〔原著論文〕

オクトツリーを用いた高速干渉チェック法

登尾　啓史*　　福田　尚三*　　有本　卓*

グラフィックス・シミュレータによるオフラインロボット教示の効率は，シミュレータ内の環境モデルとロボットモデルの干渉チェックの速度により決定される．本研究では，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を環境モデルとして，また，B-reps. という柔軟な動作—回転運動を含む—が表現できるソリッドモデルをロボットモデルとして採用し，そのもとで環境の形状の複雑さに依存しない計算手数の干渉チェックアルゴリズムを提案する．アルゴリズムの基本処理は，B-reps. が存在する領域を8つのサブ領域に分割すると同時に，その B-reps. のパッチを8つのサブ領域に分配することである．分割はオクトツリーの構造に従って行われる．この分配による領域の情報と予め環境モデルが保持している領域の情報を利用すると，アルゴリズムはロボットと干渉の可能性のあるその周囲の領域の障害物のみを選択して干渉チェックできる．このことから，アルゴリズムの計算手数は環境の複雑さに依存しなくなる．したがって，複雑な環境下でも，提案した干渉チェックアルゴリズムは高速に機能する．最後に，アルゴリズムの計算手数を評価し，その正当性とアルゴリズムの高速性を実験により確かめる．

1. はじめに

現在，ロボットの動作教示で最もよく利用されているオンライン教示は，ロボットやその環境内の物体（以下環境物体と略す）が簡単な形状の場合には，ロボット動作の検証や修正が手軽に行なえるので有用である．しかし，ロボットやその環境物体の形状が複雑な場合には，オンライン教示はかなりの時間を必要とし，その間生産ラインをとめなければならないので生産コストの点で不利である．今後このような場合には，グラフィックス・シミュレータによるオフライン教示が用いられると予想される．この場合，ロボットとその環境の干渉チェックの高速化がシミュレータの性能を左右する．従来の干渉チェック高速化の研究としては，ロボットの構成要素（例えば，リンク，車輪など．以下コンポーネントと記す）や環境物体を円柱や直方体などの物体で大まかに近似する方法がある[1,2]．この方法では比較的少数の物体でロボットや環境が表現され，かつ物体間の交差判定も簡単にできるので高速に干渉チェックできる．また，コンポーネントや環境物体を多面体で近似し，それらの間で交差判定をする方法[3,4]，これら2つを組みあわせた方法もある[5,6]．

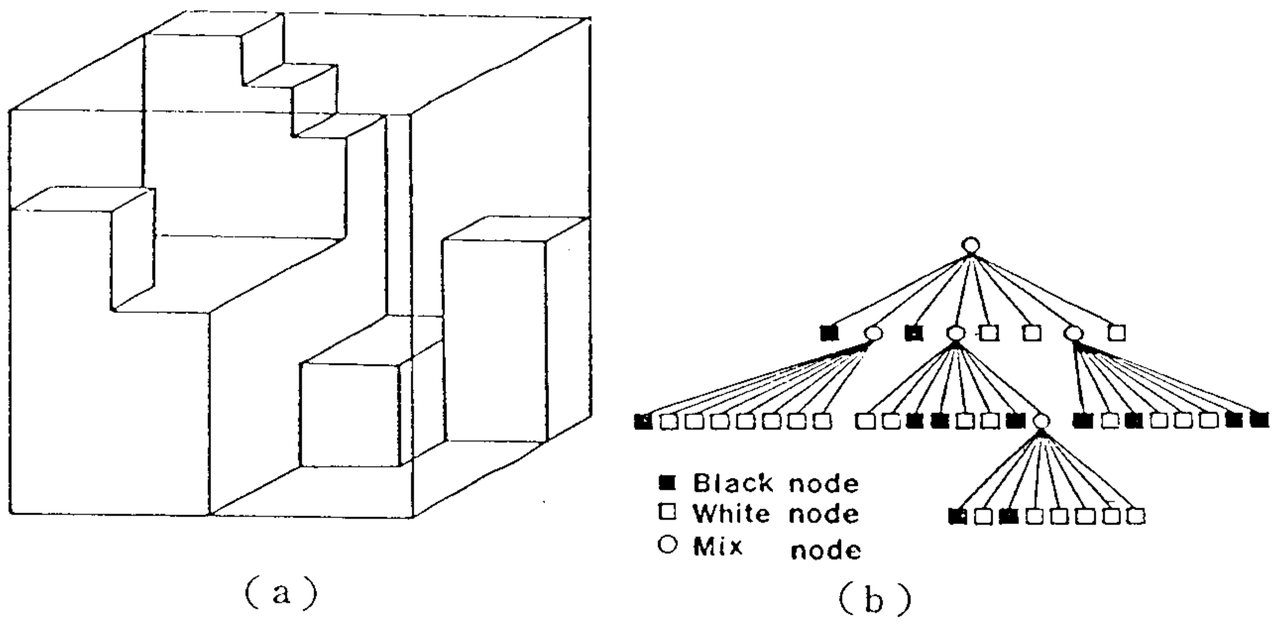
一般に，可能な動作を見逃がさないようにするためには，ロボットのコンポーネントや環境物体の複雑な形状を忠実に近似して干渉チェックする必要がある．この場合，前述の方法では物体の個数や多面体の面数が膨大になるので実時間で干渉チェックできなくなる．

この理由は，そこではロボットだけでなく環境の複雑さに依存したアルゴリズムを用いているからだと考えられる．このことを解決するために，我々は干渉の可能性があるロボット周囲の環境物体の選択による，環境の複雑さに依存しない干渉チェック法を提案する．このような選択は，ロボットやその環境を位置データに関する階層構造でモデル化することから実現される．しかし，ロボットの動作は複雑——回転移動を含む——なので，そのモデルを階層構造で持つのはむずかしい．

ここで，環境を位置データに関する階層モデルであるオクトツリー[7,8]で表現して，干渉チェックをおこなう研究[9]がある．しかし，そこでは具体的なアルゴリズム，特にオクトツリーの階層構造の利用法が示されていない．また，そこでは，ロボットを B-reps. で表現するのが適当であることが述べられているが，干渉チェックアルゴリズムが複雑になるという予想から見送られている．

そこで本論文では，環境をオクトツリー，ロボットをコンポーネント毎に B-reps. で表現し，オクトツリー

原稿受付　1986 年 10 月 31 日
* 大阪大学基礎工学部機械工学科



(a) (b)

Fig. 1 An object (a) and its corresponding octree (b)

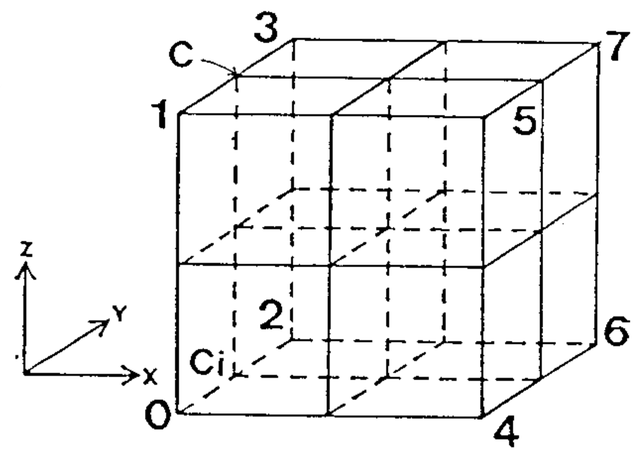


Fig. 2 The order of octants

の階層構造を利用したロボットとその環境との高速な干渉チェックアルゴリズムを提案する．２章では今回用いたモデルの定義，３章では提案するアルゴリズムの説明，４章ではアルゴリズムの手数の評価，５章では実験例について考察する．

2. モデルの定義

2.1 環境モデル

環境モデルとして用いたオクトツリーは，**Fig. 1** のように物体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．オクトツリーでは物体が存在しない領域（キューブC）を白ノード，物体が占有する領域を黒ノード，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．その領域は８つのサブ領域（子キューブ Ci）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．なおサブ領域とオクトツリーの子ノードとの対応は **Fig. 2** のようにしている．

2.2 ロボットモデル

ロボットモデルのコンポーネントは各々 B-reps. で表現されている．B-reps. は，面（パッチ）の集合で物体を定義したソリッドモデルである．本研究では，**Fig. 3** のように法線方向に対し右ネジの順に並べた頂点列で面（パッチ）を表わしている．B-reps. は物体のトポロジカルな性質を保持しており，頂点座標を変更するだけでロボットの平行・回転移動を容易に表現できる．

本研究では，凸形状ではないコンポーネントを予め凸分割し，いくつかの凸形状のサブコンポーネントとして定義する．

3. 干渉チェックアルゴリズム

3.1 干渉・非干渉の判定

前章で述べたロボットモデル（B-reps.）と環境モデル（オクトツリー）の間の干渉は，ロボットモデルの構成要素である個々のパッチと環境モデルの構成要素である個々のキューブとの干渉の有無を調べることでチェックできる．

環境モデル（オクトツリー）の構成要素であるキューブCは，それが対応するオクトツリー上のノードのカラー（Color）――属性――から“黒”（環境物体：Black）・“白”（自由空間：White）・“ミックス”（環境物体と自由空間の混在：Mix）に，また，ロボットモデル（B-reps.）との位置関係（State）――状態――から，“外側”（ロボットの外部領域：Outside）・“内側”（ロボットの内部領域：Inside）・“交差”（ロボットの交差領域：Intersection）に分類される（**Fig. 4**）．

２つのモデル間の干渉の有無・未定を判断するため，キューブの情報（前述の分類結果）を整理したのが **Table 1** である．この表の見方として，例えば，１行１列は“キューブの属性が黒ということは環境物体の領域であり，それがロボットモデルの内側に位置するのでロボットと環境物体は干渉する”ことを意味している．

ここで，“未定”（Undecidedness）と判定されたキュ

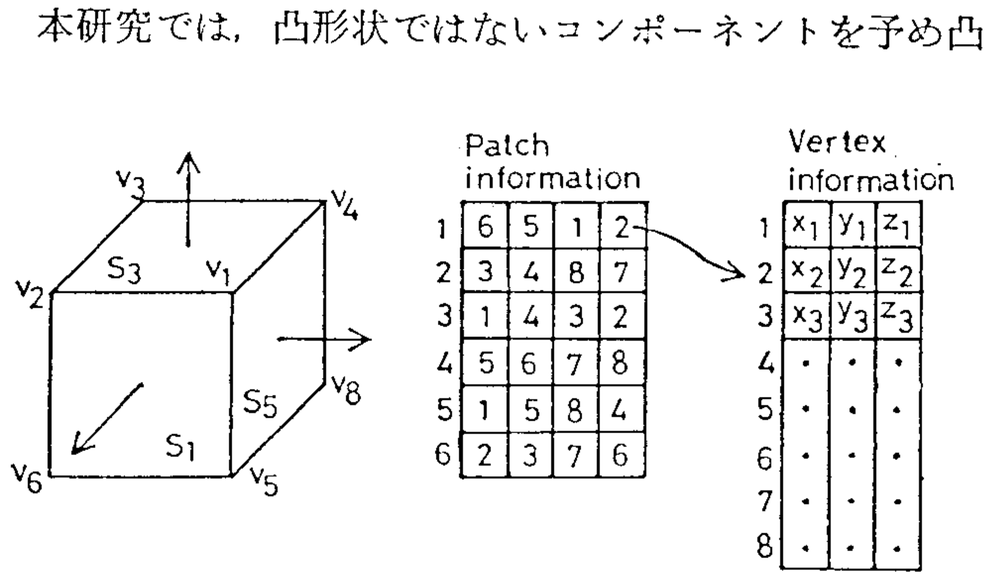


Fig. 3 Boundary representation

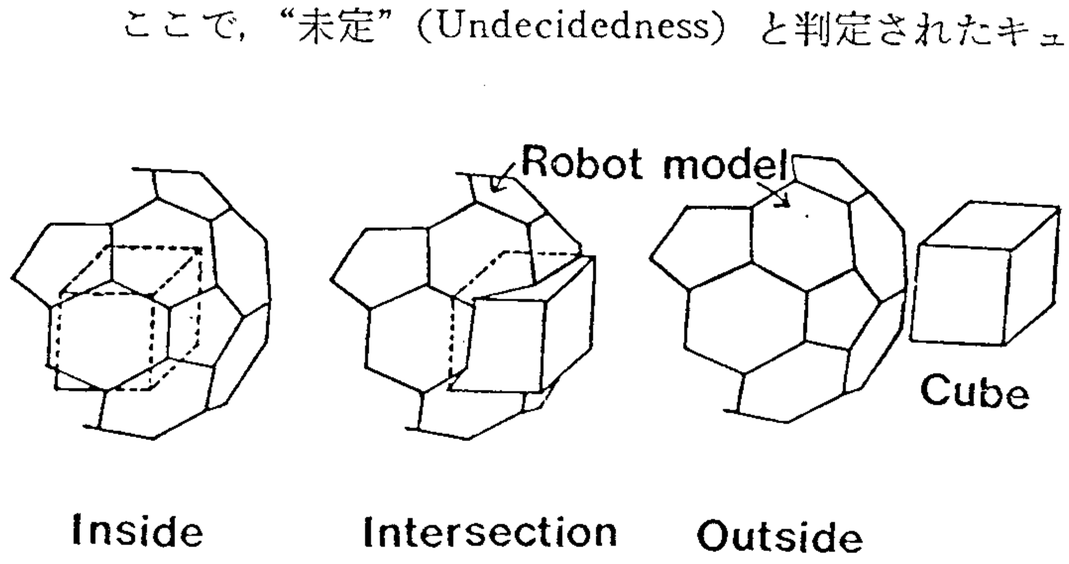


Fig. 4 Relation between robot and cube

Table 1 Relation between color of the cube in the octree and state of the cube for the robot model

Color	State		
	inside	intersetion	outside
black	interfere	interfere	non-interfere
mix	interfere	undecidedness	non-interfere
white	non-interfere	non-interfere	non-interfere

ーブC——現在の環境モデルの解像度では干渉・非干渉の判断がつかないキューブ——は8つの子キューブ Ci に分解し，改めて干渉の有無・未定を調べる．このことから，解像度を上げて干渉を調べなければならないのは，現在の環境モデルの解像度で干渉・非干渉の判断がつかないロボットモデル上の領域（“未定”キューブ）だけであることがわかる．

3.2 未定キューブの分類によるアルゴリズムの効率化

干渉チェックアルゴリズムでは，ロボットモデルが交差する“未定”キューブのみを，環境モデルの解像度を動的に上げて選択していく．このとき，表中のキューブの属性は固有の情報であるが，キューブの状態はロボットモデルが移動するたびに変化する情報である．したがって，位置の確定したロボットモデルに関してキューブの状態を高速に判断しなければならない．

そのために，ロボットモデルに関する“交差”の程度から“未定”キューブを以下の3つの場面に分類し，それぞれの場面の性質を利用してキューブの状態を効率的に判断する（**Fig. 5**）．

場面1：キューブが表現する領域が大きく，ロボットモデルのあるコンポーネントを含む場合．手続きチェック1で処理する．

場面2：キューブが表現する領域が凸形状のコンポーネントを構成する複数の面（パッチ）と交差する場合．手続きチェック2で処理する．

場面3：キューブが表現する領域が小さく，コンポーネントを構成する1つの面（パッチ）とのみ交差する場合．手続きチェック3で処理する．

処理がすすみ環境モデルの解像度が増すにつれて，ロボットモデルと交差する“未定”キューブは場面1から場面3へと変化し，それにともない処理する手続きもチェック1から3へと推移する．手続きチェック 1～3 をふるいにたとえて提案したアルゴリズムを説明すると，それは，環境モデル（オクトツリー）上でロボットモデル（B-reps.）に粗いふるい（チェック1）から細かいふるい（チェック3）の順に使い，ロボットモデル上の干渉の可能性のある部分を高速に絞り込む処理だといえる．干渉チェックアルゴリズムは，**Fig. 6** のブロックダイアグラム（3つの手続きの利用手順）に従い，前述の“未定”キューブを処理している．

3.3 干渉チェックアルゴリズム

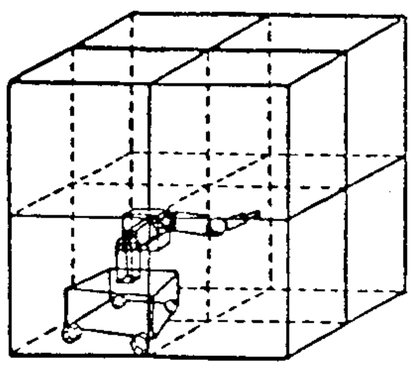
前節では，アルゴリズムを構成する3つの手続きの大まかな機能とその間の処理の流れを説明した．この節では3つの手続きの詳細を説明する．各手続きの目的は，ロボットモデルの入力キューブC内の部分を8つの子キューブ Ci に分配し，かつそれらの子キューブの状態（“内側”，“交差”，“外側”）を分類することである．

（i） 手続きチェック1

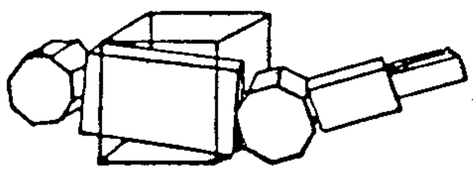
この手続きでは，コンポーネントを完全に含んだキューブ（場面1）を処理する．関数サブチェック1は，そのコンポーネントと交差する子キューブを求め，かつその個数を返値とするもので，つぎのように実現される．

座標軸に垂直な面で構成した最小直方体でコンポーネントを囲み，この直方体と交差する子キューブを“交差”，交差しない子キューブを“外側”と分類する（この時点では子キューブがコンポーネントの“内側”になることはない）．そして，“交差”子キューブとその個数を返す（**Fig. 7**）．

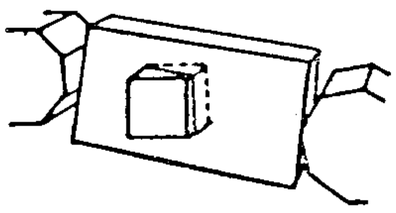
関数サブチェック1の返値が1なら，分類の結果と環境モデル（オクトツリー）固有の属性を用いて，Table 1



Scene 1



Scene 2



Scene 3

Fig. 5 Three types of the ‘undecidedness’ cube

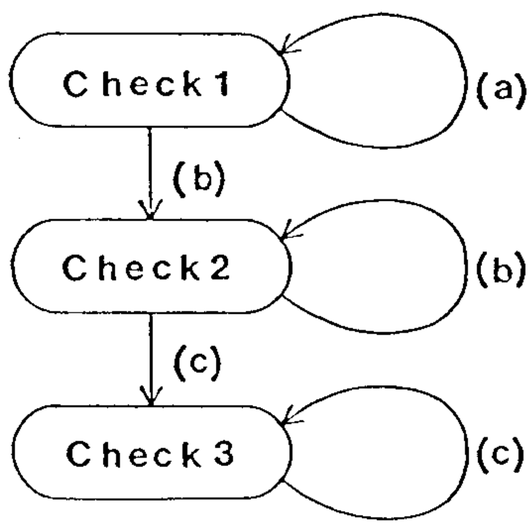


Fig. 6 Process of the ‘undecidedness’ cube in the algorithm

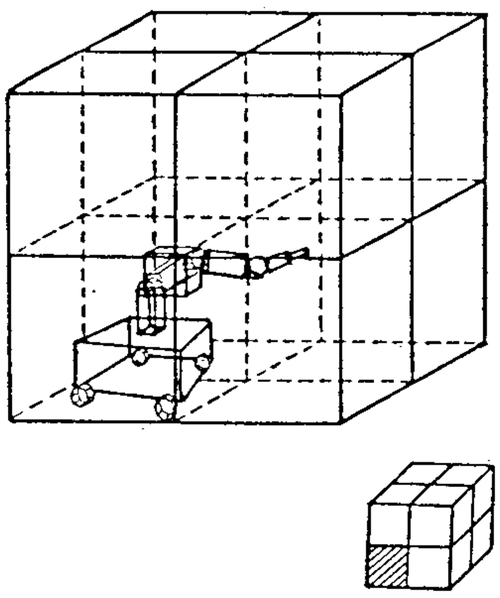


Fig. 7 CHECK procedure

より８つの子キューブの干渉の有無・未定を判断する．ここで，“未定”子キューブは場面１にあたるので，再び手続きチェック１で処理する返値が２以上なら，元のキューブを手続きチェック２で処理する．

（ii） 手続きチェック２

この手続きでは，凸形状のコンポーネントを構成する複数の面（パッチ）と交差するキューブ（場面２）を処理する．関数サブチェック２では，キューブ内の面（パッチ）毎に以下の一連の処理を行なう．

まず，パッチを囲む最小直方体を定義し，それが交差する子キューブを候補子キューブ（その時点で交差の可能性があるもの）とする（付録１[BBOX]，**Fig.** 8(a)）．次に，パッチを含む平面と交差するものをその内から選び候補子キューブをしぼる（付録１[PLANE]，Fig. 8(b)）．最後に，パッチと候補子キューブを３つの座標軸方向に射影し，すべての正射影平面においてそれらの射影像が交差する候補子キューブを“交差”子キューブと確定させる（付録１[PROJECTION]，Fig. 8(c)）．この最後の手続きは，パッチが子キューブと交差するための十分性を考慮しており，必要性を考慮したさきの２つの手続きと違いパッチと交差する子キューブを正確に選択する．

キューブ内のすべてのパッチにこの処理を逐次行なうと，“交差”子キューブが確定する．残りの“非交差”子キューブの状態（“内側”，“外側”）は，コンポーネントに関するキューブの重心点Ｇの位置（内側，外側）と同じになる．ところで，この重心点Ｇの位置は本手続き内のパッチを含む平面との判断ですでに調べている（付録１[PLANE]）．したがって，本手続き内で“非交差”子キューブの状態も同時に確定させる（付録２）．

関数サブチェック２による分類の結果と環境モデル固有の属性を用いて，Table 1 より８つの子キューブの干渉の有無・未定を判断する．ここで，“未定”と判断された子キューブの干渉・非干渉は解像度を上げて調べなければならないが，複数の面（パッチ）と交差するものは場面２にあたるので再び手続きチェック２で，また１つの面（パッチ）とだけ交差するものは場面３にあたるので手続きチェック３で処理する．

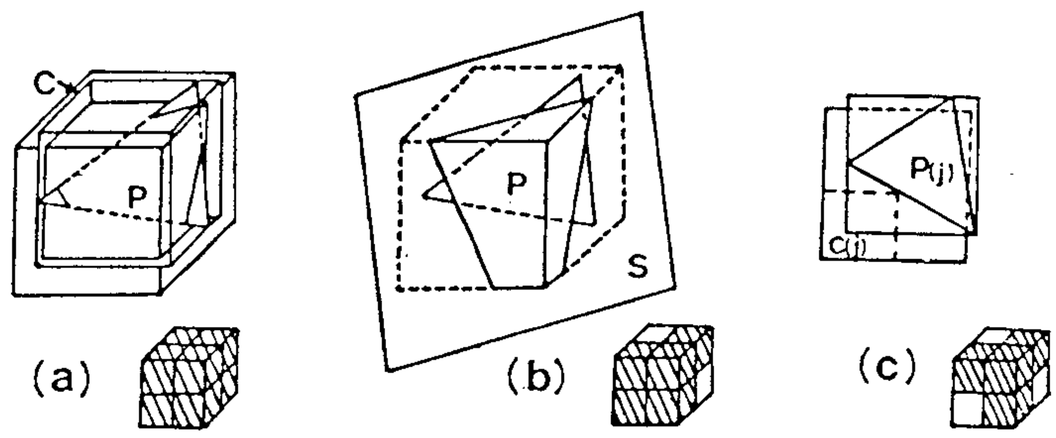


Fig. 8 CHECK procedure (a) BBOX routine (b) PLANE routine (c) PROJECTION routine

（iii） 手続きチェック３

この手続きでは，コンポーネントを構成する１つの面（パッチ）とのみ交差するキューブ（場面３）を処理する．この場合，キューブのパッチとの交差はパッチを含む平面との交差に他ならない（**Fig.** 9）．

そこで関数サブチェック３では，パッチを含む平面と交差する子キューブを選択する手続き（付録 1 [PLANE]）を用いて８つの子キューブを３つの状態（“内側”，“交差”，“外側”）に分類している（付録２）．

関数サブチェック３による分類の結果と環境モデル固有の属性を用いて，Table 1 より８つの子キューブの干渉の有無・未定を判断する．その結果，“未定”と判断された子キューブは同一のパッチと交差しているので，再び手続きチェック３で処理する．

3.4 凸形状ではないコンポーネントの処理

関数サブチェック 1，3 は，凸形状ではないコンポー

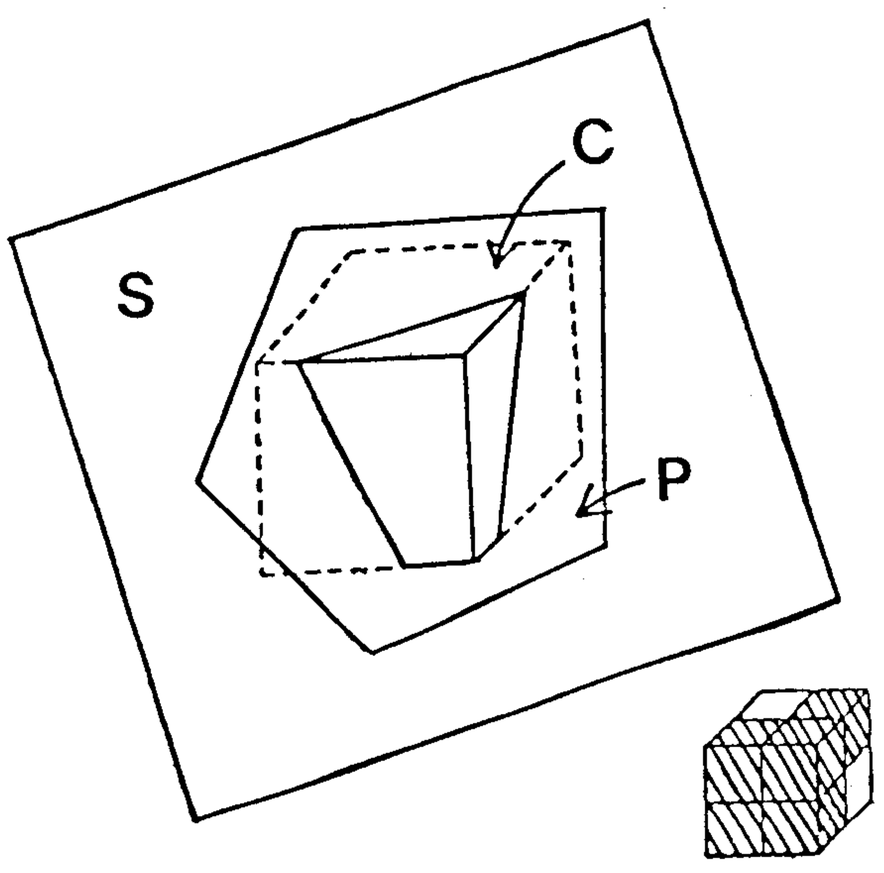


Fig. 9 CHECK procedure

Table 2 Conjunction process of BBOX routine

Plane	Region									
	C_n	C_p	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
ZY-plane	1	0	1	1	0	0	1	1	0	0
ZX-plane	1	1	1	1	1	1	1	1	1	1
XY-plane	1	0	1	1	1	1	0	0	0	0
Conjunction	—	—	1	1	0	0	0	0	0	0

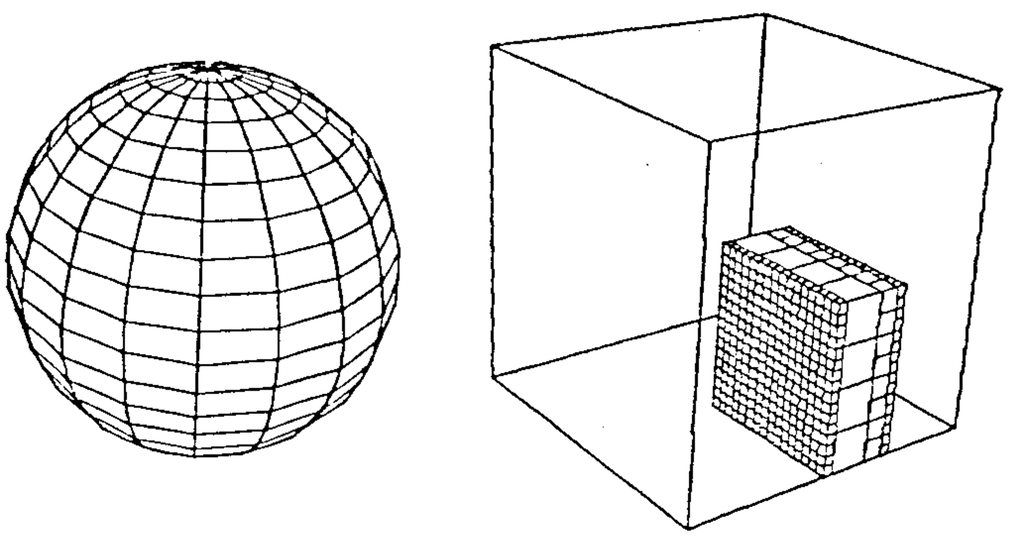
1 : Intersection
0 : Non-intersection

ネントも扱える．しかし関数サブチェック2は，キューブ内のコンポーネントの部分が凸形状でないとその内のパッチ情報のみで8つの子キューブの状態が分類できないので（付録2），凸形状ではないコンポーネントは扱えない．そこでこの節では，関数サブチェック2にかわるものとして凸形状ではないコンポーネントに関する子キューブの分類法を簡単に説明する．なお，凸形状ではないコンポーネントは凸分割により，予めいくつかの凸形状のサブコンポーネントで表現されているとする．

凸形状ではないコンポーネントと交差するキューブは，凸形状のサブコンポーネントのいくつかと交差する．そこで，そのようなサブコンポーネント毎に関数サブチェック 1～3 のうちの1つを選択し，8つの子キューブの状態を分類する．そして，ある子キューブについて得られたすべての状態に **Table** 3 の規則を順次適用すると，凸形状ではないコンポーネントに関するその子キューブの状態が決定できる．

Table 3 は，現在までの処理ですでに得られた現在状態（Present State）と，まだ処理されていないある状態（Next State）とから新たな現在状態を作成するための規則をまとめたものである．但し，最初の現在状態としてはすべての状態の中の任意のものを用いる．

最後に，凸分割によって生じた仮の面のみを含む“交差”（Intersection）子キューブが生じたときは，それを“交差”ではなく“内側”（Inside）と判断する．このことにより，仮の面に関する余分な処理をしなくてもすむ．



（a） （b）

Fig. 10 Institute A : (a) Robot model (b) Environment model

4. アルゴリズムの手数評価

アルゴリズムの計算手数Tを評価するためには，オクトツリーのすべてのレベルにおいてロボットモデルが交差するミックスキューブの個数を知る必要がある．そこで，レベルkのミックスキューブ内に属するロボットモデルの表面積 S_k を仮定し，その面積を使ってロボットモデルが交差するレベルkのミックスキューブの個数 N_k を評価する．

まず，ロボットモデル（B-reps.）を構成する各々のパッチPを XY, YZ, ZX 平面のうちの1つに射影する．この平面としては，$|\vec{l}\cdot\vec{n}|$ を最大化するものを選択する．（XY 平面にはベクトル $\vec{l}=(0,0,1)$ を，YZ 平面にはベクトル $\vec{l}=(1,0,0)$ を，そして ZX 平面にはベクトル $\vec{l}=(0,1,0)$ を対応させている．また，ベクトル $\vec{n}$ はパッチPを含む平面Sの法線ベクトルとする．）パッチPの面積を S_P とすると，この射影 P' の面積 $S_{P'}$ は $S_{P'}=S_P*\cos\theta$ ($\cos\theta=|\vec{l}\cdot\vec{n}|/|\vec{l}|\cdot|\vec{n}|$, $\sqrt{3}/3\leq\cos\theta\leq1$) と算出される．また，レベル$k$のキューブの同様の平面への射影（ブロックと呼ぶ）の面積 $S_{Bk}(=S'/4^k)$(S'：全体空間を構成する面の面積）を用いると，射影 P' と交差するブロックの個数は $S_{P'}/S_{Bk}$ で表現できる．さらに，あるブロック上で平面Sと交差するキューブの個数の最大値は3個となる（付録3）．したがって，パッチPが交差するキューブ数は，$3\cdot S_P\cos\theta/S_B$ となる．

このことから，表面積 S_k のロボットモデル上の領域が交差するミックスキューブの個数 N_k は，$3\cdot C'S_k/S_{BK}$ (C'：定数）となる．そこで，アルゴリズムの計算手数Tは以下のようになる．

$$T=\sum_{k=0}^{n-1}3\cdot C'S_k/S_B=C\sum_{k=0}^{n-1}4^k\cdot S_k/S' \quad (C：定数).$$

これらのことから以下の性質がわかる．

1) ロボットと環境物体の距離によって変化する面積 S_k が計算手数を決定する．つまり，ロボットが環境物体に近づくにつれて，面積 S_k は増加し計算手数も増大する．

2) オクトツリーのレベル（環境モデルの解像度）を上げたとき，そのレベルに対応する面積 S_k は小さいので，総計算手数 T に対して，レベルを上げたことによる計算手数の影響は小さい (**Fig.** 11).

5. 実験結果

この章では，提案したアルゴリズムを2つの設定で実験する．前者は4章で得られたアルゴリズムの性質を確認するためのものであり，後者は複雑な環境でのアルゴ

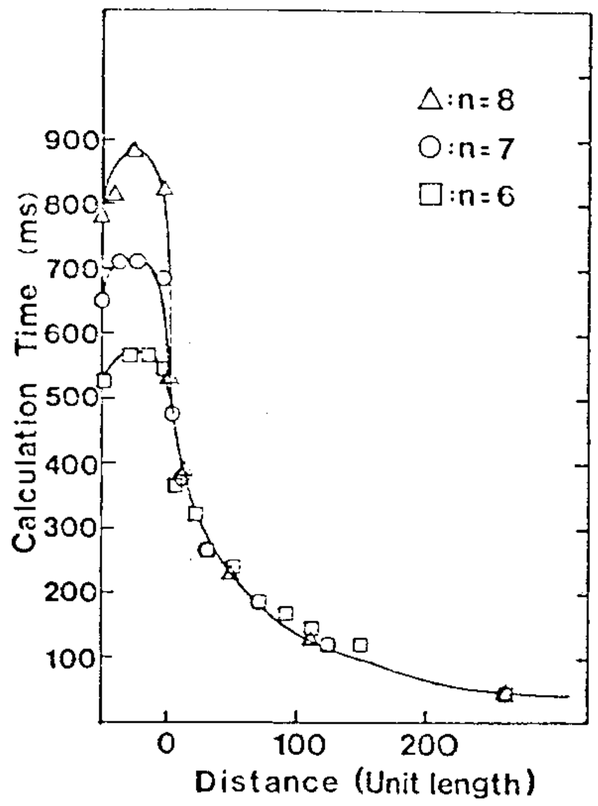


Fig. 11 Calculation time of the algorithm

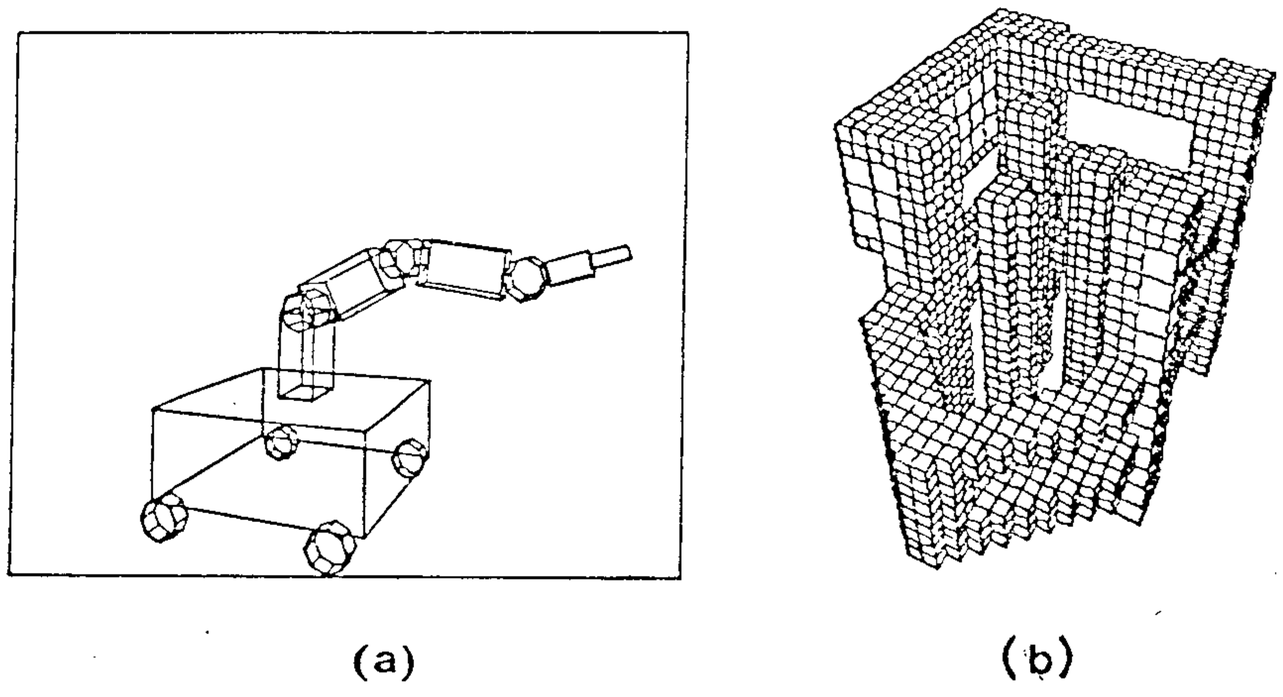


Fig. 12 Institute B : (a) Robot model (b) Environment model

リズムの高速性を確認するためのものである.

なお, オクトツリーのレベル n は環境モデルの解像度が全体空間の一辺を 2^n 分割した長さであることを表わす. 例えば, 全体空間の一辺の長さを 3 m とすると, $n=7$ の環境モデルの表現精度は約 2.3 cm (=300 cm/128) となる. なお, すべての実験は, 三菱電機 Melcom 350-60 (3.7 MIPS, 8 MB) を利用している.

(i) 設定A

4章で得たアルゴリズムの性質を確認するため, ここではロボットが環境物体に近づくと面積 S_k が連続的に変化するモデルを用いている. なお 100 面の多面体では, ロボットの形状としてはかなり複雑なもので, 計算時間がかかっている.

(a) ロボットモデル : 100 面の多面体で表現した球体 (半径 $r=50$) のロボット (**Fig. 10** (a)).

(b) 環境モデル : オクトツリー ($n=7$) で表現した直方体 [511-711, 0-512, 0-512] (全体空間 [0-1024, 0-1024, 0-1024]) からなる環境 (Fig. 10 (b)).

Fig. 11 は, ロボットモデルを環境モデル内の直方体に近づけたときのアルゴリズムの計算時間の推移である.

これから, 以下のことがわかる.

1) 物体間の距離によって, 計算時間が大きく変化する.

2) 環境モデルの解像度をあげても, アルゴリズムは環境物体を選択して干渉チェックしているので, 計算時間はさほど増加しない.

(ii) 設定B

ここでは, 提案したアルゴリズムが, 複雑な環境下でも高速であることを調べる.

(a) ロボットモデル : リンクや台車などの 12 個のコンポーネントを多面体 (総面数 100) で表現したマニピュレータ (**Fig. 12** (a)).

(b) 環境モデル : オクトツリー ($n=7$) で表現した 24 個の多面体 (総面数 138) からなる環境 (Fig. 12 (b)).

Fig. 13 の状態でのアルゴリズムの計算時間は 258.18 ms であるので, 環境物体の個数が多いという意味で複雑な環境下でも高速に干渉チェックできることがわかる.

6. むすび

環境を表現する面数 (または環境物体の個数) を N, ロボットを表現する面数 (またはコンポーネントの個数) を M とすると, 必要な計算手数は, 従来の方法では環境とロボットの個数やそれらの形状の複雑さに比例した値 $O(MN)$ であるが, 今回提案した方法では, ロボットの形状の複雑さのみに依存した値 $O(M)$ となる. また, このアルゴリズムの計算時間がロボットの位置によって変化することを Fig. 11 で示した.

このアルゴリズムにおける, ロボットと環境物体が共に存在する領域のみを解像度を上げながら選択することは, 環境物体やロボットをバウンディング・ボックス等で囲み, それらの間で干渉をラフチェックするような考え方を, 一般化かつ組織化したものとみることもできる. ロボットの動作教示では, ロボットと環境物体の極端な干渉はまれであり, むしろ非干渉であることの検証が多いことから, このような考え方で高速化がはかれる.

提案したアルゴリズムでは必要・十分条件を使って干渉・非干渉を決定するので, "架空" の干渉などで経路を見逃すことはない.

また, 現在はツリーを縦型探索することで "未定" キューブを選択し「逐次処理」している. しかし, 提案したアルゴリズムではすべてのキューブの処理が独立しており, 横型探索でそれらを「並列処理」するとアルゴリズムは格段に高速化できる.

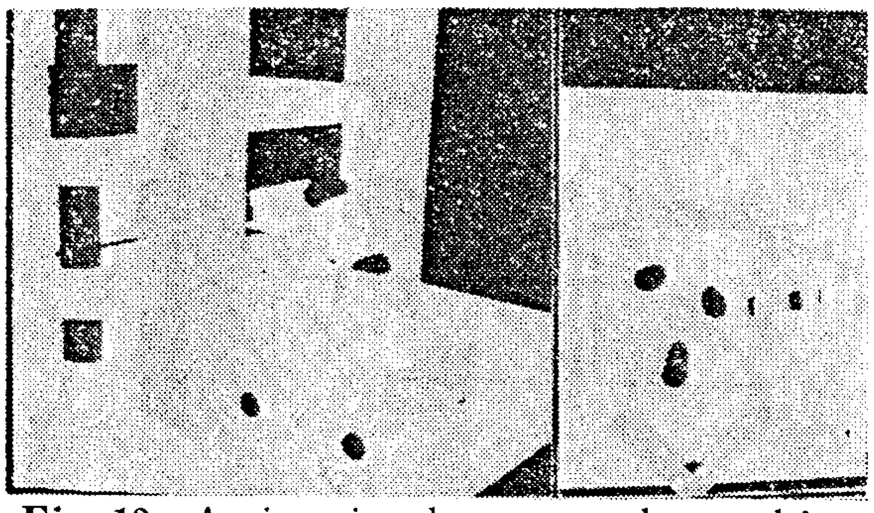


Fig. 13 A situation between robot and its environment

最後に，オクトツリーの白ノードのレベルは環境物体までの距離をある程度反映するので，干渉が起こらない場合でも，この方式では環境物体に接近しているロボット上の位置とそこから環境物体までの最短距離が容易に得られる．この情報は，ロボットの自動的な経路・動作決定に利用できる．

謝　辞

本研究に関して，日頃から貴重な御助言をいただく，三菱電機の竹垣盛一氏，大井　忠氏に感謝いたします．

参考文献

1) 尾崎他，“マニピュレータの占有空間を考慮した障害物回避動作の決定法”，計測自動制御学会論文集，Vol.18, No.9, pp.72-79, 1982
2) 尾崎，“マニピュレータの障害物回避”，日本ロボット学会誌，Vol.2, No.6, pp.76-82, 1984
3) J. W. Boyse, “Interference Detection Among Solids and Surfaces”, Communication of the ACM, Vol.22, No.1, pp.3-9, 1979
4) 重松他，“3-D形状モデル間干渉問題の一解法”精密機械，Vol.49, No.11, pp.89-94, 1983
5) 水垣他，“幾何モデルによるロボットの簡易干渉チェック”精密機械，Vol.50, No.6, pp.30-36, 1984
6) 小沢他，“オフラインロボット教示における高速干渉チェックの一方式”，日本ロボット学会誌，Vol.4, No.2, pp.5-14, 1986
7) D. Meagher, “Geometric Modeling Using Octree Encoding”, Computer Graphics and Image Processing 19, pp.129-147, 1982
8) C. L. Jackins and S. L. Tanimoto, “Oct-trees and Their Use in Representing Three-dimensional Objects”, Computer Graphics and Image Processing 14, Nov. 1980
9) V. Hayward, “Fast Collision Detection Scheme by Recursive Decomposition of a Manipulater Workspace”, IEEE International Conference on Robotics and Automation, Vol.2, pp.1044-1049, 1986
10) H. Noborio, et al., “A New Interference Check Algorithm Using Octree”, Proceedings of the 1987 IEEE International Conference on Robotics and Automation, North Carolina, 1987

Table 3 Process of the non-convex component

		Next state		
		outside	intersection	inside
Present state	outside	outside	intersection	inside
	intersection	intersection	intersection	inside
	inside	inside	inside	inside

【付録 1】

ここでは，コンポーネントのキューブ C 内に存在するすべての面（パッチ）を，重複を許して子キューブ C_i に分配する処理が基本となる．これはある面（パッチ）P を子キューブ C_i に重複を許して分配する処理を，すべてのパッチに逐次行なうと達成できる．

分配処理は3つの手続きからなる．

1) BBOX：パッチ P をとり囲む最小直方体が交差する子キューブ C_i を選択する．

2) PLANE：パッチ P を含む平面 S が交差する子キューブ C_i を選択する．

3) PROJECTION：パッチ P と子キューブ C_i の X, Y, Z 軸方向への正射影がすべて交差するような子キューブを選択する．

1),2) は面（パッチ）P が子キューブ C_i と交差する必要条件，3) は十分条件である．

[BBOX ルーチン]

まず，重心点 G を通る X, Y, Z 軸に垂直な平面により，キューブ C を2つの領域 (C_n, C_p) −4 つの子キューブ C_i に対応一に分けて考える．

次に，パッチ P をとり囲む最小直方体と重心点 G の座標を比較し，分割毎に最小直方体と2つの領域との交差を調べる（**Fig. 14**）．最後に，分割毎に得られた交差情報の論理積をとると，最小直方体と交差する子キューブが決定できる（**Table 2**）．

[PLANE ルーチン]

平面 S の法線ベクトルを $\vec{n}=(n_x, n_y, n_z)$，S 上の1点を $P_0(\vec{p}_0)$ とし，任意点 $P: \vec{p}=(p_x, p_y, p_z)$ に関して，$d(S,P)=\vec{n}\cdot(\vec{p}-\vec{p}_0)=n_x\cdot p_x+n_y\cdot p_y+n_z\cdot p_z-d$．$(d\equiv \vec{n}\cdot\vec{p}_0)$ なる関数を定義する．また平面 S で全空間を分割したとき B-reps.（コンポーネント）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 $d(S,P)$ は平面 S から P 点までの変位を表わしており，その符号が負ならば点 P は平面 S の内側に，正ならば点 P は平面 S の外側に存在する．

さらに，添字 i はキューブ C における子キューブ C_i の位置を表わす 10 進表現である．この 10 進表現の2進表現 $(r_0r_1r_2)_2$ からベクトル $\vec{i}=(r_0, r_1, r_2)$ を定義しそれを添字 i に対応づける．

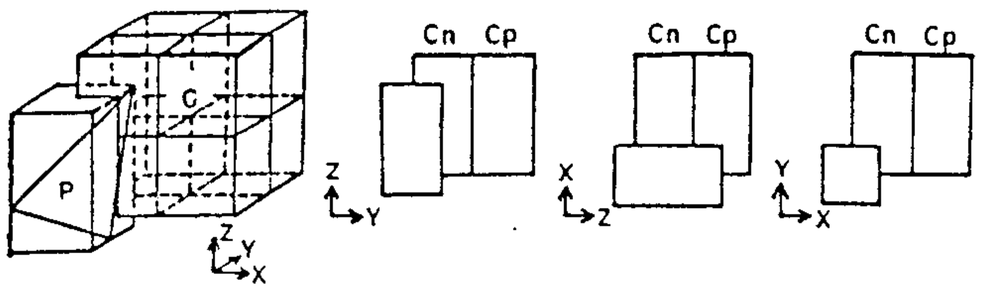


Fig. 14 Process of BBOX routine

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における $d(S,G)$ の正負より，以下の処理を行なう．

（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 $\vec{n},\vec{i}$ が最大［最小］となるベクトル $\vec{i}$ に対応する添字 i を k と表わすと，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 (**Fig. 15**)．そこで，点Gを除く C_k の7つの頂点 v_i の各々について前述の関数を計算する．関数 $d(S,v_i)>0$ $[d(S,v_i)<0]$ となると，頂点 v_i はSの外側［内側］に存在することとなり，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，$\vec{n}\cdot\vec{i}$ が最小［最大］となるベクトル $\vec{i}$ に対応する添字の子キューブはコンポーネントの“内側”［“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］

これまでのルーチンは，いずれも面（パッチ）がサブ領域（子キューブ）と交差するための必要条件を調べていたので，パッチと交差しない子キューブも得てしまう．このルーチンでは，その十分条件-パッチPと子キューブ C_i の X,Y,Z 軸に垂直な平面 $D(X),D(Y),D(Z)$ への正射影がすべて交差すれば，そのときのみパッチと子キューブは実際に交差する—を用いて，パッチと交差しない子キューブを取り除いている．ここでは，この十分条件の正当性を証明する．

まず，パッチ P，子キューブ C_i の平面 $D(X),D(Y),D(Z)$ への正射影を $P(j),C(j),(j=X,Y,Z)$ と，またパッチPが含まれる平面をSと定義する．また，C_iとSの交差領域を R，$D(j)$ 平面 $(j=X,Y,Z)$ への射影が $C(j)$ となる平面S上の領域を $R(j)\cup R$ と定義する (**Fig. 16**)．

また 2，3 次元での面（パッチ）とキューブの関係を以下のように定義する．

［3 D情報］(**Fig. 17**)．

［非交差］$C_i\cap P=\phi$.

［部分交差］$C_i\cap S\supsetneq C_i\cap P\neq\phi$.

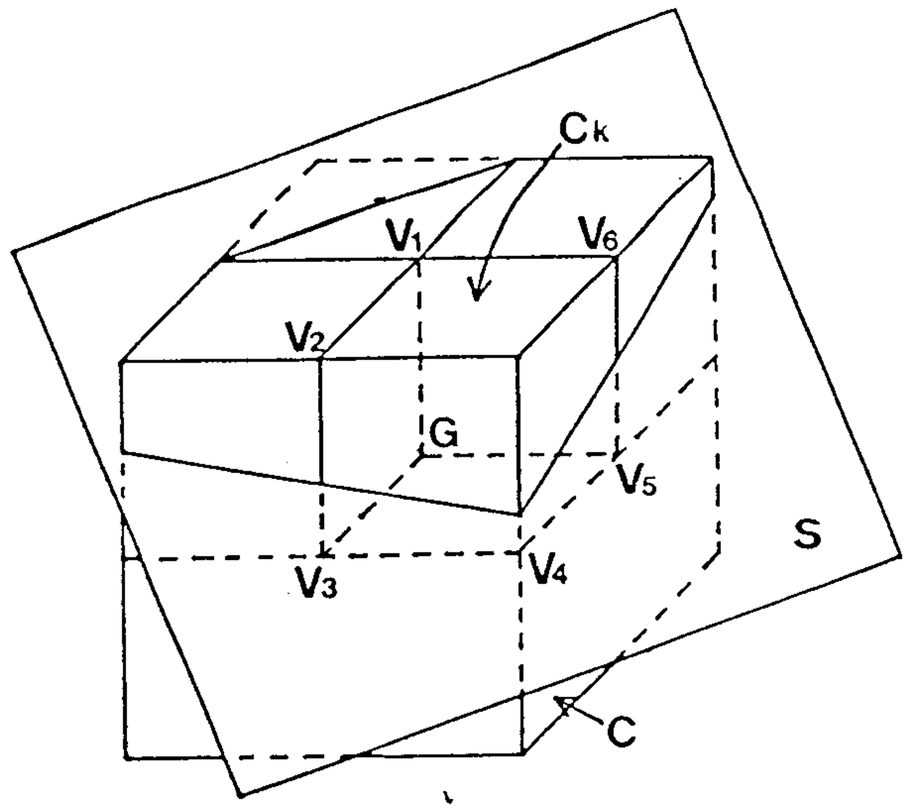


Fig. 15 Intersecting pattern between plane S and cube

［完全交差］$C_i\cap S=C_i\cap P\neq\phi$.

［2 D情報］(**Fig. 18**)．

［非交差］$C(j)\cap P(j)=\phi$.

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

［完全交差］$C(j)=C(j)\cap P(j)\neq\phi(j=X,Y,Z)$.

ここで，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 $C(j)\cap P(j)\neq\phi\Longleftrightarrow C_i\cap P\neq\phi$

［証明］(→) どの平面 $D(j)$ においても，$C(j)\cap P(j)\neq\phi$ となることは，$p_0(j)\in R(j)\cup R(j=X,Y,Z)$ となるパッチP上の点 $p_0(j)$ が存在することを意味する．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ち，パッチPを予め凸形状に分割しておくと領域Qはその内部に存在する (Fig. 16)．したがって，$C_i\cap P\neq\phi$ がいえる．□

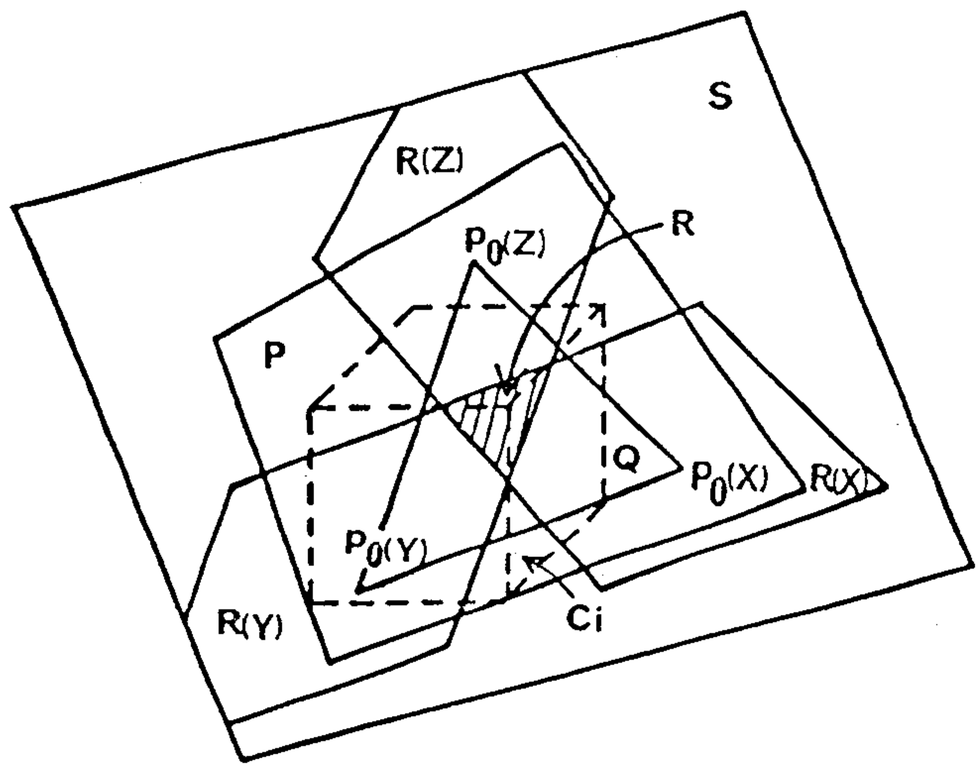


Fig. 16 Projecting regions of cube onto the plane S

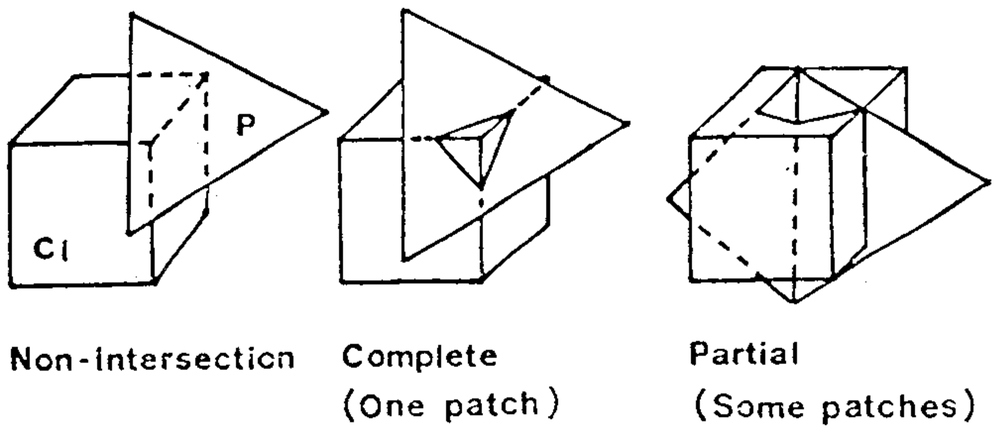


Fig. 17 Three dimensional intersecting pattern between patch and cube

(←) 点 $p \in C_i \cap P \neq \phi$ の平面 $D(j)$ への正射影を $p_1(j)$ とすると，明らかにすべての平面 $D(j)$ で，点 $p_1(j) \in C(j) \cap P(j)$ が存在するので，$C(j) \cap P(j) \neq \phi$ が成りたつ．□

［定理2］

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

［証明］(→) 定理1における (←) の証明の対偶．□

(←) 定理1における (→) の証明の対偶．□

【付録 2】

コンポーネント (B-reps.) が凸形状のとき，入力キューブ内の“非交差”子キューブは以下の性質をもっている．

性質 1　コンポーネントを構成する面が重心点Gを通らない限り，キューブC内の“非交差”子キューブの内外は同一である．そして，キューブCの重心点Gが内側（外側）なら，“非交差”子キューブ C_i は“内側”（“外側”）である．□

性質 2　キューブC内のすべてのパッチに[PLANE]ルーチン（付録1）を用いたとき，1つのパッチでも重心点Gを外側と判断すればGは外側，すべてにおいて重心点Gを内側と判断すればGは内側である．そうでなければコンポーネントを構成する面が重心点Gを通るので，[PLANE] (ii) から得られる内外情報より“非交差”子キューブの内外が決定される．□

これらの性質より，“非交差”子キューブのコンポーネントに関する内外も，その親キューブ内のパッチの情報だけから決定されることがわかる．

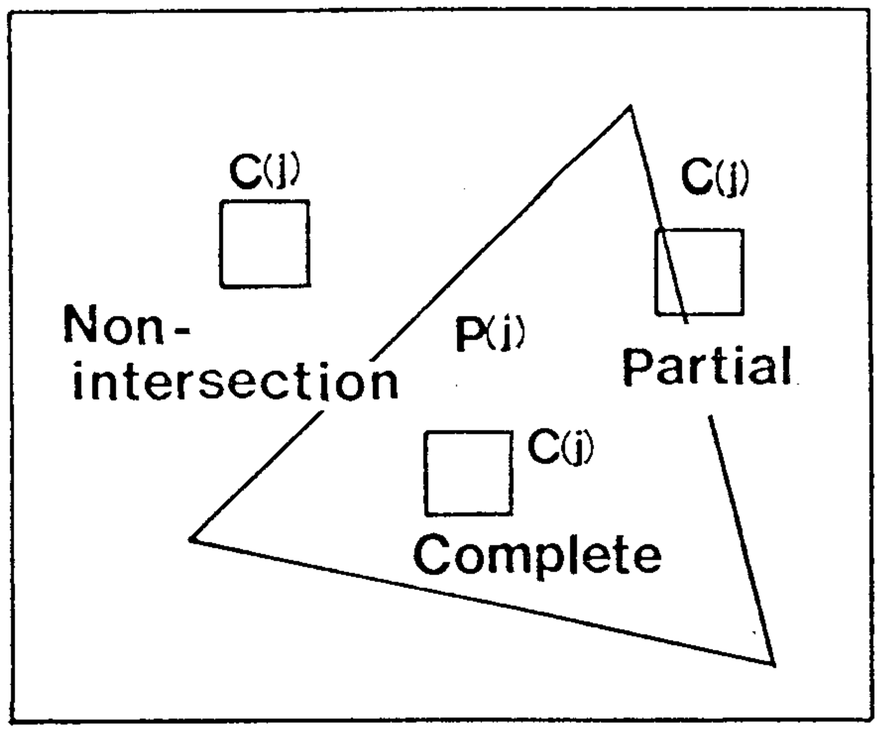


Fig. 18 Two dimensional intersecting pattern between patch and cube

このことから，ツリーをトップダウンに探索する提案したアルゴリズムでは，すべてのキューブの処理が独立しており，ある領域（キューブ）を処理する・しないが自由に選択できる．

【付録 3】

面を $f = n_x*x + n_y*y + n_z*z - d (n_z \geqq n_x, n_y)$ とし，ブロック（辺の長さ l）の対角に位置する2頂点を(x_0, y_0)，(x_1, y_1) とする．

$n_x*x_0 + n_y*y_0 + n_z*z_0 - d = 0.$

$z_0 = (d - n_x*x_0 - n_y*y_0)/n_z.$

$n_x*x_1 + n_y*y_1 + n_z*z_1 - d = 0.$

$z_1 = (d - n_x*x_1 - n_y*y_1)/n_z.$

$0 \leqq |z_1 - z_0|$

$= |(x_0 - x_1)*n_x + (y_0 - y_1)*n_y| / |n_z|$

$\leqq (|x_0 - x_1|*|n_x| + |y_0 - y_1|*|n_y|)/|n_z|$

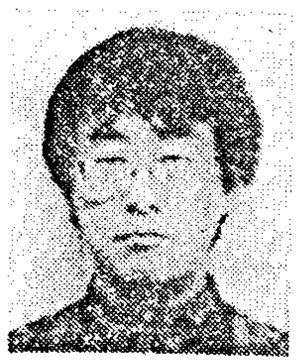
$= l*(|n_x| + |n_y|)/|n_z|.$

$0 \leqq |z_1 - z_0|/l \leqq (|n_x| + n_y|)/|n_z|.$

$1 \leqq$ (平均交差キューブ数)

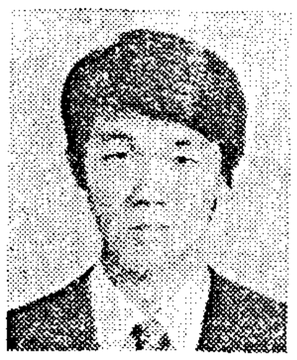
$= |z_1 - z_0|/l + 1$

$\leqq (|n_x| + |n_y|)/|n_z| + 1 = 3.$ □



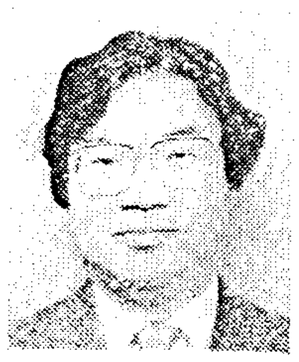
登尾啓史 (Hiroshi NOBORIO)

昭和 33 年11月4日生れ．57 年静岡大学工学部情報工学科卒業．59 年同大学院工学研究科修士課程修了．現在大阪大学大学院基礎工学研究科博士課程在学中．電子通信学会，情報処理学会，各会員．(日本ロボット学会学生会員)



福田尚三 (Shozo FUKUDA)

昭和 38 年9月 12 日生れ．61年大阪大学基礎工学部機械工学科卒業．現在大阪大学大学院基礎工学研究科修士課程在学中．



有本 卓 (Suguru ARIMOTO)

昭和11年8月3日生れ．34 年京大理学部卒．沖電気工業K．K．，東大工学部助手，講師を経て 43 年大阪大学助教授，基礎工学部機械工学科に勤務，48 年同教授．工学博士．ロボティックス，ディジタル信号処理，制御理論等の研究教育に従事．日本機械学会，計測自動制御学会，情報処理学会，等の各会員．IEEE の Fellow 会員．(日本ロボット学会正会員)

A New Interference Check Algorithm Using Octree Representation*

Hiroshi NOBORIO** Shozo FUKUDA**Suguru ARIMOTO**

ABSTRACT

A fast and general interference check algorithm which does not depend on the complexity of shape of an environment for a robot but on the distance between the robot and its nearest obstacle in the environment is proposed, which is based on the adoption of B-reps. as a model of the robot and octree as a model of the environment.

The octree is a natural hierarchical solid model that can change the resolution in positioning adaptively in reference to a region in the world space and is adequate to the environment model. Further, the B-reps. can represent easily a complex motion including rotation of an object by every time updating its coordinates table and is adequate to the robot model.

The algorithm consists of a basic process that assigns efficiently a patch of the B-reps. within a region to eight subregions when the region is divided into them. Then, this division is guided by the hierarchical structure in positioning of the octree. With the aid of both information for a region induced by the assignment and spatial information inherent in the region, the algorithm can fastly select only regions that intersect simultaneously two models.

From this selection, it follows that the interference check algorithm deals with only parts of obstacles which lie around the robot model and its computational complexity does not depend on the complexity of shape of the environment model.

Finally, the computational complexity of the algorithm is evaluated, and the reasonableness of the evaluation and the efficiency of the algorithm are further ascertained by several experiments.

Key words : Off-line teaching, Interference check, Octree representation, Graphics simulation, Hierarchical structure.

* Received October 31, 1986

** Department of Mechanical Engineering, Faculty of Engineering Science, Osaka University