アルゴリズム 63－7
（1998．7．22）

分散メモリMIMD型並列計算機を用いた3次元物体の衝突面検出

二宮 茂樹, 渋沢 進

茨城大学工学部情報工学科
〒316-8511 日立市中成沢町 4-12-1

3次元物体の衝突面検出アルゴリズムとして、衝突する可能性をもつ面を絞っていき、最後に正確に面ペアの衝突検査をする方法がこれまで提案されてきた。逐次処理の実験結果より、最後の処理がこのアルゴリズムで大きな比重を占めていることが分かった。そこで本研究では、分散メモリMIMD型並列計算機を用いて、面の衝突検査を並列化する実験を行った。その際、ホストプロセッサから各セルプロセッサに順次一定数の面ペアを送り、各セルで送られてきた面ペアの衝突を並列に検査する方法を用いた。送る面ペア数として、5・10・15・20・25・30の場合を実験した。この結果、10台のセルを用いて各セルに30個の面ペアを送ったとき、逐次処理の58.5％の実行時間で面ペアの衝突が検査できた。

A Parallel Colliding Face Detection of 3D Objects Using a Distributed-Memory MIMD Computer

Shigeki Nimiya, Susumu Shibusawa

Dept. of Computer and Information Sciences, Faculty of Engineering, Ibaraki University
Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan

For the colliding face detection of 3D Objects, an algorithm is proposed which decreases the search range of colliding faces using octree. From the sequential implementation of algorithm, the collision check of pair-faces takes the major part of execution time. In this paper, we test the parallel check of pair-face collision using a distributed-memory MIMD computer. The host processor sends the fixed number of pair-faces to cell processors in order, and each cell processor checks the collision of pair-faces. As the result of experiment, the execution time of parallel collision check of pair-faces is reduce to 58.5% of that of sequential processing for two pheres with each 162 faces, 30 pair-faces to be sent and 10 cell processors.

1 はじめに

3次元物体の衝突問題は、ロボティックス、コンピュータグラフィックス等多くの分野で中心的な問題の一つである[1][2][3]。また、仮想現実環境における物体操作シミュレーションでは、発生する物体間の衝突を見逃すことなく、実際に衝突が起きる直前に正確にこれを検出することが必要となる。特に、複雑な形状を持つ物体間の衝突を正確に検出する問題は、計算量が多く実時間で処理することが困難である。正確に物体間の衝突を検出しようと考えるとき、例えば、作業空間中の物体の全ての面と稜線の組合せに対し、交差の有無を調べるという方法が考えられる。しかしこの方法では、物体の数や物体の面数が多くなるにつれ、衝突面検出の計算量は大きく増加してしまう。

3次元物体の衝突面を検出する方法としては、離散時間の衝突検査で複雑な軌道をもつ物体間の衝突面を見逃すことなく、実際に衝突が起きる直前に衝突面を特定する衝突面検出アルゴリズムが開発されてきた[4]。このアルゴリズムは、8分木を用いて衝突する可能性のある面を絞っていき、最後に正確に衝突面を調べるというものである。この手順により、調べなければならない面の数を大幅に削減することができる。また[4]では、共有メモリを用いて、並列アルゴリズムを実現している。

本研究では、3次元物体の衝突面検出を並列化し、分散メモリMIMD型並列計算機を用いて、衝突面検出実験を行った。衝突面検出のアルゴリズムは、四つのステップから成っており、この内の最後の正確な面ペアの衝突検査を並列化することで処理時間の高速化をはかる。並列化は、ホストプロセッサから各セルプロセッサに順次一定数の面ペアを送り、各プロセッサで送られてきた面ペアの衝突を並列に検査した。

2. では、3次元物体の衝突面検出アルゴリズムを述べる。**3.** では、その逐次アルゴリズムを実現しその結果をまとめる。**4.** は、逐次アルゴリズムの並列化について述べる。**5.** では、これらのアルゴリズムを基に実現したプログラムの実験結果を示す。最後に実験結果について評価、考察を行う。

2 衝突面検出のアルゴリズム

作業空間中の物体は全て多面体で表現されている。物体の形状は、どのような形でも良いとする。また、物体はある時間間隔Δtごとの離散時刻$(\cdots,t_{i-1},t_i,t_{i+1},\cdots)$において移動し、このとき各物体の頂点位置と移動量が分かっているとする。

8分木を用いた衝突面検出を次のように行う。

1. 外接直方体の重なり領域の検出
2. 重なり領域と交差する候補面の抽出
3. 8分木を用いた面ペアの抽出
4. 面ペアの衝突検査
5. 1〜4を1離散時間ごとに行う

以下に具体的な手順を述べる。

2.1 step1—外接直方体の重なり領域の検出

各物体に対して外接直方体を生成し、これらの重なり領域を検出する。

具体的な方法としては、まず、物体の頂点座標値を用いて、x,y,z軸方向の最大値、最小値から各外接直方体の頂点座標値を求める。物体とその外接直方体を図1に示す。図において、$*_{min}$は$*$軸方向の最小値、$*_{max}$は$*$軸方向の最大値である。次に求めた頂点座標値を用いて、全ての外接直方体の組について重なり領域を調べる。外接直方体の重なり領域を図2に示す。

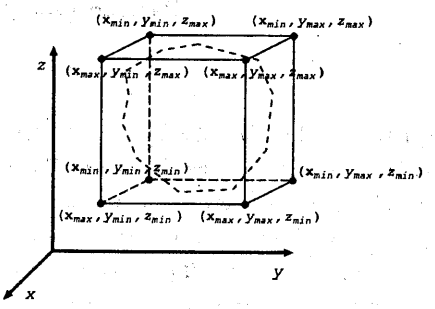

図1: 物体の外接直方体の頂点座標値

このステップでは、次の操作を行う。

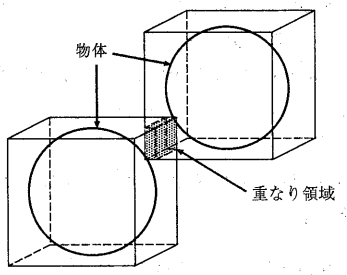

図 2: 物体の外接直方体とその重なり領域

- 重なり領域が見つかったなら、その物体の組と重なり領域をリストに蓄え、次のステップへ
- 重なり領域が見つからないなら、衝突検出処理を終了し、次の時刻へ

2.2 step2—重なり領域と交差する候補面の抽出

step1 より得られたリストから対の物体と重なり領域を一組ずつ取り出し、各物体を構成している全ての面と重なり領域との交差を調べる。この交差は、重なり領域に含まれるかまたは交差する面とし、候補面と呼ぶ。重なり領域と交差している候補面の例を図 3に示す。候補面が見つかれば、その面を候補面リストに蓄える。

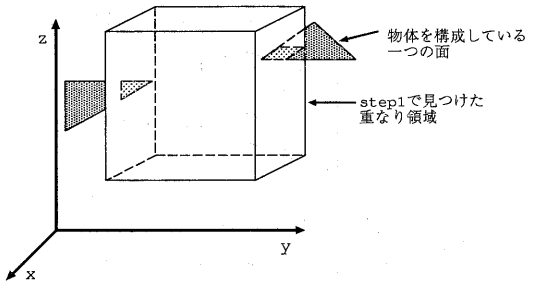

図 3: 重なり領域と交差する面

重なり領域と面の位置関係を図 4に示す。交差する面を見つけるおおまかな手順は、次のようである。

1. それぞれの面を構成している頂点が、重なり領域の内部にあるか調べる。
2. 内部にあればその面は重なり領域と交差すると判断する (図の (i))。
3. 内部に無い場合は、その面の稜線と、重なり領域を構成している平面との交点を求める。
4. 交点が重なり領域を構成している面内にあれば、その面は交差すると判断し (図の (ii))、面内に無ければ交差しないと判断する (図の (iii))。

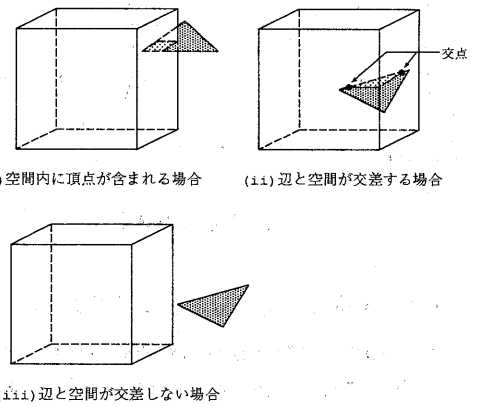

図 4: 重なり領域と面の関係

またこのとき、候補面が一つの物体からのみ抽出された場合は、衝突は起きないと判断する。このステップでは、次の操作をする。

- 候補面が二つ以上の物体から抽出されたなら、次のステップへ
- 交差する面が見つからないかもしくは一つの物体からのみ抽出されたなら、衝突検出処理を終了し、次の時刻へ

2.3 step3—8 分木を用いた面ペア検出

step2 で得られた候補面リストに含まれている面が作業空間中に存在する位置を、深さ lの 8 分木で表す。8 分木は 3 次元空間の 8 分割の繰り返しを表す。8 分木の構成を図 5に示す。8 分木は、次のように生成する。

- ルートノードを作業空間全体とする一辺の大きさ L とした立方体とし、順次それを 8 分割した木で表現する。

- 最下層のノードの大きさは、面が1離散時間内に移動する量よりも十分大きいとする。

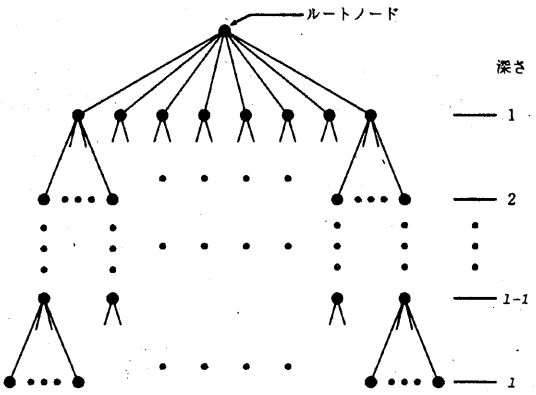

図 5: 8分木の構成

面ペアを検出する手順は次のようである。

1. 二つ以上の物体からの候補面が含まれている場合に限り、そのノードを八つの子ノードに分割し、候補面との交差を調べる。
2. 最下層の各ノードに対して、異なる物体からの面の組合せを重複がないように選びだし、面ペア検査リストに書き込む。
3. 異なる物体からの面の組合せが見つかったなら、次のステップへ
4. 異なる物体の面の組合せが見つからないなら、衝突検出処理を終了し、次の時刻へ

2.4 step4—面ペアの衝突検査

任意の時刻 t_i において、時間 $[t_i, t_{i+1}]$ に各面によって構成される物体が交差していれば、これらの面は時刻 t_i と t_{i+1} の間に衝突すると判断される。step3で得られた面ペア検査リストから面ペアを一つずつ取り出し、次の手順を行う。

1. 取り出した二つの面それぞれについて、現在面がいる時間 t_i の位置と1離散時間後 t_{i+1} の面の位置から、面の凸包を作る。二つの面が作る凸包を図6に示す。
2. 次にこの二つの凸包の間の交差を調べる。各々の面と稜線の組合せに対して、これらの交わり方を調べ、次の三つのうち (3) の組を見つけることにより交差を検出する。
   (1) 稜線の両端点が面に対して同じ側にある。
   (2) 稜線と面を含む平面との交点が面の外部にある。
   (3) 稜線と面を含む平面との交点が面の内部にある。

上の三つの場合を図7に示す。

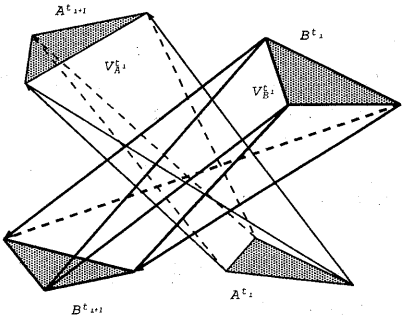

図 6: 二つの面からできる凸包

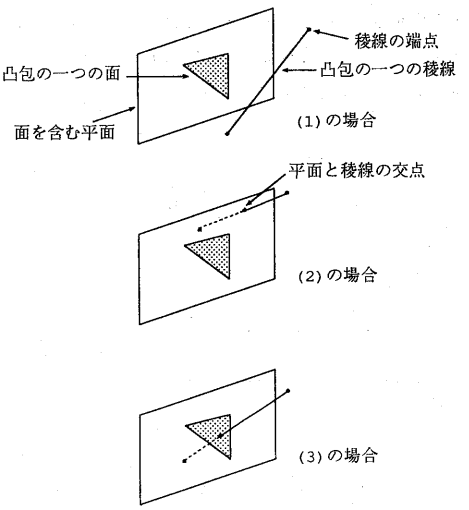

図 7: 面と稜線の関係

このステップでは、step3により得られた面ペア検査リストから面ペアを一組ずつ取り出し、次の時刻の頂点位置を求めて、面の凸包を得る。そして、以

下の手順を面ペア検査リストに蓄えられている全ての面ペアについて行う。

まず、面ペアのそれぞれの凸包によってできる稜線の両端点が、面を通る平面に対してどちら側にあるか調べる。

- 両端点が同じ側にない (図 7(2),(3)) なら、平面と直線の交点を求める。
- 両端点が同じ側にある (図 7(1)) なら、稜線と面の別の組合せについて調べる。

求めた交点について、次の操作を行う。

- 交点が面内にある (図 7(3)) とき、この面ペアを面ペアリストに蓄え、次の面ペアについて調べる。
- 交点が面内にない (図 7(2)) とき、稜線と面の別の組合せについて調べる。

以上の操作を、面ペアから生成される互いの凸包が交差しない間、稜線と面の全組合せについて行う。全組合せについて交差しない場合、この面ペアは衝突しないと判断する。そして、次の面ペアについて調べる。

3 逐次的な衝突面検出

3.1 逐次プログラムの実行

まず、**2.** で説明した衝突面検出アルゴリズムの逐次プログラムを作り、実行時間を測定した。後の並列プログラムの実験と比較評価できるように、この逐次プログラムは次節で用いる並列計算機の一つのプロセッサによって実行する。

作成したプログラムは、任意の 1 離散時刻での 3 次元物体の状態について、その 1 離散時間後の位置を考慮して、衝突面を検出するものである。そのため、物体の状態を決め、1 離散時間の移動量を決めておく必要がある。

この実験では、二つの球体が衝突するとき (実験 1) と、二つのある領域内で生成した面の集合の衝突 (実験 2) の二つの場合の衝突面検出を行った。次に設定した値を示す。

- 実験 1 は一方は半径 50、中心 (50,50,50)、他方は半径 50、中心 (100,100,100) の球体を用いて、実験 2 は、約 $100 \times 100 \times 100$ の領域内で生成した面を用いた
- 各球体は三角形で構成された面を持ち、実験 1 の球体の面数は 162、実験 2 の領域内の面数は 100
- 物体の 1 離散時間の移動量は、一方の物体は x, y, z 軸の正の方向に 3 移動、他方の物体は x, y, z 軸の負の方向に 3 移動
- 作業空間は $400 \times 400 \times 400$ の 3 次元空間
- 衝突面検出アルゴリズムの step3 で用いる 8 分木の深さ $l=4$

このとき、8 分木のルートノードは一辺の大きさ L=400 の立方体となり、最下層のノードは $25 \times 25 \times 25$ の立方体となる。また実験 2 については、step1 の重なり領域の大きさが異なる二つの場合の実験 (実験 2 (a) と実験 2 (b)) を行った。実験 2 (a) の方が実験 2 (b) より重なり領域が大きい。

まず、以上のような状態において、各実験で step2, 3, 4 と処理を行っていくときに、衝突する可能性のある面ペアが絞られていった過程を表 1に示す。

step1 は外接直方体の重なり領域を求める処理なので、表には書かれていない。step2 での値は、一方の物体から求められた重なり領域と交差する面の数と、他方の物体から求められた重なり領域と交差する面の数を掛けたものである。これは、step2 で得られた衝突する可能性のある面ペア数を考えたとき、一方の物体から得られた各面に対して、他方の物体の全ての面が衝突する可能性を持つことになり、衝突する可能性を持つ面ペア数は、一方の面の数と他方の面の数を掛けた数になるためである。step3 での値は、8 分木を用いて求めた面ペアの数である。step4 での値は、衝突すると判断された面ペアの数である。

表 1: 衝突する可能性のある面ペア数が絞られていく経過

	実験 1	実験 2 (a)	実験 2 (b)
step2	1932	1638	928
step3	355	1348	727
step4	140	280	210

各実験の実行結果を表 2に示す。実行時間は、3 回測定しその平均とした。表の比率は、各ステップの実行時間の全実行時間に占める割合である。実行

時間は小数点第5位で、また比率は小数点第2位で四捨五入している。

表 2: 逐次処理の実行結果 (実行時間:秒, 比率:%)

	実験1		実験2 (a)		実験2 (b)	
	実行時間	比率	実行時間	比率	実行時間	比率
step1	0.0119	1.3	0.0077	0.2	0.0077	0.4
step2	0.1077	12.2	0.0299	0.7	0.0343	1.6
step3	0.0745	8.5	0.0717	1.8	0.0648	3.1
step4	0.6839	78.0	3.9592	97.3	1.9870	94.9
計	0.8780	100.0	4.0684	100.0	2.0938	100.0

3.2 考察

今回の実験で衝突した状態を用いたのは、次のような理由による。この衝突面検出のアルゴリズムでは、衝突しない場合はstep1や、step2、step3で処理が終了するが、最も実行時間がかかるのは衝突面が検出される場合、つまりstep4まで処理を行った場合である。このときの実行時間の短縮を目指すために、実験では衝突した状態を用いた。

表2の実行時間を見ると、どの実験もstep4は他のstepに比べて、最も実行時間がかかっている。また、表の比率から分かるように、この衝突面検出アルゴリズムの中で大きな比重を占めている。

このことから、step4を並列化し高速に処理することで、大幅な実行時間の短縮につながることが分かる。

4 並列化

衝突面検出のstep4は、面ペア検査リストに含まれている面ペア一つ一つに同様の処理を行っている。そこで、面ペア検査リストの面ペアを複数のプロセッサに割り当てて処理することで、実行時間の短縮をはかることを考えた。

4.1 使用した並列計算機

本実験では、非同期メッセージパッシングをベースとする分散メモリMIMD型並列処理コンピュータAP1000を用いた。本体にはホストと呼ばれるフロントエンドプロセッサが接続されていて、プログラムの実行はホストから行う。

本実験で用いたAP1000はセル数が16であり8×2の構成になっている。各セルは25MHzのSPARCプロセッサで、メモリを16Mbyte持っている。

AP1000は3種類の結合構造を持っている。S-Netはホストと全セル間での同期をとるための木構造のネットワークであり、B-Netはホストとセル間の1対多通信を行うためのリングネットワークと階層バスから成るネットワークである。また、T-Netは任意のセル間の1対1通信を行うための2次元トーラス構造のネットワークである。

AP1000では、ホストで実行するプログラムとセルで実行するプログラムの2種類のプログラムを用意する必要がある。ホストプログラムでは、セルの実行環境、セルで実行するタスクの生成、セルへのメッセージの送信、セルからのメッセージの受信、セルの状態監視などを行う。セルプログラムでは、ホストやセル間で通信しながら、セルで並列に実行させる処理を行う。

4.2 並列アルゴリズム

ホストが衝突面検出のstep1～3を実行し、step3で得られた面ペア検査リストの中から、一定数ずつ各セルへ送る。その際、決められた面ペア数を一度に送れるように、step3では一定数の面ペアを一つの組として面ペア検査リストをつくる。今回の実験は、各セルに一度に送る面ペア数が5・10・15・20・25・30という六つの場合について行った。

ホストプログラムとセルプログラムの手続は、次のようである。

ホストプログラム：

```
procedure parallel_CFD_at_host
  1. while  セルへ送っていない面ペアリストがある
  2.    do  決められた数の面ペアをセルに順に送る
  end while
  3. 各セルから送られてきた衝突する面ペアを受け取る.
```

セルプログラム：

```
procedure parallel_CFD_at_cell
  1. while  決められた数の面ペアが送られてくる
  2.    for  送られてきた面ペアの数だけ繰り返す
  3.      if  面ペアが衝突する
  4.        then  衝突する面ペアのリストに蓄える
          end if
        end for
  end while
```

5. リストに蓄えた面ペアをホストに送る.

5 実験結果

衝突面検出のstep3で得られた面ペアを、実験1では5・10・15・20・25・30ずつ、実験2では5・10・15・20・25ずつセルに送ったときの実行結果を図8～10に示す。図中のsend5～send30はそれぞれ各セルへ一度に送る面ペアの数が5～30のときの結果を示している。実行時間は、ホストからデータを送り、セルからデータが返ってくるまでの時間である。またこの実行時間は、3回実験したときの平均値である。

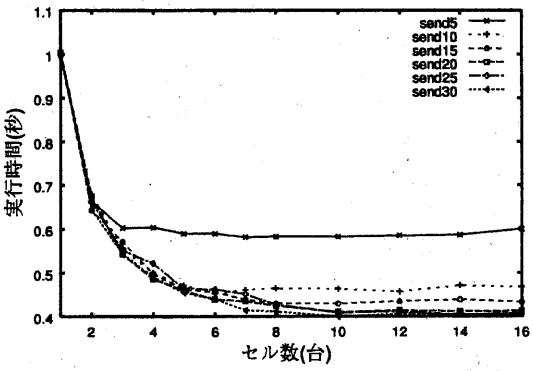

図 8: 実験1の実行時間とセル数の関係

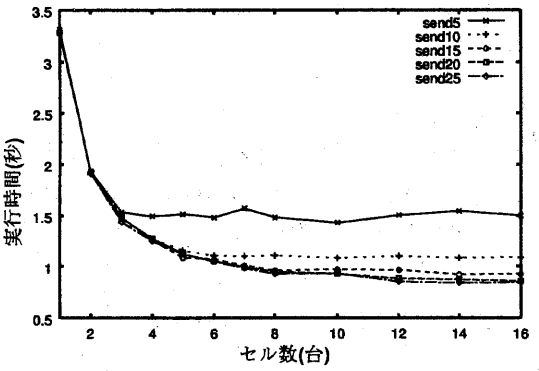

図 9: 実験2 (a) の実行時間とセル数の関係

図8～10をみると、それぞれ一定数送ったときセル数が増えるにつれ実行時間が減少しているので、

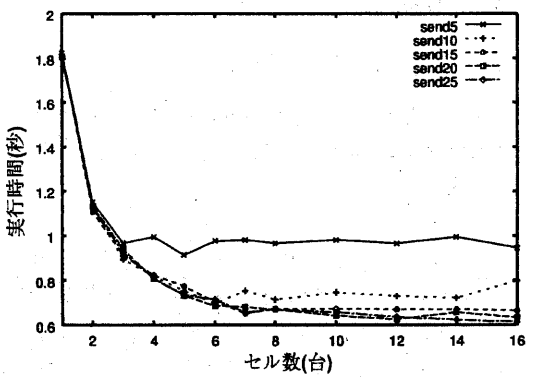

図 10: 実験2 (b) の実行時間とセル数の関係

並列化の効果が現われていることが分かる。send5に注目すると、セル数が3程度になると実行時間はそれ以上良くならない。これは、ホストが各セルへ5個ずつの面ペアを送る時間より、セルが5個の面ペアを処理する速度のほうが速くなってしまうため、セルが何もしない時間が生まれてしまうためだと考えられる。このことから、ある程度多くの面ペアを一度に送った方が実行時間が減少する。

図8からsend5～send30の最小の実行時間をみると、送るペア数が増えるにつれて、つまり5と10、10と15、15と20、20と25、25と30を比べて、実行時間は減少しているが、5から10へ、10から15へ、15から20へ、20から25へ、25から30への実行時間の減少率は段々小さくなっている。このことから、実行時間の減少に限界があることが分かる。

また、send5～send30について、セル1～3までは同様な実行時間の減少を示している。これは、セル1～3のときは、実行途中に何もしない時間を持つセルが生まれず、全てのセルが働き続けているため、一度に送る面ペア数が変わっても、セルで処理できる面ペア数は同じであり、実行時間が同様になると考えられる。

またこのようなことは、実験2の結果の図9, 10についても言える。

6 考察

6.1 逐次処理との比較

実験1と実験2より得られた最も速い実行時間と逐次処理の実行時間の比較を、表3に示す。表の%は、比率を表す。

表 3: 逐次処理と並列処理の実行時間の比較

	実験1	実験2 (a)	実験2 (b)
逐次処理	0.6839	3.9592	1.9870
並列処理	0.4002	0.8374	0.6124
%	58.5	21.1	30.8
使用セル数	10	14	16

この結果と表1から、step4で処理するべき面ペアの数が多いほど、並列化の効果が現れている。このことから、実現した衝突面検出アルゴリズムの並列化は、処理する面数が多いほど実行時間の短縮が図れることが分かる。これは用いる並列計算機の仕組みにも関係し、今後更に検討が必要である。

6.2 並列実験の評価

並列実験の結果より得られた図8から、セルをある程度増やしていくと実行時間の改善が起こらなくなる。これは、一つのセルが送られてきた一定数の面ペアの衝突検査をする時間より、ホストから全セルへ一定数の面ペアを送る時間の方が時間がかかるようになるためだと考えられる。実行時間の改善が行われなくなる最小のセル数をp_0とすると、p_0より多いセル数では実行時間が一定である。また、ホストからセルへの面ペアの送信を考慮すると、p_0より少ないセル数ではプロセッサ数に反比例して実行時間が減少する。このことから、実行時間Tに関して次のような関係が成り立つ。

$$T = \begin{cases} t_0 + \frac{t_1}{p} & (p < p_0 \text{のとき}) \\ t_2 & (p \leq p_0 \text{のとき}) \end{cases}$$

ここで、pはプロセッサ数、t_0, t_1, t_2は定数である。$p = p_0$となるときは、一つのセルで一定数の面ペアを処理する時間と次の一定数の面ペアが送られる時間が同じとなるときである。このときは、その用いたセル数において最も効率の良い場合だと考えられる。

7 おわりに

本研究では、分散メモリMIMD並列計算機AP1000を用いて、3次元物体の衝突面検出のアルゴリズムの並列化を行った。まず、逐次処理の結果より、衝突面検出のstep4の面ペアの衝突検査の処理がこのアルゴリズムで大きな比重を占めていることが分かり、step4の並列化を行った。並列化は、ホストプロセッサから各セルプロセッサに順次一定数の面ペアを送り、各セルプロセッサで送られたきた面ペアの衝突を並列に検査することで実現した。この並列実験の結果から、ホストプロセッサから各セルプロセッサに一度に送る面ペアの数が小さいとき、セル数を増やしても、実行時間の改善ができなくなっている。そのため、ある使用セルプロセッサ数に対して、送る面ペア数に最適値があることが考えられる。

今後の課題としては、今回の実験結果を定式化することで、最も効率の良くなる送る面ペアの数を出し、またそのときのセル台数を求めることが挙げられる。また、他の並列計算機でアルゴリズムを実現することで、性能の違いによる結果の相違を見ることや、異なるアーキテクチャに適した並列化を実現することなども挙げられる。

謝辞

本研究に際し、ご討論頂いた茨城大学工学部の研究室の皆様に深く感謝致します。

参考文献

[1] M. de Berg, M. van Kreveld, M. Overmars and O. Schvargkapf: Computational Geometry - Algorithms and Applicatins, Splinger, 1997.

[2] F.P. Preparata and M.I. Shamos(著), 浅野孝夫, 浅野哲夫 (訳):計算幾何学入門, 3章, 総研出版, 1992.

[3] 電子情報通信学会 (編):, ニューメディア技術シリーズ コンピュータグラフィックス, オーム社, 1987.

[4] 北村喜文, Andrew Smith, 竹村治雄, 岸野文郎: 並列計算機による3次元物体の実時間衝突面検出, 電子情報学会論文誌 D-I, Vol.J78-D-I, No.8, pp788-797, 1995.