）上で直接行い（4，5章），かつすべての視点の錐体の間で同期をとることで共通領域のみをオクトツリーに変換するアルゴリズムを提案する．アルゴリズムは実物体に関して不必要な領域を処理しないので，Hong らと同様のオクトツリーが高速に作成できる.

2. 錐体相貫法

カメラで物体を撮影したとき，画像多角形と投影中心とで定義される錐体内に実物体は存在する．これより，複数の視点の錐体の共通領域が実物体の近似領域となる（図1）.

Stereo Vision[7]，Shape from Shading[8]，CT 画像による方法に比べ，錐体相貫法による形状入力には次のような利点がある.

（1） 投影像の輪郭は画像内で最もコントラストの強い情報なので，濃淡情報を利用する方法（Shape from Shading）などに比較して画質の善し悪しの影響をうけにくい.

（2） 輪郭からの画像多角形の作成は，多量の点の対応づけ（Stereo Vision）などに比較して高速におこなえる.

欠点は，投影像に現れない実物体の凹部分が入力できないことである.

3. モデルの定義

3.1 オクトツリー

オクトツリーは，図2のように全体空間を位置に関して階層的に表現したソリッドモデルである．オクトツリーでは，“外側” キューブ（物体が存在しない領域）を白ノード，“内側” キューブ（物体が占有する領域）を黒ノード，“交差” キューブ（物体が存在するが占有しない領域）をミックスノードで表現する．この

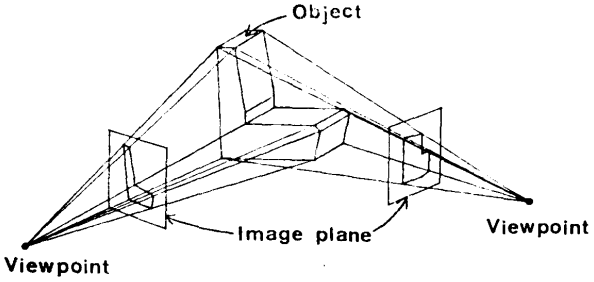
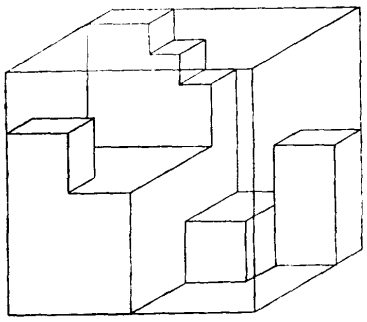
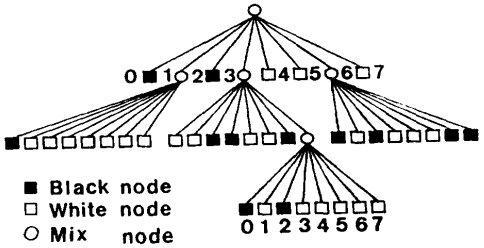


図1 錐体相貫法
Fig. 1 Volume intersection method.



(a) Object



(b) Octree

(1(10000000)1(0011001(10100000))00(10100011)0)

(c) DF－representation

図2 物体(a)と対応するオクトツリー(b)，DF-表現(c)
Fig. 2 An object (a) and its corresponding octree (b), DF-representation (c).

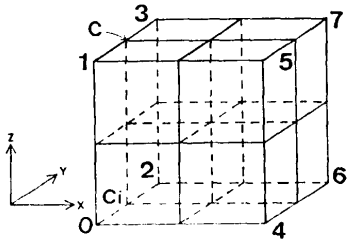


図3 8分割による位置の定義
Fig. 3 The order of octants.

領域は8つのサブ領域（子キューブ）に分割され，それらはミックスノードの子ノードとして表現される．なお，サブ領域と子ノードの位置の対応は図3のようにしている.

3.2 DF 表 現

DF (Depth-First) 表現[9]はオクトツリーの線形かつ高圧縮な表現法で，例えば図2(b)のオクトツリーは同図(c)のように表現される．ここで記号 ‘(’ はミックスノードに入りツリーのレベルが1つ増加すること

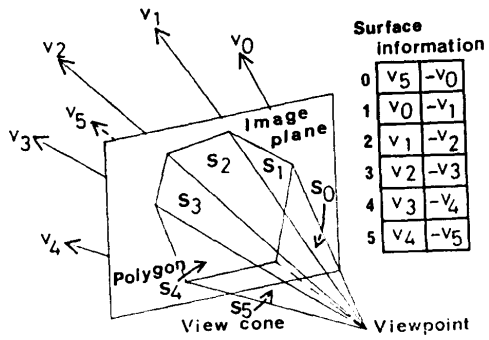


図 4 錐体モデル
Fig. 4 Cone model.

を，記号 ')' はミックスノードから出てレベルが1つ減少することを意味し，記号 '1' は黒ノードを，記号 '0' は白ノードを表している．

3.3 錐体モデル

投影中心と画像多角形とで定義される錐体は，面を2本の半直線で定義した無限錐体である（図4）．本研究では，面の外向き法線ベクトルに対して右ネジの方向に半直線のベクトルを設定している．

3.4 非凸錐体の凸分割

投影中心と画像多角形とで定義される錐体を凸形状の錐体（以後凸錐体と記す）と仮定すると，ある領域（キューブ）の DF-表現（オクトツリー）はその内の錐体の情報のみで作成できる．このことから，DF-表現を作成する・しないが領域ごとに選択できる．したがって，本稿では非凸な錐体は凸分割により凸錐体の集合で表現することにする．

本研究では，非凸な画像多角形をいくつかの凸多角形に分割することで，その凸分割を実現している[10]（図5）．この分割にかかる計算量は $O(N_1+N_2^3)$ である（N_1：非凸な画像多角形の頂点数，N_2：内角の大きさが 180° 以上の頂点数）．

4. 実物体のオクトツリー作成アルゴリズム

この章では，実物体の近似領域をオクトツリーで表現するアルゴリズムを説明する．はじめに，すべての視点に対する錐体はすでに得られており，さらにそのうち非凸錐体は凸錐体の集合で表現されているものとする．

このアルゴリズムでは，まず全体空間に対応するキューブを8等分割し，それらの子キューブの共通領域に対する "内側"，"外側"，"交差" を判断する．次に，"内側"，"外側" 子キューブにはそれに対応する DF-表現 '1'，'0' を出力し，"交差" 子キューブには全体

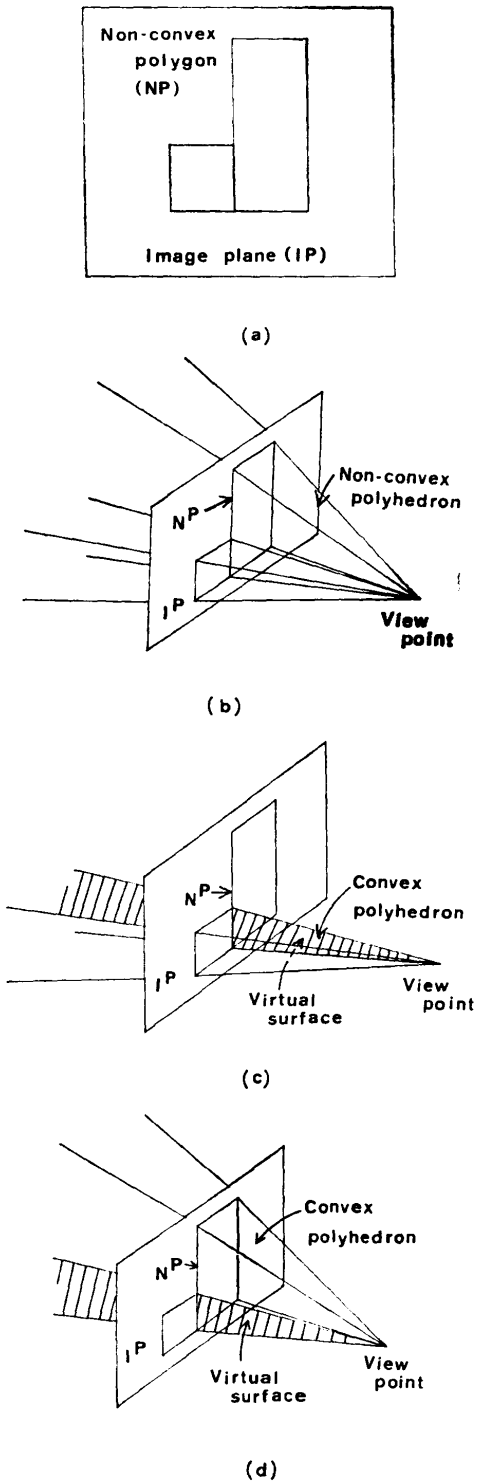


図 5 非凸錐体の凸分割 (a) 非凸な画像多角形の凸分割，(b) 非凸錐体，(c), (d) 凸錐体
Fig. 5 Convex division of a non-convex cone. (a) Division of a non-convex polygon, (b) its view cone, (c) and (d) convex cones for the view cone.

空間に対応するキューブに行ったのと同様の処理を施す．これによりアルゴリズムは再帰的に進行する．このアルゴリズムでは，“交差”キューブを［基本処理］で再帰的に処理しているが，その［基本処理］はこれから説明する2つのサブステップから構成される．

4.1 子キューブの内外・交差判定[サブステップ 1]

このサブステップでは，［基本処理］が処理するキューブの8つの子キューブの共通領域に対する“内側”，“交差”，“外側”を判断し，それらを**分類情報**として保持する．このサブステップは3つの手続き（錐体作成，錐体結合，錐体統合手続き）から構成される．錐体作成手続きは，ある凸錐体に対する8つの子キューブの分類情報を作成する．錐体結合手続きは，ある非凸錐体に対する8つの子キューブの分類情報を作成する．ここで，その分類情報は非凸錐体を構成する凸錐体の分類情報—錐体作成手続きによりすでに作成されている—に**表1**の規則を適用することで得られる．錐体統合手続きは，共通領域に対する8つの子キューブの分類情報を作成する．すべての視点の錐体に対する分類情報は上の2つの手続きから得られている．それらの分類情報に**表2**の規則を適用することで共通領域に対する8つの子キューブの分類情報は得られる．

4.1.1 錐体作成手続き

この手続きでは，キューブC内のある凸錐体についての分類情報を作成している．まず，その凸錐体が交差する子キューブを選択し，すべての“交差”子キューブを決定する．次に，“交差”子キューブ以外の子キューブである“非交差”子キューブの凸錐体に対する内外を判断し，それらを“内側”，“外側”子キューブに分類する．

（1）“交差”子キューブの決定

凸錐体のキューブ内での面数により［PLURAL 手続き］，［SINGLE 手続き］のうちの1つを選択し，キューブ内の凸錐体のすべての面を8つの子キューブに重複を許して分配する．

［PLURAL 手続き］

キューブが2つ以上の凸錐体の面を含むとき，この手続きを利用する（**図6**）．［PLANE ルーチン］と［PROJECTION ルーチン］を用いると，ある面Pと交差する子キューブが選択される．そこで，すべての面にこれら2つのルーチンを用いて，キューブ内のすべての“交差”子キューブを決定する．

［PLANE ルーチン］

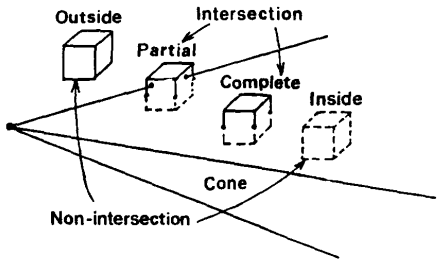


図6 錐体とキューブの交差パターン
Fig. 6 Three dimensional intersecting pattern between cone and cubic region.

このルーチンでは，面 P を含む平面 S が交差する子キューブを選択する．平面Sの外向き法線ベクトルを $\vec{n}=(n_x, n_y, n_z)$，平面 S 上の1点を $U_0(\vec{u}_0)$ とし，全体空間の任意点 $U:\vec{u}=(u_x, u_y, u_z)$に関して，$d(S, U)=\vec{n}\cdot(\vec{u}-\vec{u}_0)=n_x*u_x+n_y*u_y+n_z*u_z-d$ $(d\equiv\vec{n}\cdot\vec{u}_0)$ なる関数を定義する．また，平面Sで全体空間を分割したとき，凸錐体が存在する領域を内側，そうでない領域を外側と定義する．関数 $d(S, U)$ は平面 S から点 U までの変位を表しており，その符号が負ならば点 U は平面 S の内側に，正ならば点 U は平面 S の外側に存在する．

子キューブ C_i の位置を表す10進表現の添字 i の2進表現が $(r_0r_1r_2)_2$ であるとき，$\vec{i}=(r_0, r_1, r_2)$ なるベクトルを定義する．

平面 S と交差する子キューブを見つけるため，まずキューブ C の重心点 G に関する $d(S, G)$ の正負より以下の処理を行う．

（1） $d(S, G)<0$ [$d(S, G)>0$] のとき

点 G は平面 S の内側［外側］に存在する．この場合，内積 $\vec{n}\cdot\vec{i}$ が最大［最小］となる位置 i を k と表すと，子キューブ C_k の周辺で平面 S は子キューブと交差することがわかる（**図7**）．点 G を除く C_k の頂点 v_i に関し $d(S, v_i)>0$ [$d(S, v_i)<0$] となると，v_i は S の外側［内側］に存在し，平面 S は線分 Gv_i に接するすべての子キューブと交差する．

（2） $d(S, G)=0$ のとき

点 G は平面 S 上に存在する．この場合，内積 $\vec{n}\cdot\vec{i}$ が最小となる子キューブは平面Sの内側，最大となる子キューブは平面Sの外側となる．その他の子キューブは平面 S と交差する．

［PROJECTION ルーチン］

キューブ内に面が2つ以上存在するとき，平面 S と子キューブの交差は面 P が子キューブと交差する必要条件にしかならないので，面 P と交差しない子

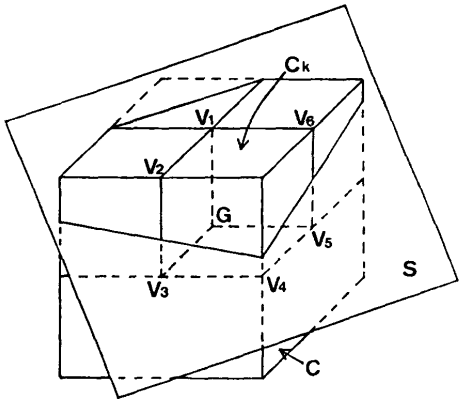


図 7 平面Sと子キューブの交差
Fig. 7 Intersection between plane S and cubic subregions.

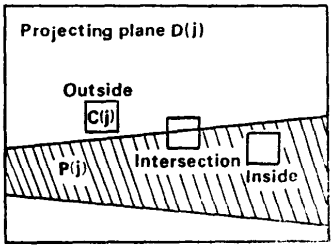


図 8 正射影平面での面 P とキューブの交差パターン
Fig. 8 Two dimensional intersecting pattern between surface P and cubic region.

キューブも得てしまう．そこで，このルーチンでは必要十分条件—X, Y, Z 軸に垂直な平面 $D(X), D(Y)$, そして $D(Z)$ への面 P と子キューブの正射影がすべて交差すれば，そのときのみ面 P と子キューブは実際に交差する—を用いて，面 P と交差する子キューブを正確に決定する（図8）．

この必要十分条件の正当性は，平面 S と子キューブが交差するという前提のもとで［付録1］で証明される．ゆえに，[PLANE] と [PROJECTION] ルーチンはこの順に利用される．

[SINGLE 手続き]

キューブが凸錐体の1つの面のみを含むとき，この手続きを利用する（図6）．このとき，平面 S と子キューブの交差は面 P と子キューブの交差の必要十分条件になる．したがって，前述の [PLANE ルーチン] のみでキューブ内のすべての“交差”子キューブが決定できる．

（2）“非交差”子キューブの内外判定

[JUDGE 手続き]

キューブ内の凸錐体のすべての面に [PLANE ルーチン] を用いたとき，

1) 1つの面でも重心点 G を外側と判断すれば，“非交差”子キューブは凸錐体の“外側”．

2) すべての面で重心点 G を内側と判断すれば，“非交差”子キューブは凸錐体の“内側”．

3) そうでなければ，[PLANE ルーチン]（2）で得られた平面Sに対する内外情報より“非交差”子キューブの内外は決定される．

これらの処理は以下の性質を用いている．

性質1 凸錐体を構成する面が重心点 G と交差しなければ，すべての“非交差”子キューブの内側・外側という属性は同じになる．そして，キューブの重心点 G が内側（外側）なら，すべての“非交差”子キューブは“内側”（“外側”）となる．□

性質2 キューブ内の凸錐体のすべての面に [PLANE ルーチン] を用いたとき，1つの面でも重心点 G を外側と判断すれば点 G は外側，すべての面で内側と判断されれば点 G は内側である．□

4.1.2 錐体結合手続き

この手続きでは，キューブ内のある視点から得られる錐体についての分類情報を作成する．この錐体は，一般には凸錐体の集合で定義されるがそれは以下のような理由による．

1) 画像多角形が非凸になればその錐体も非凸となり，その錐体は複数個の凸錐体で表現される．

2) 2つ以上の実物体が存在するとき，それらに対応する複数個の凸錐体が現れる．

すべての凸錐体に対して“外側”となる子キューブはある視点の錐体の“外側”，ある凸錐体に対して“内側”となる子キューブはある視点の錐体の“内側”，そうでない子キューブはある視点の錐体と“交差”と判断できる．そこで，ある視点の錐体を構成する各々の凸錐体の分類情報—錐体作成手続きで得られている—に表1の規則を逐次用いて，ある視点の錐体についての分類情報を作成する．

ここで注意しなければならないのは，凸分割によって生じた面とのみ交差する子キューブは“交差”ではなく“内側”として扱うことである．この例外処理に

表 1 錐体結合手続き
Table 1 Cone unifying rules.

Cone union rules		Previous state		
		outside	intersection	inside
Present state	outside	outside	intersection	inside
	intersection	intersection	intersection	inside
	inside	inside	inside	inside

より分割面を表現するキューブは作成されず，アルゴリズムの効率は低下しない．

4.1.3 錐体統合手続き

すべての視点の錐体のうちキューブ内に存在する錐体の共通領域—実物体の近似領域—についての分類情報を作成する．キューブ内のある視点の錐体に対して“外側”である子キューブは共通領域の“外側”，キューブ内のすべての視点の錐体に対して“内側”である子キューブは共通領域の“内側”，そうでない子キューブは共通領域と“交差”と判断できる．各々の視点の錐体についての分類情報に表2の規則を逐次用いて，共通領域についての分類情報を作成する．

4.2 DF-表現の作成［サブステップ2］

［基本処理］が扱うキューブはいつも“交差”キューブ（ミックスノード）なので，まず記号‘(’を出力する．そして，［サブステップ1］で得られた分類情報にしたがって，番号0～7（図3）の順に8つの子キューブの DF-表現を作成する．

（1）“内側”子キューブの場合

記号‘1’（共通領域の内部）を出力する．

（2）“外側”子キューブの場合

記号‘0’（共通領域の外部）を出力する．

（3）“交差”子キューブの場合

共通領域の境界と交差する子キューブであり，内部を詳細化—内部の DF-表現（オクトツリー）を作成—しなければならないので，この子キューブは［基本処理］で処理する．この再帰処理よりアルゴリズムはオクトツリー上を縦型探索(深さ優先探索)することになり，DF-表現が自然に作成される．ただし，子キューブがオクトツリー作成レベルにあれば［基本処理］を呼び出さず記号‘1’を出力する．

8つの子キューブで上の処理が終了したら記号‘)’を出力し，現在のキューブの［基本処理］を終了する．そして，その親キューブの［基本処理］にもどる．

このアルゴリズムが［基本処理］を用いて処理するのは，オクトツリーの各レベルで共通領域と交差するキューブだけであり，実物体に関して不必要な領域は処理しないことがわかる．

表 2 錐体統合手続き
Table 2 Cone intersecting rules.

Cone intersection rules		Previous state		
		outside	intersection	inside
Present state	outside	outside	outside	outside
	intersection	outside	intersection	intersection
	inside	outside	intersection	inside

5. アルゴリズムの計算量の評価

この章では，画像多角形の辺数 N，視点数 n，オクトツリーの作成レベル r，実物体の近似領域としての共通領域の表面積 S，共通領域上の稜の合計長さ L に関して，共通領域を作成するためのいくつかのアルゴリズムの計算量を評価する．

1) 3次元モデルによる比較

（a） 多面体で表現した場合[1)]

これまでの処理で作成されている $k*N$ 個の面をもつ共通領域（多面体）V と面数 N の錐体の交差を調べ，$(k+1)*N$ 個の面をもつ共通領域 V を新たに作成する．すべての視点に関してこの処理を逐次行うと，アルゴリズムの計算量は

$$\sum_{k=1}^{n-1} C_0 * kN * N = C_0 n(n-1)N^2/2$$

となり，n^2 そして N^2 に比例した値となる．ここで，定数 C_0 は2つの面の交差を調べ，もし交差するときにはその交線・交点を面の隣接関係にしたがって共通領域 V に登録するのに必要な計算量である．

（b） オクトツリーで表現した場合

提案したアルゴリズムでは，［基本処理］を利用して共通領域と交差するキューブをオクトツリーのレベルごとに処理しているが，その交差キューブにも図9に示すように3つのタイプがあり，それらを処理するのに必要な計算量も異なる．タイプAのキューブはある視点の錐体の一つの面とのみ交差するもので，［PLANE ルーチン］で一回だけ処理すればよい．しかし，タイプBのキューブは複数の錐体の面と交差するもの

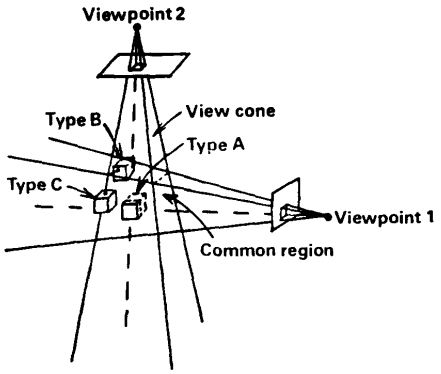


図 9 共通領域と交差するキューブの分類
Fig. 9 The three types of cubic regions intersecting the common region.

で，[PLANE ルーチン] で二回以上処理され，タイプCのキューブはある視点の錐体の二つの面と交差するもので，[PLANE ルーチン] と [PROJECTION ルーチン] とで処理される．

ここで，タイプAのキューブの個数は共通領域の表面積 S，タイプBとCのキューブの個数は共通領域上の稜の長さ L で評価でき，アルゴリズムの計算量もそれらのパラメータ S と L に依存する．これらのことは [付録2] で正確に議論され，レベル j でのアルゴリズムの計算量は $C_1*4^j*S/S_W+C_2*2^j*L/L_W$ となる（S_W：全体空間を構成する面の面積，L_W：全体空間の一辺の長さ，C_1, C_2：定数）．

そして，レベル $r-1$ までのすべてのレベルにおいてこの計算量を合計するとアルゴリズムの計算量が得られる．

$$\sum_{j=0}^{r-1}\{C_1*4^j*S/S_W+C_2*2^j*L/L_W\}$$
$$=C_1(4^r/3)S/S_W+C_2*2^rL/L_W.$$

この式から，アルゴリズムの計算量はパラメータ S, L, そして r に依存することがわかる．定数 C_1 そして C_2 は，共通領域の面そして稜を含むキューブを処理するのに必要な計算量である．ここでパラメータ L はパラメータ n そして N に依存するが，それは高々 n そして N に比例する程度である．

実物体に対する共通領域の近似精度を高めるにつれて，稜の長さ L は増加するが表面積 S は一定である．したがって，実物体を高精度で近似した共通領域を作成するとき，アルゴリズムの計算量は表面積 S よりもむしろ稜の長さ L に依存する．

3次元空間内での面と面の交差判定は多大な計算量を必要とするので，C_0 は C_1 そして C_2 より大きな値をとる．さらに，実物体に対する共通領域の近似精度をあげるためにはパラメータ n と N を大きくしなければならない．したがって，3次元モデルとして多面体を利用するよりもオクトツリーを利用するほうが計算量の点で有利である．

2) 錐体とキューブの交差判定法の比較

3次元モデルとしてオクトツリーを利用する場合でも，錐体とキューブの交差を2次元平面上で判定する方法[6]と3次元空間内で判定する方法がある．これら2つの方法は，キューブ内の8つの子キューブの分類情報の作成に必要な計算量の比較で評価できる．

（a） 2次元平面上で判定する場合

交差判定はキューブを2次元平面に投影して得られる投影六角形（透視変換で得られる単一でない六角形）と画像多角形の間で行われる．文献 6) では画像メモリを利用して，子キューブごとに投影六角形と画像多角形の交差を調べている．このとき，投影六角形（子キューブ）を2つの状態にわけて考える．

（a-1） 子キューブの投影六角形内にある画像多角形の辺が1つの場合

オクトツリーのレベルが大きくなると，この状態が一般的になる．このとき，8つの子キューブのうち平均的に4つは“内側”または“外側”，4つは“交差”となる．

1つの子キューブを“内側”または“外側”と判定するのに必要な計算量は，$6*15C_0+C_1+C_2$（C_0：乗除算一回の計算量，C_1：投影六角形の塗りつぶしにかかる計算量，C_2：画像多角形と投影六角形の交差を完全に調べるのに必要な投影六角形周囲のバイナリーサーチにかかる計算量）となる．また，1つの子キューブを“交差”と判定するのに必要な計算量は $3*15C_0$ となる．ここで，子キューブの1つの頂点を投影面に透視変換するのに必要な乗除算は15回としている[11]．

（a-2） 子キューブの投影多角形内にある画像多角形の辺が2つの場合

（a-1）の場合とほぼ同等の計算量となる．

したがって，キューブ内の8つの子キューブの分類情報の作成に必要な計算量は，$4(6*15C_0+C_1+C_2)+4*3*15C_0=540C_0+4(C_1+C_2)$ と評価できる．

（b） 3次元空間内で判定する場合

交差判定は錐体の面とキューブとの間で行われるが，3次元空間では子キューブごとに分類処理するのではなく，8つの子キューブを一度に分類処理している．この場合も，（a）と同様にキューブを2つの状態にわけて考える．

（b-1） キューブ内に面が1つ

このキューブは [SINGLE 手続き]，[JUDGE 手続き] の組み合わせで処理される．[SINGLE 手続き] は [PLANE ルーチン] のみを用いており，そこでは平均で6つの点の平面方程式への代入が必要である．また，[JUDGE 手続き] は [PLANE ルーチン] で得られた情報をそのまま利用しているので新たな計算を必要としない．したがって，このキューブ内の8つの子キューブの分類情報の作成に必要な計算量は $6*3C_0$ と評価できる．

（b-2） キューブ内に面が2つ

このキューブは [PLURAL 手続き]，[JUDGE 手

続き] の組み合わせで処理される. [PLURAL 手続き] は [PLANE ルーチン], [PROJECTION ルーチン] を用いている. この場合, [PLANE ルーチン] では平均で6つの点の2つの平面方程式への代入が, [PROJECTION ルーチン] では3つの点の2つの直線の方程式への代入が3つの射影面において必要である. また, [JUDGE 手続き] は [PLANE ルーチン] で得られた情報をそのまま利用しているので新たな計算を必要としない. したがって, このキューブ内の8つの子キューブの分類情報の作成に必要な計算量は $(6*6+3*12)C_0$ と評価できる.

これら2つの方法を比較すると以下のようなことがいえる.

1) 前者ではキューブの頂点を投影面へ透視変換する必要があるが後者ではその必要はない. この透視変換にはかなりの計算量が必要となる.

2) 前者の計算量は画像メモリの機能に依存する. すなわち, メモリの読みだしや投影六角形の塗りつぶしなどの時間がアルゴリズムの計算量に大きく影響する. 一方, 後者は画像メモリを必要としない.

3) 前者では投影六角形が対称に8分割できないので, 各々の子キューブを独立に処理している. しかし, 後者ではキューブが対称に8分割されるのですべての子キューブを一度に処理できる.

これらのことから, 3次元空間内でキューブ内の8つの子キューブの分類を行うほうが計算量の点でよいことがわかる.

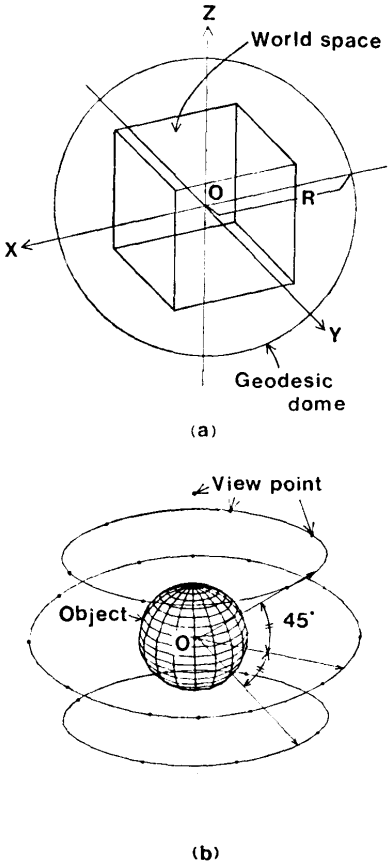


図 10 観測ドームと視点位置
Fig. 10 Geodesic dome (a), and view points (b).

6. 実験結果

まず, 観測ドーム (点O: 中心, R: 半径) とよばれ

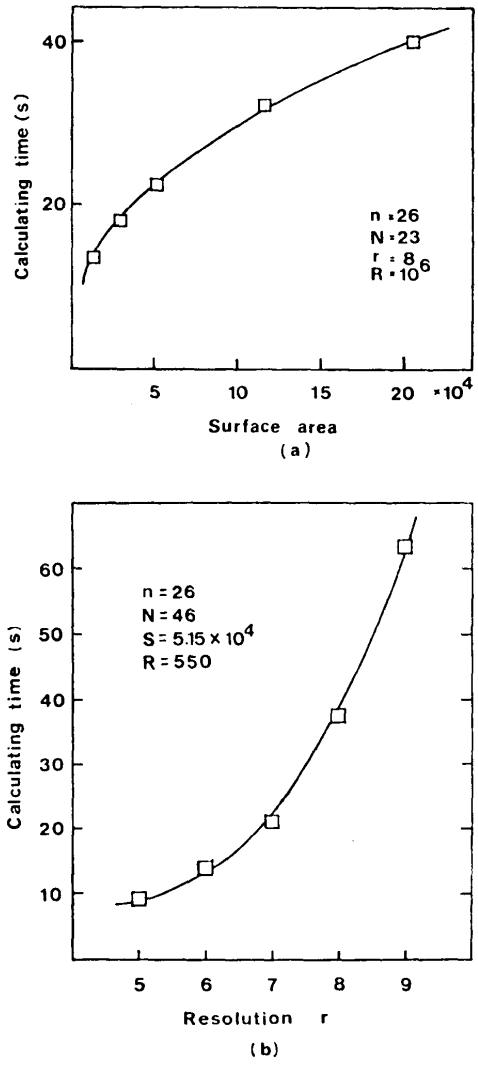


図 11 アルゴリズムの計算時間の変化, (a)共通領域の表面積 S, (b)オクトツリーの作成レベル r
Fig. 11 Calculating time of the algorithm, the surface area S of the common region (a), the finest resolution level r of the octree (b).

る球を定義しその上にすべての投影中心をおく．また，実物体と全体空間 [1024, 1024, 1024] ($S_w=4^{10}$) をそれらの重心Gが点Oと一致するようにおく（**図 10**）．

（1） 評価の検証

ここでは，共通領域の表面積 S，オクトツリー作成レベル r に関するアルゴリズムの計算量の評価を確認する．面積 S は単位面積 ($=S_w/4^{10}$) で定義する．

この実験では，高精度で実物体を近似した領域を作成しているので ($n=26$, $N=23$)，アルゴリズムの計算量は長さ L の影響をうける．ここでは，実物体として球を近似した多面体を利用しており，稜の長さLは表面積Sのルートに比例する ($L\propto\sqrt{S}$)．5章の評価が正当であることが**図 11** のグラフからわかる．

（2） アルゴリズムの高速性

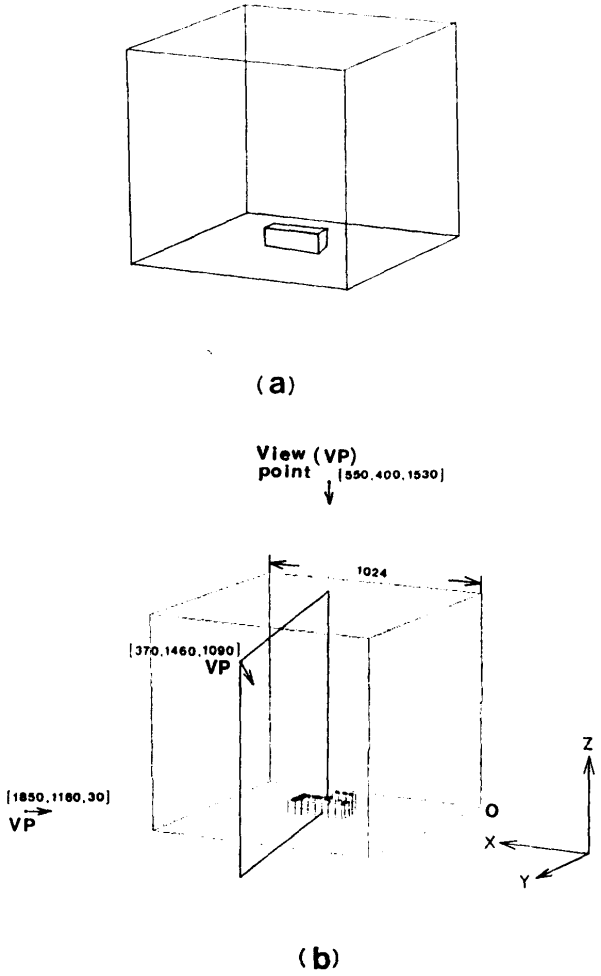


図 12 直方体の変換，（a）直方体と，（b）生成されたオクトツリーモデル

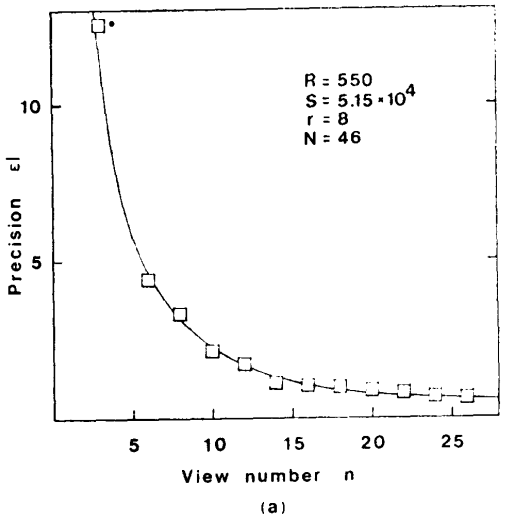
Fig. 12 Building the octree for a parallelepiped, a parallelepiped (a), and its octree (b).

Hong らの方法一提案した方法と同じように透視投影を利用し，世界座標系でオクトツリーを作成できる一との比較で本方法の高速性を確認する．

本研究の方法では，直方体 [220, 75, 60]（**図12**(a)）に対するオクトツリー（作成レベル $r=5$，半径 $R=1500$：図 12(b)）を生成するのに 100 ミリ秒(Melcom 350-60：3.7 MIPS を使用）かかるが，Hong らの方法では同様の処理に約 180 秒（Vax 11/780：1.8MIPS を使用）かかる．

（3） 共通領域の近似精度

ここでは，パラメータ n または N の変化に関して，実物体を直接変換したオクトツリー O_o[12] に対す



* Comparative experiment with Kim's method[6] ($n=3$; three image planes which are vertical together, $R=10^4$)

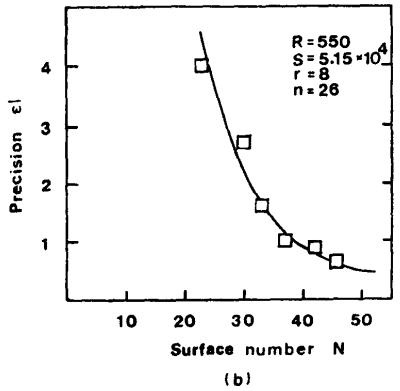


図 13 共通領域の近似精度の変化，（a）視点数n，（b）画像多角形の辺数 N

Fig. 13 Approximating precision of the common region for the object, view number n(a), the number N of segments of the polygon (b).

る実物体を錐体相貫法で変換したオクトツリー O_v の近似精度を検討する（図 13）.

V_o：オクトツリー O_o の黒ノードに対応する領域の体積.

D：O_o と O_v において，一方は白ノード，他方は黒ノードとなるノードに対応する領域の体積.

ε_1：O_o と O_v の近似誤差（$\triangleq 100 * D/V_o$）.

図 13 が示すように，パラメータ n または N の増加にしたがって，実物体に対する共通領域の近似精度は良くなる.

7. むすび

錐体相貫法ではパラメータ n そして N を大きくとらないと，実物体に対する3次元モデルの近似精度はよくならない．これは，1つの視点からの処理では，実物体のある周囲の形状情報しか収集できないからであり，近似精度を良くするためには多くの視点からの情報を利用しなければならない．そこで，1つの視点あたりの処理を高速にすることが必要になる.

この観点から，本報告では実物体を近似した3次元モデル（オクトツリー）を任意の座標系のもとで高速に作成するアルゴリズムを提案した．簡単な前処理により，この方法は非凸形状の実物体の入力または2つ以上の実物体の同時入力にも効率的に適用できる.

すべての領域（キューブ）は完全に独立に処理できるので，このアルゴリズムには「並列処理」の概念が利用できる．そのうえ，キューブ内の処理は単純なのでハードウェア化も容易である．したがって，提案したアルゴリズムにはさらに高速化が期待できる.

参考文献

1) 三宅，土居：立体形状の多面体近似システム，情報処理学会論文誌，Vol. 25, No. 5, pp. 745-754 (1984).
2) 例えば，Pavlidis, T.: *Structural Pattern Recognition*, Springer-Verlag, New York (1977), 安居院ほか：原図形に忠実な直線近似法，信学論 (D), Vol. J-68 D, No. 8, pp. 1539-1540 (1985).
3) Meagher, D.: Geometric Modeling Using Octree Encoding, *Computer Graphics and Image Processing*, Vol. 19, No. 2, pp.129-147 (1982).
4) Jackins, C. L. and Tanimoto, S. L.: Oct-Trees and Their Use in Representing Three-Dimensional Objects, *Computer Graphics and Image Processing*, Vol. 14, No. 3, pp. 249-270 (1980).
5) Kim, Y. C. and Aggarwal, J. K.: Rectangular Parallelepiped Coding: A Volumetric Representation of Three-Dimensional Objects, *IEEE J. of Robotics and Automation*, Vol. RA-2, No. 3, pp. 127-134 (1986).
6) 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).
7) 例えば，大田，金出：走査線上の整合性を考慮した2段の動的計画法によるステレオ対応探索，情報処理一創立25周年記念論文特集一，Vol. 26, No. 11, pp. 1356-1363 (1985).
8) Horn, B.: Obtaining Shape from Shading Information, Winston, P. H. (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York (1975).
9) Kawaguchi, E. and Endo, T.: On a Method of Binary Picture Representation and Its Application to Data Compression, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, pp. 27-35 (1980).
10) Shamos, M. I.: Computational Geometry, Ph. D. Thesis, Yale University (1978).
11) 例えば，守川 穣：3次元グラフィックス入門，ASCII, Vol. 9, #9 September, pp. 136-145 (1985).
12) 登尾，福田，有本：BRep からオクトツリーへの変換アルゴリズムとその評価，情報処理学会論文誌，Vol. 28, No. 10, pp. 1003-1012 (1987).

付録 1

面 P そして子キューブ C_i の，YZ 平面（$D(X)$ 平面），ZX 平面（$D(Y)$ 平面），XY 平面（$D(Z)$ 平面）への正射影を領域 $P(j)$ そして領域 $C(j)$ $(j=X, Y, Z)$ と定義する．また，面 P を含む平面 S と子キュー

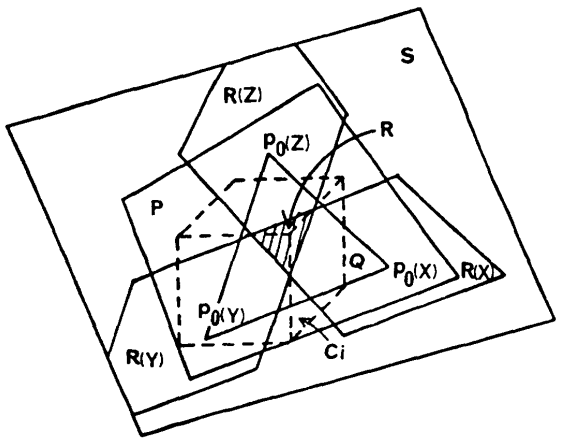


図 14 平面 S 上のキューブの射影領域
Fig. 14 Projecting region of cubic region onto the plane S.

ブ C_i の交差領域を領域 R, 平面 $D(j)$ $(j=X,Y,Z)$ への射影が領域 $C(j)$ となる平面 S 上の領域を $R(j)\cup R$ と定義する (**図14**).

また，2，3次元での面 P と子キューブ C_i の関係を以下のように定義する.

[3D 情報] (図6)

[非交差] $C_i\cap P=\phi$.

[交差] $C_i\cap P\neq\phi$.

[2D 情報] (図8)

[外側] $C(j)\cap P(j)=\phi$.

[交差] $C(j)\supsetneq C(j)\cap P(j)\neq\phi$.

[内側] $C(j)=C(j)\cap P(j)\neq\phi$.

3D 情報と 2D 情報との間には以下の関係がある.

[定理1]

すべての平面 $D(j)$ において，$C(j)\cap P(j)\neq\phi$
$\Leftrightarrow C_i\cap P\neq\phi$.

[証明] ($\Rightarrow$) すべての平面 $D(j)$ において，$C(j)\cap P(j)\neq\phi$ は，面 P 上に $p_0(j)\in R(j)\cup R$ となる点 $p_0(j)$ $(j=X,Y,Z)$ が存在することを意味する (図14). その3点から構成した領域 Q は領域 R と明らかに交差をもつ. また，面 P は凸形状なので領域 Q は面 P に含まれる. したがって，$C_i\cap P\neq\phi$ がいえる. □

($\Leftarrow$) 点 $p\in C_i\cap P\neq\phi$ の平面 $D(j)$ $(j=X,Y,Z)$ への正射影をおのおの $p_1(j)$ とすると，すべての平面 $D(j)$ において $p_1(j)\in C(j)\cap P(j)$ が存在し，$C(j)\cap P(j)\neq\phi$ が成立する. □

[定理2]

ある平面 $D(j)$ において，$C(j)\cap P(j)=\phi$
$\Leftrightarrow C_i\cap P=\phi$.

[証明] ($\Rightarrow$) 定理1における($\Leftarrow$)の証明の対偶. □

($\Leftarrow$) 定理1における($\Rightarrow$)の証明の対偶. □

付 録 2

共通領域 V を構成するある面を $\tilde{P}$ と記述し，面 $\tilde{P}$ の面積と周囲の辺の長さをそれぞれ S_P と L_P で表現する. この S_P と L_P を用いて，面 $\tilde{P}$ とその周囲の辺が交差するレベル j のキューブの個数の上限 $UB(S_P)$ と $UB(L_P)$ を評価する. ここで，そのキューブの平面 $D(X)$, $D(Y)$, そして $D(Z)$ への正射影をブロックとよぶことにする. このとき，ブロックの面積と一辺の長さは，おのおの $S_B=S_W/4^j$ と $L_B=L_W/2^j$ と記述できる. 面積 S_W と長さ L_W は，おのおの全体空間を構成する面の面積と一辺の長さを表している.

上限 $UB(S_P)$ を得るために，前述の3平面のうちのある平面に面 $\tilde{P}$ を正射影し，射像 $\tilde{P}'$ とその周囲の辺が交差するブロックの個数の上限 $UB(S_{P'})$ と上限 $UB(L_{P'})$ を評価する. 射影面としてはその単位法線ベクトルと面 $\tilde{P}$ の単位法線ベクトルの内積の大きさ Mが最大になるものを選ぶ. このとき，射像 $\tilde{P}'$ の面積とその周囲の辺の長さは，おのおの，$S_{P'}=S_P*M$ と $L_{P'}=L_P*M$ で記述される.

上限 $UB(L_{P'})$ は $\alpha L_{P'}/L_B+\beta$ (α,β: 定数) と記述できるが，それは以下のように証明される.

第一に，射像 $\tilde{P}'$ を構成する辺 L_i (長さ l_i) と交差するブロックの個数の上限 $UB(L_i)$ を評価する.

$$\begin{aligned}UB(L_i)&\leqq\lceil(1+\tan\rho)\cdot l_i\cos\rho/L_B\rceil+1\\&\leqq l_i\cdot(\cos\rho+\sin\rho)/L_B+2\\&=\sqrt{2}\,l_i\cdot\sin(\rho+\pi/4)/L_B+2\\&\leqq\sqrt{2}\,l_i/L_B+2.\end{aligned}$$

ここで，角度 ρ は射影面を構成する2つの座標軸と辺 L_i がなす角度のうち小さいものを表す (**図15** (a)).

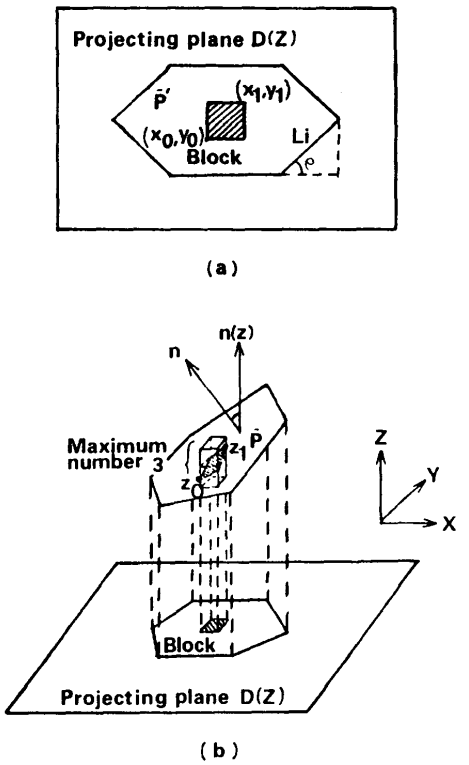


図15 キューブを通してみた共通領域を構成する面とその正射影の関係

Fig. 15 The relation between a finite surface on the common region and its projection via a cubic region.

第二に，射像 $\tilde{P}'$ を構成するすべての辺について上限 $UB(L_i)$ を合計すると上限 $UB(L_{\tilde{P}'})$ が得られる．

$$UB(L_{\tilde{P}'}) \leqq \sum_{L_i \in \tilde{P}'} (\sqrt{2}\ l_i/L_B + 2) = \alpha L_{\tilde{P}'}/L_B + \beta.$$

同様に，上限 $UB(S_{\tilde{P}'})$ は $S_{\tilde{P}'}/S_B + \alpha L_{\tilde{P}'}/L_B + \beta$ と表現される．

さて，ブロックの上で面 $\tilde{P}$ が交差するキューブの個数の上限を評価する．その上限は以下のようにして計算できる．

一般性を失うことなく，面 $\tilde{P}$ を含む平面の方程式：$f = n_x * x + n_y * y + n_z * z - d$ $(n_z \geqq n_x, n_y)$，またブロックの対角に位置する二頂点 (x_0, y_0) と (x_1, y_1) を定義する（図15）．

$$n_x \cdot x_0 + n_y \cdot y_0 + n_z \cdot z_0 - d = 0.$$
$$z_0 = (d - n_x \cdot x_0 - n_y \cdot y_0)/n_z.$$
$$n_x \cdot x_1 + n_y \cdot y_1 + n_z \cdot z_1 - d = 0.$$
$$z_1 = (d - n_x \cdot x_1 - n_y \cdot y_1)/n_z.$$
$$0 \leqq |z_1 - z_0|$$
$$= |(x_0 - x_1) \cdot n_x + (y_0 - y_1) \cdot n_y|/|n_z|$$
$$\leqq (|x_0 - x_1| \cdot |n_x| + |y_0 - y_1| \cdot |n_y|)/|n_z|$$
$$= L_B \cdot (|n_x| + |n_y|)/|n_z|.$$
$$0 \leqq |z_1 - z_0|/L_B \leqq (|n_x| + |n_y|)/|n_z| \leqq 2.$$
$$1 \leqq (\text{交差キューブ数}) = |z_1 - z_0|/L_B + 1 \leqq 3.$$

この結果，上限 $UB(S_{\tilde{P}}) \triangleq 3 * UB(S_{\tilde{P}'})$ と $UB(L_{\tilde{P}}) \triangleq 3 * UB(L_{\tilde{P}'})$ は，各々 $C_1''4^j S_{\tilde{P}}/S_W + C_2''2^j L_{\tilde{P}}/L_W$ と $C_2''2^j L_{\tilde{P}}/L_W$ $(C_1'', C_2'':$ 定数$)$ となる．面 $\tilde{P}$ に関するこれらの上限を共通領域 V を構成するすべての面について合計すると，共通領域とその稜が交差するキューブの個数の上限 $UB(S)$ と $UB(L)$ が，各々 $C_1'4^j S/S_W + C_2'2^j L/L_W$ と $C_2'2^j L/L_W$ $(C_1', C_2':$ 定数$)$ として得られる．ここで，$S = \sum_{\tilde{P} \in V} S_{\tilde{P}}$ と $L = \sum_{\tilde{P} \in V} L_{\tilde{P}}$ である．

したがって，オクトツリーのレベル j でのアルゴリズムの計算量は $C_1 4^j S/S_W + C_2 2^j L/L_W$ $(C_1, C_2:$ 定数$)$ と記述できる．□

(昭和62年6月8日受付)
(昭和62年9月9日採録)

登尾 啓史（正会員）

昭和33年生．昭和57年静岡大学工学部情報工学科卒業．昭和59年同大学院修士課程修了．昭和62年大阪大学大学院基礎工学研究科物理系専攻機械工学分野後期(博士)課程修了．同年同学機械工学科助手．大阪大学工学博士．ロボティックス，コンピュータビジョン，コンピュータグラフィックスの研究に従事．電子情報通信学会，日本ロボット学会，IEEE 各会員．

福田 尚三

昭和38年生．昭和61年大阪大学基礎工学部機械工学科卒業．現在，同大学大学院在学中．ロボティックスの研究に従事．人工知能，並列処理システムなどに興味をもつ．

有本 卓（正会員）

昭和11年生．昭和34年京都大学理学部卒業．同年沖電気(株)入社．電子計算機の開発に従事．37年東京大学工学部助手，42年講師，43年大阪大学基礎工学部助教授，48年教授，42年工学博士．この間，情報理論，制御理論，ディジタル信号処理の高速アルゴリズムに従事する一方，ロボット工学に興味をもち，そのインテリジェント化の研究開発を行っている．電子情報通信学会，計測自動制御学会，日本機械学会，ロボット学会各会員．IEEE の Fellow 会員．