超並列シミュレーションと
ビジュアライゼーション

堀口　進, 井口　寧, 山森　一人, 林　亮子, 加藤　聡

23/ May/ 1997
IS-RR-97-0023

北陸先端科学技術大学院大学
情報科学研究科
〒923-12 石川県能美郡辰口町旭台 1-1
hori@jaist.ac.jp

ISSN 0918-7553

要旨

コンピュータ・シミュレーションは、様々な科学技術分野で用いられ実用規模の数値模擬実験には巨大なメモリ空間と膨大な計算時間が必要とされている。現在、この分野ではスーパーコンピュータに代わって多数の高速プロセッサからなる超並列コンピュータが注目され、最先端分野の大規模シミュレーション法および可視化に関する研究が切に望まれている。

本研究では、基礎物理・化学・物性材料設計・流体問題や神経回路学習などの最先端分野におけるコンピュータ・シミュレーションを疎結合型超並列コンピュータで実行する超並列シミュレーションとその可視化について詳しく検討してきた。その結果、超並列コンピュータは、その巨大実メモリと多数の高速プロセッサにより、従来困難であった大規模シミュレーションが出来ることが明かになった。例えば、物理・化学分野での分子の振舞いをシミュレーションする分子動力学法では、分子数が数千個に限られていた物を数万個に容易に拡張できる。また、シミュレーションデータの可視化により気体から液体への相移転などの分子の振舞いを確認できた。更に、超並列シミュレーションの高速化については、プロセッサ間ネットワーク、メッセージパッシング、データ配置を考慮した動的負荷分散並列シミュレーション・アルゴリズムの提案を行ないその有用性を確認した。

本論文では、分子動力学の他に流体シミュレーション、3次元ウェーハスタック構造超並列コンピュータの発熱シミュレーション、脳における視覚野神経細胞の学習時の活性化シミュレーションや自己組織化の超並列シミュレーションを行ないその高速性と有効性を明らかにした。更に、超並列シミュレーションからの膨大なシミュレーションデータの可視化を3次元コンピュータ・グラフィックスを用いた行なった。その結果、超並列シミュレーションデータのカラー可視化や3次元可視化手法の有効性を明かにした。

第 1 章

研究の背景

資源の乏しい我が国が今後工業立国ならびに経済国家として発展を続けるために、今まで以上に先端科学技術分野において効率のよい知的活動を支援するコンピュータシステムが要求されている。コンピュータを用いたシミュレーション技術は、自然科学・工学だけでなく経済学など広く社会的分野にも頻繁に用いられている。1950 年代以降のコンピュータの実用化と進歩によりシミュレーション技術は発展してきた。特に、最先端科学技術分野では、シミュレーションは不可欠な技術であり、大規模シミュレーションに必要とされるコンピュータ能力は膨大になってきている。また、物理分野においてシミュレーションは、理論物理、実験物理に分類されない第 3 の物理分野として注目されている。

最先端科学技術分野における大規模シミュレーションは、巨大なメモリ空間と膨大な計算時間を必用としており、従来のスーパ・コンピュータに代り、多数のプロセッサからなる超並列コンピュータ上でのシミュレーション技術開発が必要になってきている。また、1983 年に誕生したコンピュータグラフィクワークステーションの処理性能の著しい向上と操作性により、シミュレーションデータを種々の表現法で可視化する技術の開発が期待されている。

本研究では、計算機科学、画像処理、計算物理、材料科学やソフトウェア工学の研究者が超並列コンピュータ上でのシミュレーション技術の開発と膨大なデータの可視化（ビジュアル化）技術に関して総合的に研究を行ってきた。具体的には、物理化学分野における分子の振舞いを詳細にシミュレーションする分子動力学法、高分子化学分野におけるモンテカルロ法を用いた星状ポリマーの成長シミュレーション、機械工学における流体力学シミュレーション、大脳視覚野における学習時における神経細胞活性化シミュレーション、ニューラルネットワークの自己組織化シミュレーション、3 次元実装超並列コンピュータの発熱シミュレーション、海流のシミュレーション、照明シミュレーションや粒子を用いたダクト内の障害物の影響シミュレーションなどの広い分野で超並列シミュレーションを行なった。

これらのシミュレーショアルゴリズムは、超並列計算機 CM-5、nCUBE/2、Parsytec GC 上にインプリメントされ並列化性能およびシミュレーション結果の検討を行なった。シミュレーションデータは、カラー画像や 3 次元コンピュータグラフィックスにより可視化しインタラクティブなシミュレーションの遂行が可能である。

本報告書は、これらの研究成果をまとめたもので 8 章から成っている。第 1 章は、研究

の背景を説明し、本研究の目的について簡潔に述べる。

第2章では、物理化学分野における分子の振舞いを詳細に検討できる分子動力学法の大規模シミュレーションについて検討する。

第3章では、流体力学分野における問題を超並列計算機CM-5で実行する並列流体シミュレーションについて議論する。

第4章では、大脳視覚野における階層神経細胞ネットワークの学習における神経細胞活性化パターンのシミュレーションを行ない生理学分野での実験結果と比較検討する。

第5章では、自己組織化ニューラルネットワークの並列学習法について提案し、自己組織化マップのシミュレーションデータの可視化を行なう。自己組織化シミュレーションは256台のプロセッサからなる超並列計算機 nCUBE/2 に実装した。

第6章では、ウェーハスタック構造の3次元超並列コンピュータの実装時における発熱シミュレーションを行ない、発熱によるウェーハ上のプロセッサ故障回避を行なうアーキテクチャについて検討する。

第7章では、ラジィオシティ法を用いたコンピュータグラフィックスによる室内照明シミュレーションを行なった。シミュレーションは超並列計算機 Parsytec GC 上で、ラジオシティ法を並列化したものである。

第8章は、まとめである。

第 2 章

超並列計算機を用いた分子動力学法による大規模シミュレーションと可視化

2.1 はじめに

近年、計算機の高速化にともなって先端科学技術分野におけるコンピュータ・シミュレーションの果たす役割が大きくなっている。本論文では、材料設計への応用が期待される手法の一つである、分子動力学法 [1] によって得た計算結果の可視化について検討する。

分子動力学法は、分子一つ一つの運動を模擬することにより、安定な分子構造や、熱伝導度および拡散係数などの物質の動的な性質を調べる手法である。物理や化学をはじめとする材料科学の分野で、盛んに用いられている。逐次処理で計算できる分子の数は、計算時間およびメモリ使用量の限界から、数千個から非常に簡単な相互作用の場合で百万個程度である。それに対して、本来物質はアボガドロ数 6.02×10^{23} 程度の分子の集まりである。現実の物質に比較して、現在計算可能な物性は量的、質的に限られたものであり、複数プロセッサを用いて膨大な計算を行なうことができる並列計算機による高速化が期待されている。

扱う分子の量的な拡大を主な目的として、これまでにプロセッシングエレメント (PE) 数、CPU、結合方式の面で様々な並列計算機を用いた研究が行なわれている。Beazley や Lomdahl 等 [2][3] は 1024PE の超並列計算機上で、空間を分割して PE に割り当てる coarce-grained cell method (以下 CGC 法と称す) を用いて 1.3×10^8分子のシミュレーションを行なっている。このように、並列計算機を使用することによって一億個の分子のふるまいを模擬することも可能となってきている。これまで分子動力学法の計算結果は、分子の位置座標に球を表示して直観的に表現されていた。しかし、一億個程度の分子を扱うシミュレーションでは、もはや分子の位置を表示するだけでは全体像を把握することが困難である。そのため、よりわかりやすい可視化表現が必要である。

科学技術では可視化は大きな問題であり、種々の可視化システムが開発されている。高機能の可視化システムの一つとして AVS[4][5][6] がある。本稿では、3 次元空間での短距離相互作用を扱う分子動力学法シミュレーションにおける可視化手法を検討する。ここで

は低温、低密度のシミュレーションで気相と液相が共存する状態を、AVS を用いて可視化する。

本稿の構成は以下の通りである。第 2 節では分子動力学法に簡単に触れる。第 3 節では可視化の目的およびデータの可視化前処理について述べる。第 4 節では AVS を使用した可視化結果を示す。第 5 節はまとめである。

2.2 分子動力学法シミュレーション

2.2.1 分子動力学法の概要

分子動力学法とは、物質を構成する分子一つ一つの運動を模擬することによって、物質の状態を調べる計算手法である。多くの場合、分子の運動はニュートンの運動方程式に従うものとし、分子に働く力は分子の 2 体力の和によって与える。扱う分子の数を N とすると、分子の組みあわせは

$$ {}_NC_2 = \frac{N(N-1)}{2} \tag{2.1} $$

通りである。そのため、一般に計算時間は $O(N^2)$ で増加する。分子の間に働く力は、ポテンシャル関数で表した分子間相互作用として与えられることが多い。ここでは、相互作用としてレナード・ジョーンズ型ポテンシャル

$$ V(r) = 4\epsilon \left\{ \left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right\} \tag{2.2} $$

を用いる。式中の r は分子間距離、ϵ および σ は物質によって異なる定数である。このポテンシャルは、アルゴンやヘリウムなどの、電気的に中性な稀ガス元素の単原子分子の気体および液体に適用されてきた球対称ポテンシャルである。レナード・ジョーンズ型ポテンシャルでは、分子間距離が増大すると相互作用の大きさが急速に零に近づく。そのため、ある距離以上では相互作用を零として、力の計算時間を短縮する。このとき、相互作用の有効範囲を「力の切断距離」といい、このような相互作用を短距離相互作用と呼ぶ。

時刻 t での位置 $r(t)$、速度 $v(t)$ から微小時間 Δt 後の位置および速度を求めるために用いる数値積分は多くの場合、ベルレの方法の速度形式 [1] として知られる

$$ r_i(t+\Delta t) = r_i(t) + \Delta t v_i(t) + \frac{\Delta t^2}{2} \sum_{j \neq i} f(r_{ij}(t)) \tag{2.3} $$

$$ v_i(t+\Delta t) = v_i(t) + \frac{\Delta t}{2} \left(\sum_{j \neq i} f(r_{ij}(t)) + \sum_{j \neq i} f(r_{ij}(t+\Delta t)) \right) \tag{2.4} $$

を用いる。ここで $f(r_{ij}(t))$ は分子 i と分子 j の間に働く力、$r_{ij}(t)$ は分子間距離である。微小時間ごとに全ての分子について数値積分と相互作用の計算を行ない、これを 1 タイムステップと呼ぶ。計算する空間全体の境界条件は、通常周期境界条件が用いられる。

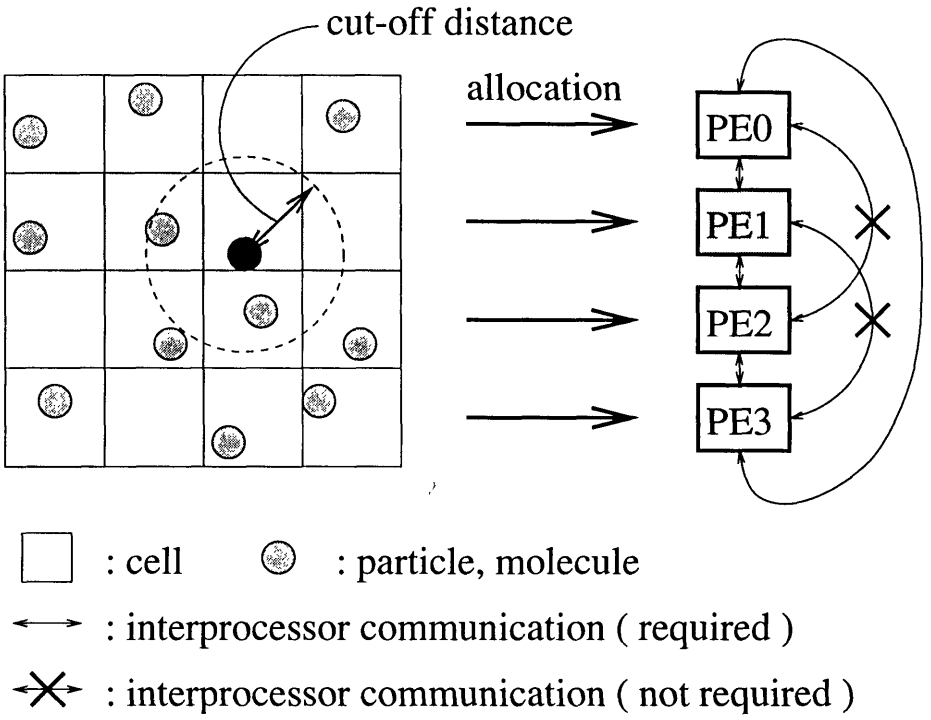


Figure 2.1: coarse-grained cell method(CGC 法) の概要

2.2.2 並列計算機を用いた分子動力学法シミュレーション

並列計算機上で分子動力学法シミュレーションを実行するため、種々の計算法が考案されてきた。coarse-grained cell method(CGC 法) は最も有効な並列化手法の一つである。CGC 法は、2 次元空間の場合は正方形、3 次元空間の場合は立方体にシミュレーションを行なう空間を分割 (セル) し、セル単位で PE に割り当てる方法である。2 次元のシミュレーションを例にして、CGC 法の概要を図 2.1に示す。力の切断距離よりもセルが大きいならば、各セル中の分子は近傍セル (2 次元空間では 8 近傍、3 次元空間では 26 近傍) 中の分子との相互作用のみを考慮すればよい。

2.2.3 シミュレーション条件

ここでは、レナード・ジョーンズポテンシャルの定数にアルゴンの数値を用いた．分子の直径を単位として、力の切断距離は通常 2.5 ～ 3.5 が用いられるが、ここでは 2.5 を使用した。数値積分の微小時間として、無次元化時間$\Delta t = 0.064$ を用いた。シミュレーションの間は分子の数 N，体積 V，エネルギー Eを一定に保ち，温度は 50 タイムステップごとに T_{ref}に補正する．密度は$\rho = 0.256$ とし、温度は高温の場合 (無次元化温度 $T_{ref} = 2.53$) および低温の場合 (無次元化温度 $T_{ref} = 0.722$) である。

2.3 シミュレーションデータの可視化

2.3.1 AVS の概要

AVS (Advanced Visual Systems) は科学技術のあらゆる分野に適用可能な可視化ツールである。処理単位をモジュール化し、ビジュアルプログラミングによって結合することによって、容易に可視化を行なうことができる。等値面や断面図の作成および画像処理などを行なう豊富なモジュールを持ち、最も高機能な可視化ツールの一つである。適用分野は多岐にわたり、医療画像や構造体力学、材料科学などに用いられている。AVS の特徴的な機能のひとつは、3 次元物体を詳細に描画するボリューム・レンダリングである。ここではボリューム・レンダリングを利用して可視化を行なう。

2.3.2 低密度における分子位置

512 分子のシミュレーション結果を、分子位置の表示によって示す。高温、低密度の結果を図 1、低温、低密度の結果を図 2 に示す。この程度の数のシミュレーションでは、分子の位置を表示することによって、低温の場合に分子が集中して液体となる直観的な様子がわかる。しかし、数十万、数百万個の分子を扱う場合には、分子の位置を表示するだけでは分子の集中する様子を把握することができない。そのため、分子が集中することを数値化して可視化することが必要である。分子の集中を示す量の一つは、局所的な密度である。

非平衡状態を調べるとき、局所的な物理量の変化は興味深い。局所的な物理量として、密度だけでなく温度も考えられる。気体分子運動論により、理想気体の N 個の分子を含む物理系において、分子の速度と温度の関係は

$$\frac{1}{2}m\sum_{i=1}^{N}\mathbf{v}_i^2 = \frac{2}{3}Nk_BT \tag{2.5}$$

である (ここで m は分子の質量、$\mathbf{v}$は分子の速度ベクトル、k_Bはボルツマン定数、Tは温度である)。(2.5) 式から、逆に分子一つの温度や、空間の局所的な温度を求めることができる。

ここでは

- 局所的な密度
- 局所的な温度

について可視化を行なうので、これらの量を可視化前に求める必要がある。空間を規則的に小さな立方体に分割し、各小立方体中に位置する分子の個数から局所的な密度を求める。ここでは小立方体の大きさを力の切断距離の半分とした。さらに、小立方体中の分子の速度から、局所的な温度を求めた。この手法は、2.2.2節で述べた並列化手法と容易に組みあわせることができる。

2.3.3 局所密度と局所温度の可視化

$T_{ref} = 0.722,\ \rho = 0.256$ の条件で 3375 分子のシミュレーションを行ない、可視化を行なった。その結果を図 2.6 および図 2.7 に示す。図 2.6 は局所密度と局所温度を 3 次元のボリューム上に色づけして表示し、同時に座標軸方向の断面を表示したものである。図 2.7 は同じデータを 3 次元のボリューム上に透明に色づけして表示し、斜め方向の断面を作成して同時に表示したものである。これらの断面は、任意の位置に対して作成することができる。分子の数が小さいために、表示にあたっての解像度は小さい。逆に、表示の解像度を上げることによって、大規模シミュレーションの結果を扱うことが充分可能である。

図 2.6 によると、局所密度の図からは高密度の部分を中心に分子が集まっていることがわかる。局所温度の図では、ほとんどの空間で局所温度は低くなっているが、まれに温度が高くなる部分がある。この部分は局所密度の図で密度が低い部分に相当することから、全体として低温であるが、まれに速度の大きい分子が存在することにより、高温部分ができることを示す。

図 2.7 によると、局所密度の図からは、分子が集中する中核となる部分 (図中の赤色の部分) がいくつかあり、その周辺に分子が集まってきていることがわかる。局所温度の図からは、局所的に高温となる部分はさほど多くないことがわかる。

図 2.6 は断面を移動させながら、各部分の物理状態を調べることに適し、図 2.7 は全体の分布を把握して注目すべき断面を作成することに適している。さらに、局所密度の図と局所温度の図を画像処理によって合成すれば、液相部分を決定し、液相部分の境界面を表示させることも可能である。

2.4 まとめ

本稿では、分子動力学法の大規模シミュレーション結果の可視化について検討した。ボリューム・レンダリングを利用することにより、全体の構造を把握しながら局所的な構造を調べることが可能であった。さらに、種々の物理量を表示して比較することによって、計算結果を多角的に検討することが可能であることを示した。分子動力学法シミュレーションの大規模化にともなって、ボリューム・レンダリングを用いた可視化の重要性はますます大きくなると考えられる。

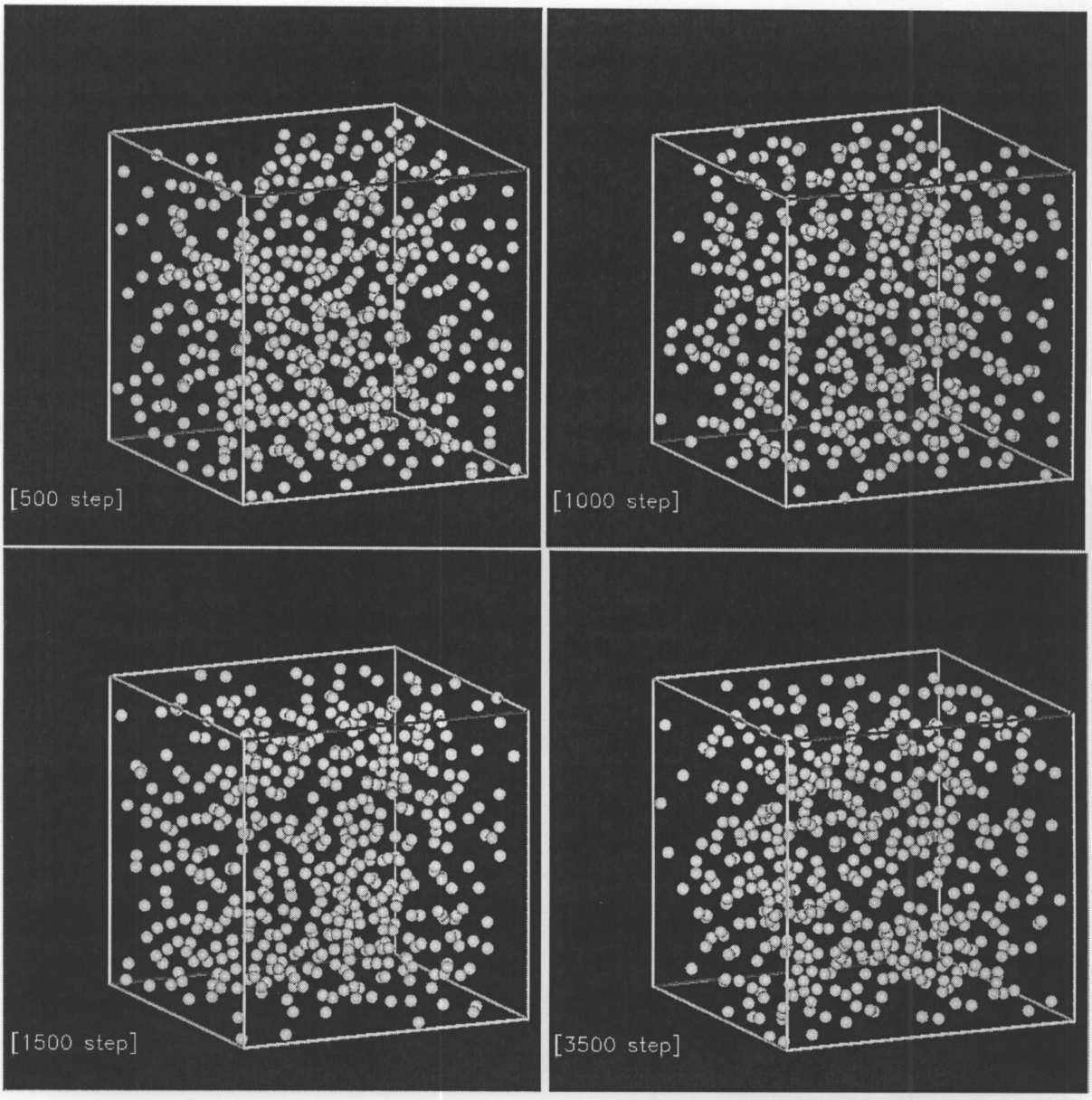


Figure 2.4: 高温、低密度のシミュレーション結果
($N = 512,\ T_{ref} = 2.53,\ \rho = 0.256,\ r_c = 2.5$)

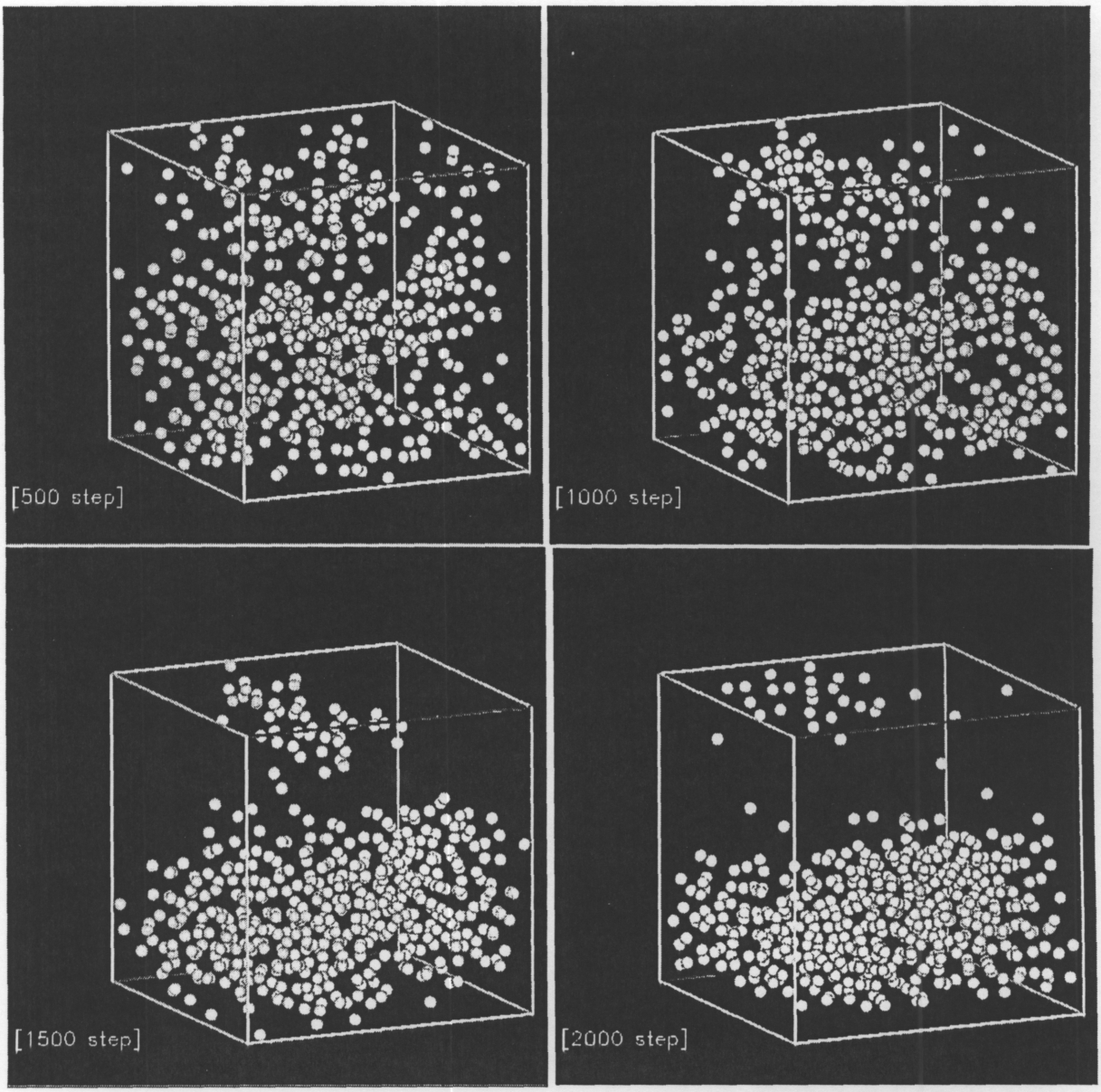
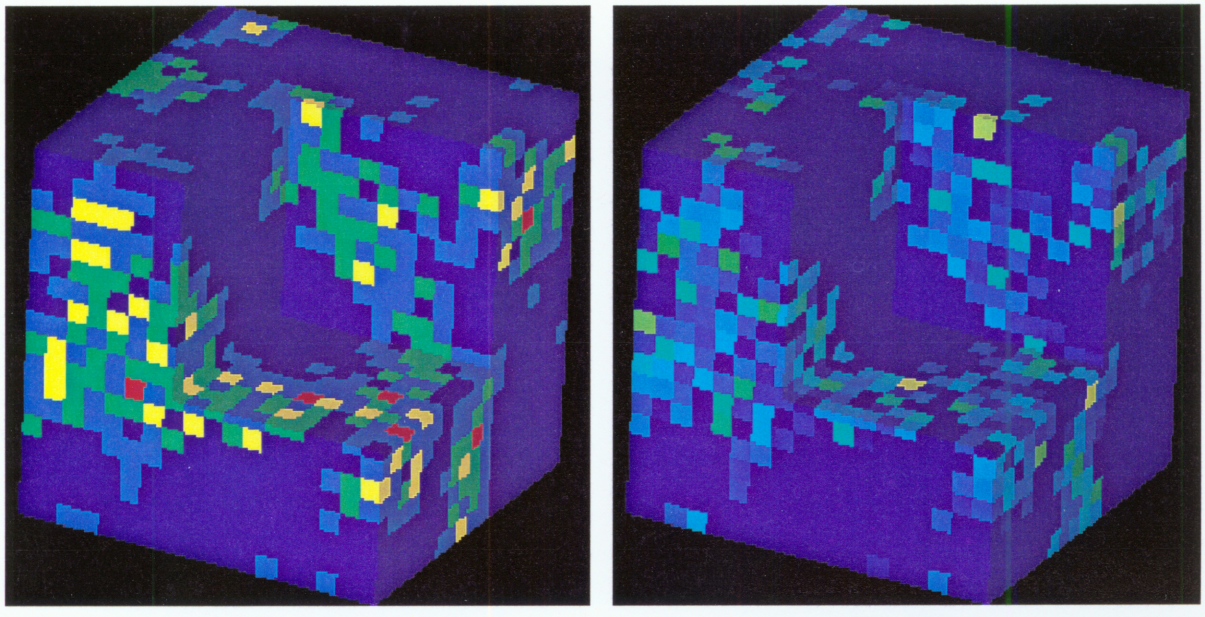
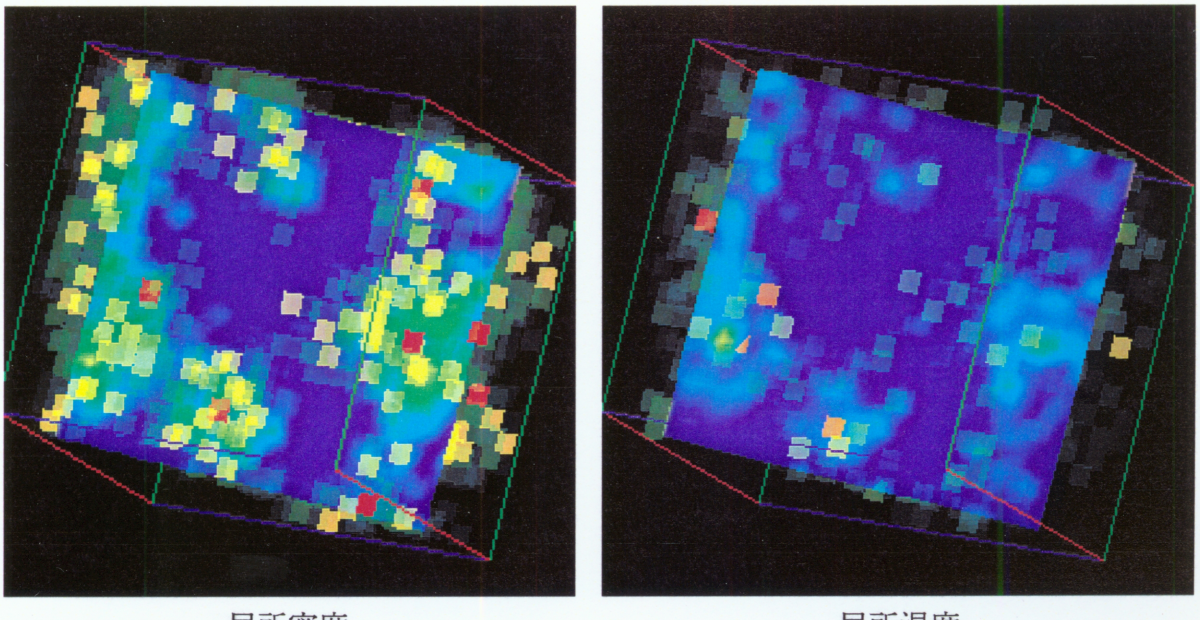


Figure 2.5: 低温、低密度のシミュレーション結果
($N = 512,\ T_{ref} = 2.53,\ \rho = 0.256,\ r_c = 2.5$)



局所密度　　局所温度

Figure 2.6: 座標軸に垂直な断面のコンター図による表示



局所密度　　局所温度

Figure 2.7: ボリューム・レンダリングを用いた表示

第 3 章

超並列計算機を用いた流体力学シミュレーションと可視化

3.1 はじめに

流体力学の数値シミュレーションでは本質的に並列処理が可能なことから、マルチプロセッサ・システムによる高速処理に期待が寄せられている。本章では代表的な超並列計算機である Thinking Machines 社の CM-5 [7] を用いて、流体の数値シミュレーションの典型的な検証問題である 2 次元正方流れを MAC 法 [8] により解き、その実行速度について評価を行なう。

最初に流体力学の基礎方程式の差分法に基づく離散化を行ない、この離散化方程式に基づく流体の数値シミュレーションを CM-5 に実装し、実行速度の評価を行なう。離散化の方式は、差分法に基づく MAC 法を用いる。差分法 [9] は対象領域を格子状に区切り、格子間で離散的に積分し、各点の物理量を時間積分する数値解法である。

使用する超並列計算機 CM-5 は、SIMD 型、MIMD 型など複数の並列処理形態を統合したハイブリッド型の超並列計算機である。プログラム言語は FORTRAN90 の規格に基づく CM-FORTRAN を使用し、Data Parallel Model [10] に従い、SIMD 型の並列処理を行なう。求めた差分方程式を CM-5 上に実装し、実行速度について検討する。データの PE への割り当て方式が実行速度に及ぼす影響について議論する。

また、流体力学シミュレーションで求めた差分方程式を、圧力分布や流速などの可視化について議論する。

3.2 差分法

3.2.1 基礎方程式

流体力学の数値シミュレーションは、次の基礎方程式を離散化することにより計算される。流体力学の基礎的な方程式は、連続の式と運動方程式 (ナビエー ストークス方程式) で

ある。連続の式は、流体が連続していることから、次の様に書ける。

$$\frac{\partial \rho}{\partial t} + \mathrm{div}(\rho \boldsymbol{v}) = 0 \tag{3.1}$$

運動方程式(ナビエー ストークス方程式)は、微少空間内の運動方程式であり、次式で与えられる。

$$\frac{D\boldsymbol{v}}{Dt} = -\frac{1}{\rho}\mathrm{grad}\ p + \frac{1}{3}\nu \mathrm{grad} \cdot \mathrm{div}\ \boldsymbol{v} + \nu \nabla^2 \boldsymbol{v} \tag{3.2}$$

簡単のため、非圧縮粘性流体を対象とすると、非圧縮の条件より

$$\rho = const. \tag{3.3}$$

であるから、連続の式は、

$$\mathrm{div}(\boldsymbol{v}) = 0 \tag{3.4}$$

と書ける。これを式(3.2)に代入すると、

$$\frac{D\boldsymbol{v}}{Dt} = -\frac{1}{\rho}\mathrm{grad}\ p + \nu \nabla^2 \boldsymbol{v} \tag{3.5}$$

である。レイノルズ数を用いて無次元化すると、

$$\frac{D\boldsymbol{v}}{Dt} = -\mathrm{grad}\ p + \frac{1}{Re}\nabla^2 \boldsymbol{v} \tag{3.6}$$

が得られる。

これら連続の式(3.4)及び運動方程式(3.6)をスカラー形式で書けば、それぞれ

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{3.7}$$

$$\frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{3.8}$$

であるが、ここで

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \tag{3.9}$$

であるから、これを式(3.8)に代入して定常非圧縮粘性流体の運動方程式

$$\frac{\partial u}{\partial t} = -\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right) - \frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{3.10}$$

を得る。v, wについても同様に求められる。

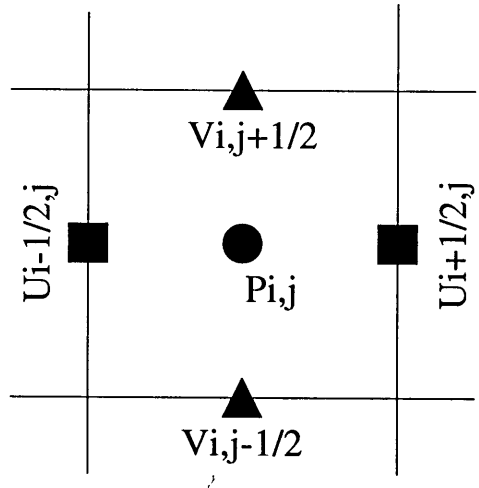


Figure 3.1: 食い違い格子

3.2.2 離散化方程式

次にこれを離散化することを考える。図3.1の様な3次元食い違い格子を考える。式(3.7)を中心差分すると、

$$D_{i,j,k} = \frac{u_{i+1,j,k} - u_{i,j,k}}{\Delta x} + \frac{v_{i,j+1,k} - v_{i,j,k}}{\Delta y} + \frac{w_{i,j,k+1} - w_{i,j,k}}{\Delta z} = 0 \tag{3.11}$$

また式(3.10)を$u_{i,j,k}$について時間微分を前進差分で離散化すると、

$$u^{n+1}_{i,j,k} = u^{n}_{i,j,k} + \Delta t(F_{i,j,k} - \frac{p_{i,j,k} - p_{i-1,j,k}}{\Delta x}) \tag{3.12}$$

$$v^{n+1}_{i,j,k} = v^{n}_{i,j,k} + \Delta t(G_{i,j,k} - \frac{p_{i,j,k} - p_{i,j-1,k}}{\Delta y}) \tag{3.13}$$

$$w^{n+1}_{i,j,k} = w^{n}_{i,j,k} + \Delta t(H_{i,j,k} - \frac{p_{i,j,k} - p_{i,j,k-1}}{\Delta z}) \tag{3.14}$$

$F_{i,j,k}$は

$$\begin{aligned} F_{i,j,k} &= -(u_{i,j,k}\frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\Delta x} + v'_{i,j,k}\frac{u_{i,j+1,k} - u_{i,j-1,k}}{2\Delta y} + w'_{i,j,k}\frac{u_{i,j,k+1} - u_{i,j,k-1}}{2\Delta z}) \\ &+ \frac{1}{Re}(\frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{\Delta x^2} + \frac{u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{\Delta y^2} \\ &+ \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{\Delta z^2}) \end{aligned} \tag{3.15}$$

$$v'_{i,j,k} = \frac{v_{i,j,k} + v_{i,j+1,k} + v_{i-1,j,k} + v_{i-1,j+1,k}}{4} \tag{3.16}$$

$$w'_{i,j,k} = \frac{w_{i,j,k} + w_{i,j,k+1} + w_{i-1,j,k} + w_{i-1,j,k+1}}{4} \tag{3.17}$$

(3.18)

である。$G_{i,j,k}, H_{i,j,k}$も同様である。

$t+\Delta t$での値を正規化するため、ここで式(3.12),(3.15)を式(3.11)に代入すると、

$$\begin{aligned} D_{i,j,k} &= \left(\frac{u_{i+1,j,k}-u_{i,j,k}}{\Delta x}+\frac{v_{i,j+1,k}-v_{i,j,k}}{\Delta y}+\frac{w_{i,j,k+1}-w_{i,j,k}}{\Delta z}\right) \\ &+ \Delta t\left(\frac{F_{i+1,j,k}-F_{i,j,k}}{\Delta x}+\frac{G_{i,j+1,k}-G_{i,j,k}}{\Delta y}+\frac{H_{i,j,k+1}-H_{i,j,k}}{\Delta z}\right) \\ &- \Delta t\left(\frac{p_{i+1,j,k}-2p_{i,j,k}+i_{i-1,j,k}}{\Delta x^2}+\frac{p_{i,j+1,k}-2p_{i,j,k}+i_{i,j-1,k}}{\Delta y^2}+\frac{p_{i,j,k+1}-2p_{i,j,k}+i_{i,j,k-1}}{\Delta z^2}\right) \\ &= 0 \end{aligned} \tag{3.19}$$

即ち、

$$\begin{aligned} &\frac{p_{i+1,j,k}-2p_{i,j,k}+i_{i-1,j,k}}{\Delta x^2}+\frac{p_{i,j+1,k}-2p_{i,j,k}+i_{i,j-1,k}}{\Delta y^2}+\frac{p_{i,j,k+1}-2p_{i,j,k}+i_{i,j,k-1}}{\Delta z^2} \\ &= \frac{1}{\Delta t}\left(\frac{u_{i+1,j,k}-u_{i,j,k}}{\Delta x}+\frac{v_{i,j+1,k}-v_{i,j,k}}{\Delta y}+\frac{w_{i,j,k+1}-w_{i,j,k}}{\Delta z}\right) \\ &\quad +\left(\frac{F_{i+1,j,k}-F_{i,j,k}}{\Delta x}+\frac{G_{i,j+1,k}-G_{i,j,k}}{\Delta y}+\frac{H_{i,j,k+1}-H_{i,j,k}}{\Delta z}\right) \end{aligned} \tag{3.20}$$

を得る。

3.3 並列計算機による流体力学シミュレーション

本節では、差分法による流体力学シミュレーションの典型的な検証問題である2次元正方流れを、代表的な超並列計算機であるThinking Machines社のCM-5[7]を用いて、MAC法[8]により解き、各処理の実行速度について評価を行なう。CM-5はSIMD型、MIMD型など複数の並列処理形態を統合したハイブリッド型計算機である。プログラム言語はFORTRAN90を使用し、Data Parallel Model[10]に従い、SIMD型の並列処理を行なう。差分法を超並列計算機上で実行した場合の、演算速度に影響を及ぼす要因を明らかにする。

計算速度の評価のため、MAC法による2次元正方流れのシミュレーションプログラムを修正した試験プログラムを作成した。差分法を用いた流体力学シミュレーションの手順は以下のようになる。

1. 式(3.15)の$F_{i,j,k}$を計算する。
2. $F_{i,j,k}$を用いて式(3.12)を計算し、時間$t+\Delta t$の状態を得る。
3. u,v,w,pについて、誤差平均をとる。
4. 式(3.20)をSOR法により解き、系を正規化する。

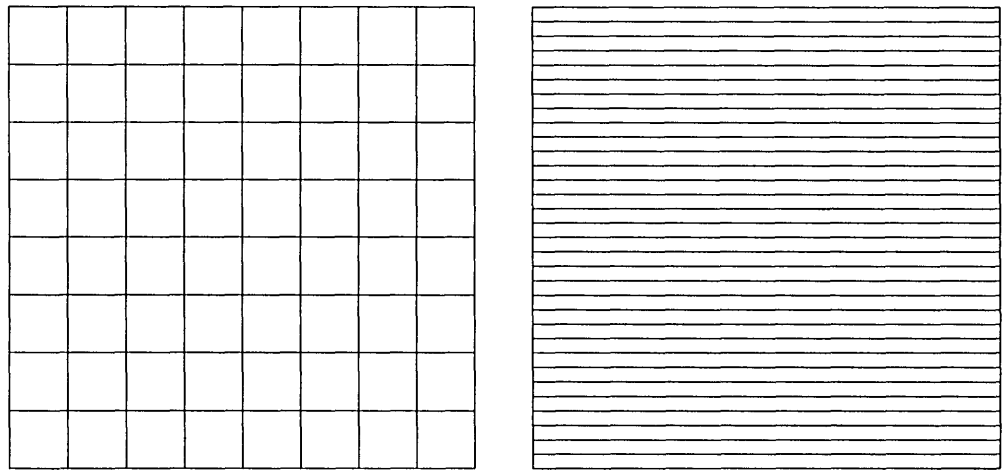


Figure 3.2: 配列の分割方法

これを繰り返し、誤差が一定以下あるいは所定の回数を繰り返した後、結果を出力する。誤差は

$$\left(\left|F-\frac{\partial p}{\partial x}\right|<\epsilon\right)\ and\ \left(\left|G-\frac{\partial p}{\partial y}\right|<\epsilon\right)\ and\ \left(\left|H-\frac{\partial p}{\partial z}\right|<\epsilon\right) \tag{3.21}$$

で求める。ここでは他の計算機との比較のため、ループを規定の回数実行することで終了する。

並列処理を行なうためには、配列変数を部分配列に分割し、この部分配列を各 Processing Node (PN) に割り当てる必要がある。配列の分割方法には複数の方法があるが、本試験プログラムでは、2 次元配列を正方形の部分配列に分割する方法 (cm5n)、及び短冊状の部分配列に分割する方法 (cm5s) の、2 種類の分割方法で計算を行なった。分割の様子を図 3.2 に示す。短冊状の分割では、インプリメントが単純で、各短冊の幅を変化させることにより、動的な負荷分散が行ないやすい。また、短冊状のデータ配置は、各短冊を直線状に並べたものと考えることができるので、CM-5 の結合網 Fat Tree に適合しやすいという利点がある。一方、格子状の分割は、PN 間の境界要素数が短冊状の分割に比べ少ないため、プロセッサ間通信の通信量を減らすことができる。なぜならば、$m \times m$ の要素を N個の PN にマッピングすることを考えると、境界線の長さは、短冊状の分割だと mNであるのに対し、格子状の分割では $2m\sqrt{N}$となるからである。分割方法は、FORTRAN コンパイラに対する並列化指示行で指示する。

ここで配列変数の大きさを次のように定義する。

大きさ (size) 配列の 1 次元当りの要素数
要素数 配列に含まれる要素の数 2 次元配列の場合 $size^2$

また、計算で消費する時間を次のように定義する。

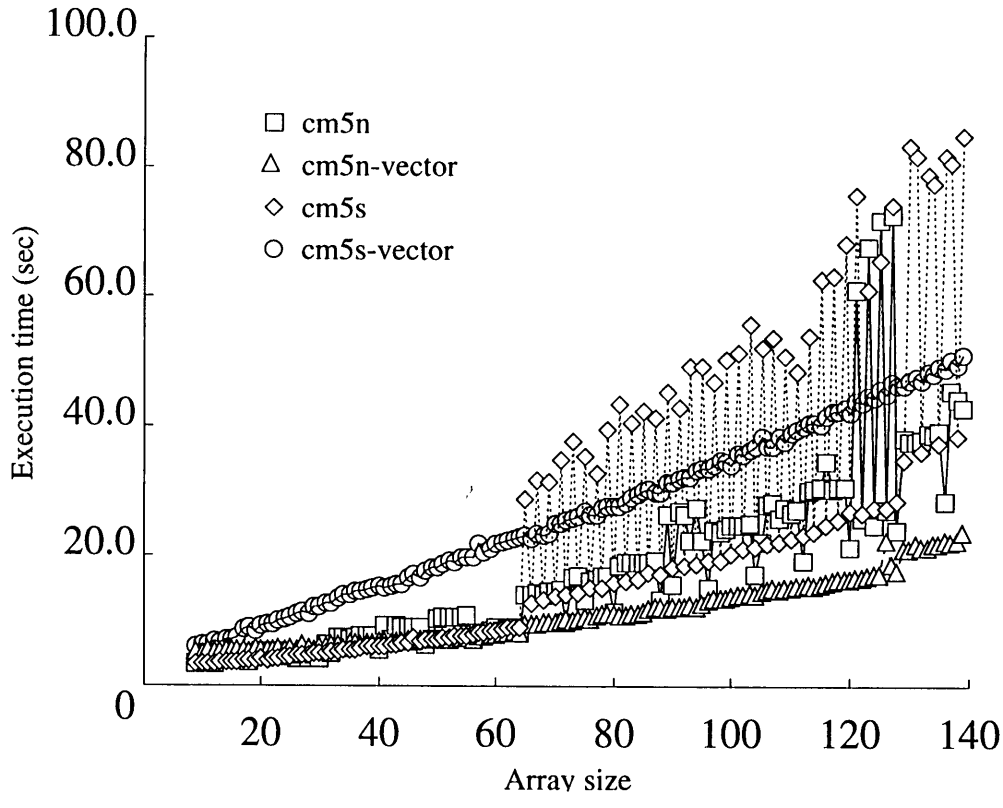


Figure 3.3: 実行時間

ノード cpu 時間 プログラムが PN で演算に消費した時間
通信時間 プログラムがノード間通信に消費した時間
計算時間 プログラムが処理に消費した時間 (ノード cpu 時間+通信時間)

測定に用いた超並列計算機は PN 数が 64 の CM-5 である。CM-5 上での実行時間内訳は、CM-5 のデバッガ prism [11] を用いて測定した。

配列の大きさに対する実行時間のグラフを図 3.3に示す。図中、cm5n は格子状の分割、cm5s は短冊状の分割を用いた場合の実行時間である。また cm5nv, cm5sv は、ベクトルユニットを使用した場合の、格子状、短冊状の配列分割法による実行時間である。格子状の分割と短冊状の分割を比較すると、ベクトルユニットを使用した場合も使用しない場合も、それぞれ格子状の分割の方が高速に実行できた。

次に、格子状の分割による、各処理の処理時間について検討する。図 3.4 に配列の大きさが 64 の時の、ノード cpu 時間を示す。上段左のグラフの Node cpu (user) が ノード cpu 時間、Comm(NEWS) が隣接ノード間通信時間である。上段右のグラフは、各サブルーチンの処理時間である。全体の処理時間の殆どを、SOR 法によって圧力に関するポアソン方程式 (3.20) を解くサブルーチン poisn で消費していることが分る。下段は、サブルーチン poisn における各文の実行時間を示す。740~800 行は全配列要素に対する計算を行なって

いる。Data Parallel Model によるプログラムでは、配列変数をスカラー変数のように一括して扱うことができる。境界条件設定 (730 行) は、全配列要素に対する演算 (740~800 行) よりも処理する配列要素数が大幅に少ないが、ノード cpu 時間は同じ時間消費する。また差分式の計算 (720 行、上下左右の配列要素との演算) では、配列の全要素を処理するが、演算が複雑であるため、ノード cpu 時間が長く必要となっている。

図 3.5に、同サブルーチンの隣接ノード間通信時間を示す。通信時間は差分式の計算 (720 行:式 (3.20)) で全て消費されている。この時、通信時間はノード cpu 時間の約 3 倍であるので、差分式における隣接ノード間通信は、並列処理の処理効率に大きく関係することが分る。ノード cpu 時間に対する通信時間の割合は、配列の大きさが増大するに従い減少する。

3.4 流体力学シミュレーションの可視化

図 3.6に正方流れの圧力分布を示す。圧力が低い領域は青色、圧力が高い領域は赤色で示している。図の右上の一部分に、非常に高い圧力の領域があり、このため、右上の領域以外の図全体では、殆ど差が分らなくなっている。図 3.8は、等圧線による圧力分布図である。直観的には図 3.6の方が分りやすいが、この図では、圧力が高い領域がどのようになっているかが、より正確に分る。

図 3.7は、正方流れの流速分布である。流速が遅い領域は青色、流速が速い領域は赤色で示している。全体でいびつなドーナツ状の流れが生じているのが分る。この場合も図面上端の領域の流れが非常に速いため、全体の流速分布が分りにくくなっている。一方、各点での流れのベクトルを示した図 3.9では、多少細かく入りまじって混雑しているが、全体の様子を把握しやすく、また図の四隅に二次渦が形成されている様子まで良く分る。

3.5 まとめ

本章では、超並列計算機を用いた流体力学シミュレーションとその可視化について議論した。最初に流体力学の基礎方程式の差分法に基づく離散化を行ない、この離散化方程式を用いて流体の数値シミュレーションの典型的な検証問題である 2 次元正方流れを、超並列計算機の 1 つである CM-5 に実装し、その実行速度について評価を行なった。その結果、並列計算機へのデータの割り当て方式は、短冊状に分割するよりも格子状に分割して各 PE に割り当てる方式がより効率的に実行できることが分った。

また、求めた正方流れの圧力分布や流速分布を、色分けや、等圧線及び流線によって可視化した。

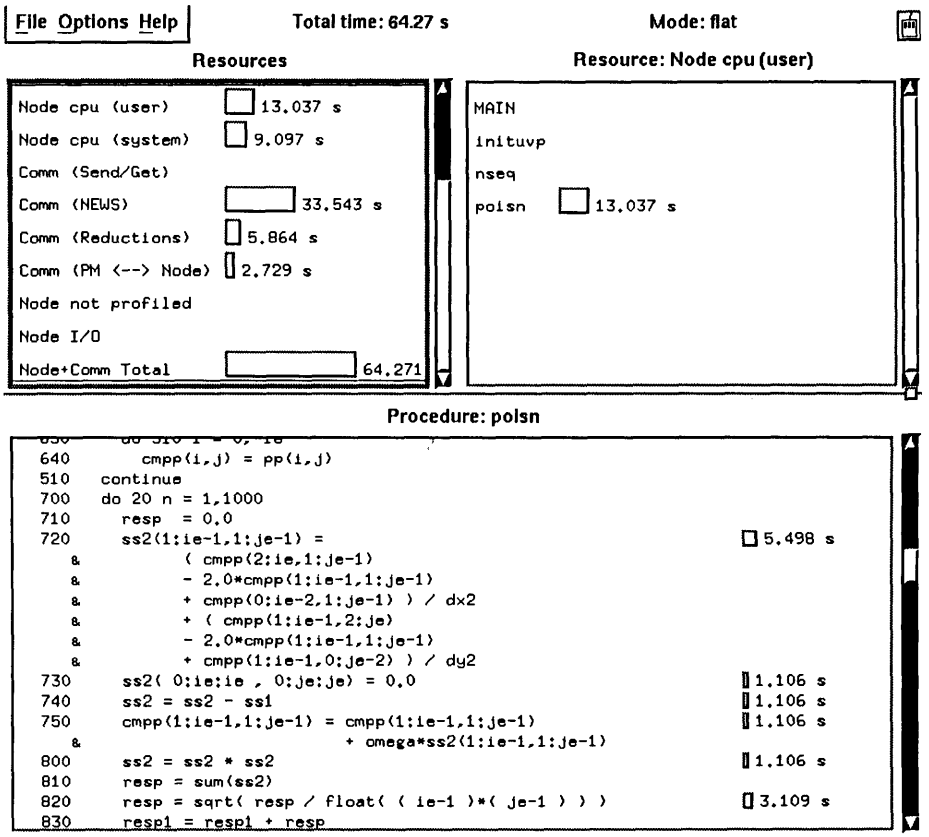


Figure 3.4: ノードの演算時間

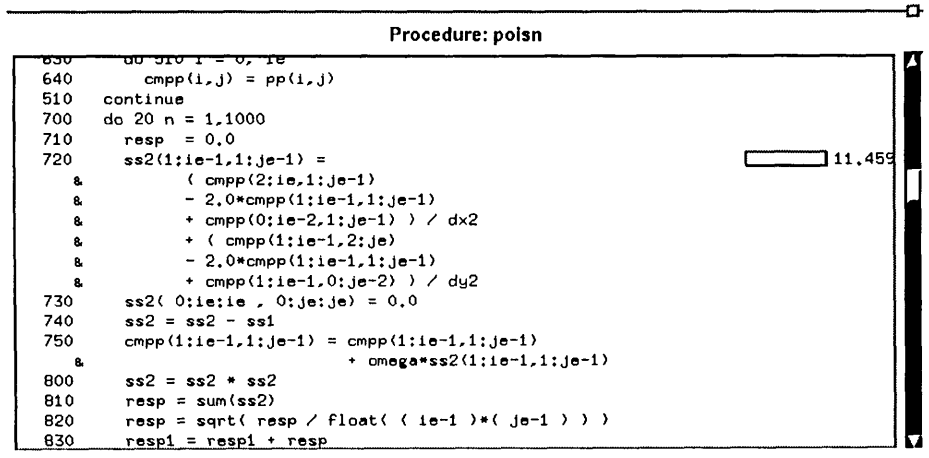


Figure 3.5: 隣接ノード間通信時間

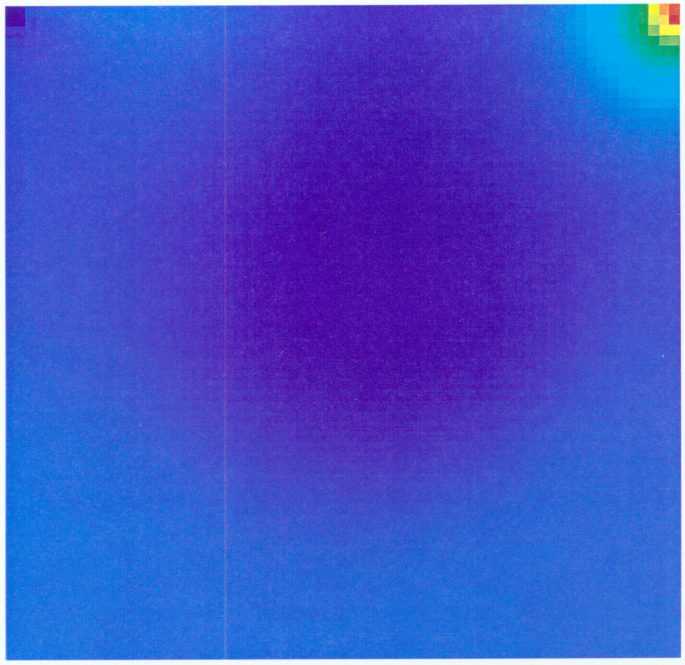


Figure 3.6: 正方流れの圧力分布 (1)

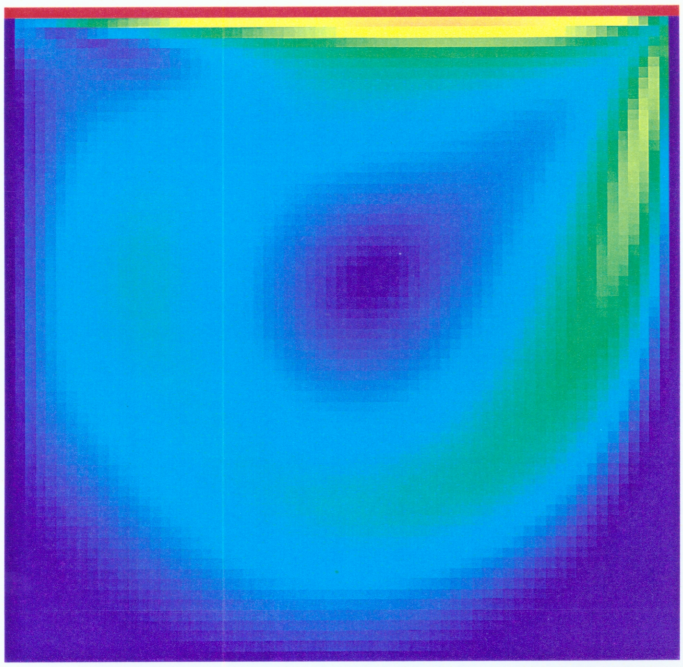


Figure 3.7: 正方流れの流速分布

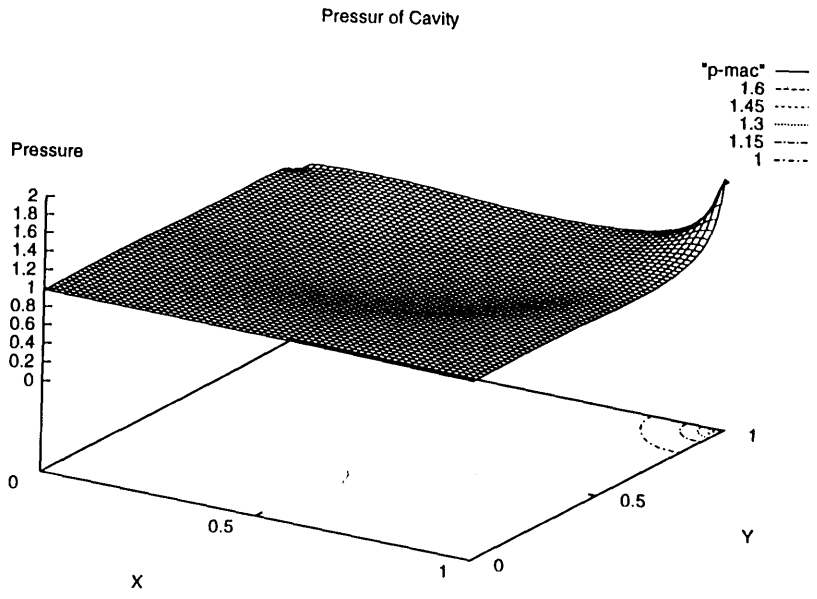


Figure 3.8: 正方流れの圧力分布 (2)

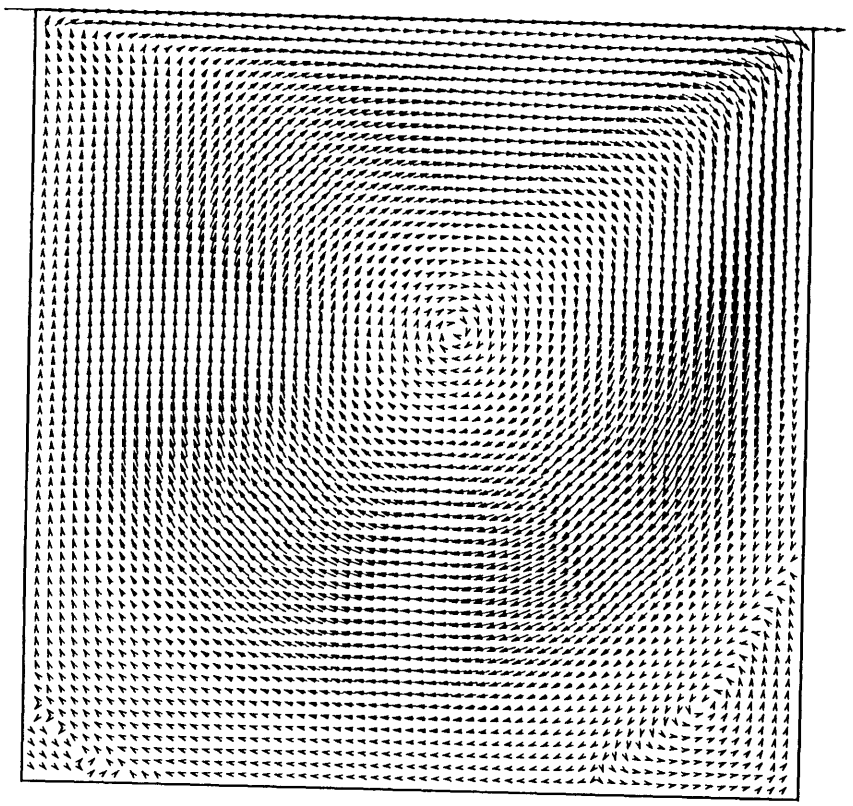


Figure 3.9: 正方流れの流線

第 4 章

大脳視覚野における学習シミュレーションと神経細胞活性パターンの可視化

4.1 はじめに

大脳の視覚野における刺激選択性細胞としては、第 1 次視覚野において、ある特定の傾きを持つスリット光が呈示されたときに限り強く活性化する方位選択性細胞が古くから知られている [12]。近年、Tanaka[13] らはマカカ属のサルの下部側頭葉皮質（IT 野）において、特定の単純図形やテクスチャ、コントラストの変化など、中程度に複雑な図形に対して選択的に活性化する細胞を発見した。さらに、Fujita[14] らは類似した図形特徴に選択的に活性化する細胞が皮質表面に対し垂直に分布し、第 1 次視覚野と同様のコラム構造を形成していることを報告している。しかしながら、一つのコラム内における細胞間の結合、コラム細胞間の結合や学習方式がどうなっているかはまだ明らかになっていない。

本研究では、生理学的知見に基づいて、V1 野から下部側頭葉皮質 IT 野の階層型神経結合モデル [15] における神経細胞の活性パターンを詳細に検討する。最初に、Kohonen 自己組織化モデル [16] に基づいた神経細胞結合モデルについて述べ、学習シミュレーションにより V1 野の方位選択性細胞における生理学的知見との比較検討を行なう。さらに、階層型神経細胞結合モデルのシミュレーションにより、中程度に複雑な視覚パターンに対する IT 野のコラム構造細胞の活性反応について詳しく検討する。また、階層型神経細胞結合モデルが、人間の視覚現象の一つであるブルドン効果 [17] を再現可能であることを示す。

4.2 大脳視覚野

4.2.1 V1 野

視覚系は機能的側面から約 40 の小領野に分けられることが明らかにされている。外界の視覚情報は、まず第 1 次視覚野（V1 野）へと送られる。V1 野は、送られてきた視覚情報を統合し、処理された情報をその属性によって V1 野以降の各領野（V2 野、V3 野、V4

野、MT 野）へ分配することである。

V1 野には、特定の傾きを持つスリット光に対して選択的に活性化する細胞が多数存在する。これらの細胞は方位選択性細胞と呼ばれる。方位選択性細胞は V1 野の皮質上で無秩序に分布してはおらず、類似した傾きを持つスリットに対して選択的に活性化する細胞が、皮質表面に対して垂直方向に分布している。皮質に対して垂直な柱状を成すこのような構造はコラム構造と呼ばれ、方位選択性細胞の場合は方位選択性コラムと呼ばれている。また、隣合う方位選択性コラムの間には、皮質表面に沿って細胞が最も強く活性化するスリットの傾きが徐々に変化するという関係がある。これらのことから、皮質内のある点を中心として時計周り（反時計周り）に個々の細胞の最適方位が連続的に変化するモデルが報告されている [18]。

4.2.2 IT 野

IT 野を破壊されたサルは、視覚対象を知覚あるいは記憶するような行動課題が著しく困難となる。このことから、IT 野は図形の識別や、物体の認知に関わっていると考えられる。IT 野の細胞が選択的に活性化する図形としては図 4.1に示すような逆 T 字型、ドットで出来た傾いた線、星型、赤のような飽和色、はだ色のような不飽和色、2 つの色が組み合わさった色、横縞、ドットパターン、濃淡の勾配といったテクスチャ、及びこれらの組み合わせである [14]。これらの図形特徴は、一つの物体を特定できるほど複雑ではなく、V1 野の細胞の活性反応を引き起こすようなスリットのように単純でもない。また、IT 野は V1 野から間接的な神経投射を受けている。以上のことから、IT 野の細胞は V1 野が抽出するスリットの情報を統合した、単純な幾何学図形に選択的に活性化していると考えられる。

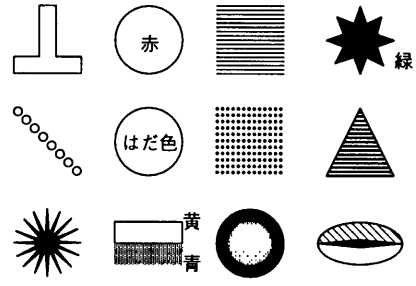


Figure 4.1: IT 野の細胞が選択的に活性化する図形の例 [14]

4.3 階層型神経結合モデル

4.3.1 ネットワーク構造

大脳視覚野では、機能的に独立したV1野やIT野と言った領野が階層的に結合している。物体認知はV1野に始まり、V2野、V4野を経てIT野でほぼ完了すると考えられている[19]。これらの生理学的知見から、IT野は階層構造によってV1野の抽出したスリットの情報を統合すると仮定する。以下では、V1野-IT野階層型神経結合モデルについて詳細に述べる。

まず網膜から第1次視覚野に相当する神経結合モデルとして、入力層と競合層からなる図4.2(a)に示すネットワークを用いる。これをV1野の神経結合モデルとしてV1野モデルと呼ぶ。入力層には$n \times n$のノードが、競合層には$m \times m$のノードがそれぞれ2次元格子状に配置されている。入力層は網膜に相当し、競合層のノードはV1野における神経細胞に相当する。入力層の各ノードは0.0〜1.0までの値をとり、$n \times n$の視覚イメージを表現できる。視覚イメージは、入力層と完全結合した$m \times m$のノードから構成される競合層の各ノードに結合重みw_{ij}を介して伝達される。V1野モデルの自己組織化学習は、入力層に任意の位置、傾きを持つスリットパターンをランダムに呈示することによって行なう。

V1野モデルの学習が終了した時点で、図4.2(b)に示すように競合層を階層的に追加した階層構造の神経結合モデル（V1野-IT野モデル）を構成する。図4.2(c)に示すV1野-IT野モデルにおいて、入力層は網膜に相当し、第1競合層はV1野に、第2競合層はIT野に相当する。ここで、第1競合層の結合重みベクトルはV1野モデルにおいてスリットパターンを学習した時の結合重みベクトルを用いる。第2競合層の学習時には第1競合層は新たな学習をせず、w_{ij}の更新は行なわれない。

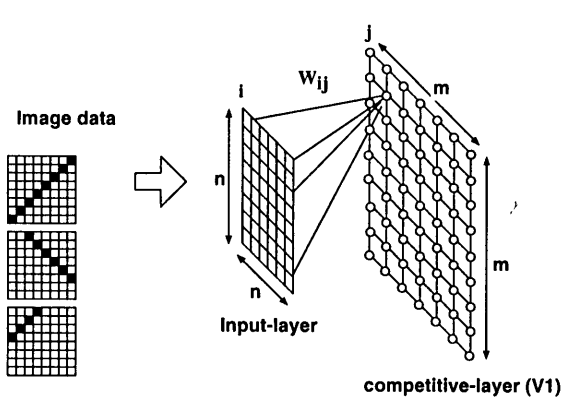
第1競合層が第2競合層に与える入力信号は、図4.2(c)に示すように、第1競合層が持つ入力層に図形パターンが呈示された時の、第1競合層の各ノードjの活性値（式（4.4）における$f_j(u)$の値）となる。学習を行なうのは第2競合層だけである。第2競合層は、第1競合層の活性パターンを入力ベクトルとしてw_{jk}を更新し、学習を行なう。

4.3.2 学習アルゴリズム

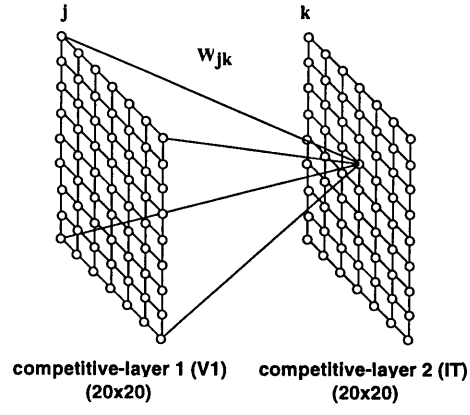
階層構造神経結合モデルの学習にはKohonen自己組織化モデル[16]における学習アルゴリズムを用いる。入力層から競合層に入力パターンが与えられたとき、入力ベクトルEと最も良く一致する結合重みベクトルをW_cとし、W_cを持つ競合層のノードcを勝者ノードとする。すなわち、勝者ノードcおよびW_cは次式によって定義される。

$$\| E - W_c \| = \min_j \{ \| E - W_j \| \} \tag{4.1}$$

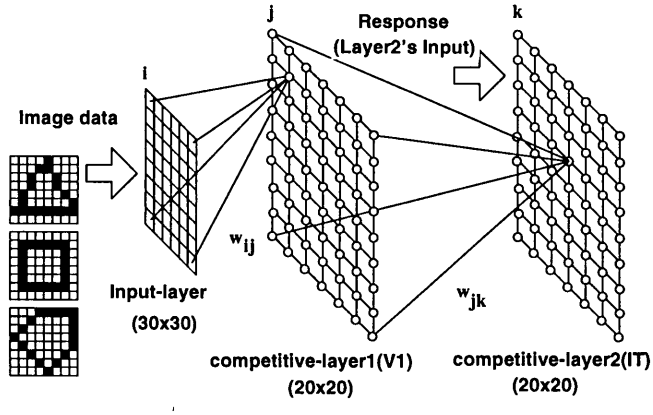
次に、勝者ノードcおよびその近傍にあるノードの重みベクトルW_jの各要素w_{ij}を次式に従って修正する。



(a) V1 野モデルのネットワーク構成



(b) 競合層の階層構造



(c) V1 野-IT 野モデル間の階層構造神経結合モデル

Figure 4.2: V1 野-IT 野の神経結合モデル

$$\Delta w_{ij} = \begin{cases} \alpha(e_i - w_{ij}) & \text{ノード j が近傍にある} \\ 0 & \text{それ以外} \end{cases} \tag{4.2}$$

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij} \tag{4.3}$$

ここで、αは学習率である。また、図4.3に示すように、近傍ノードとは勝者ノードを中心とした8-近傍距離 d の範囲に含まれるノードを指す。

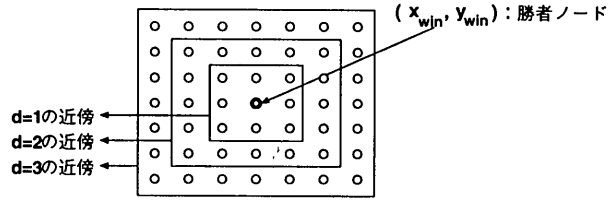


Figure 4.3: 近傍の定義

以上の操作を繰り返すことにより学習を進め、更に近傍半径 d と学習率α を徐々に減少させる。

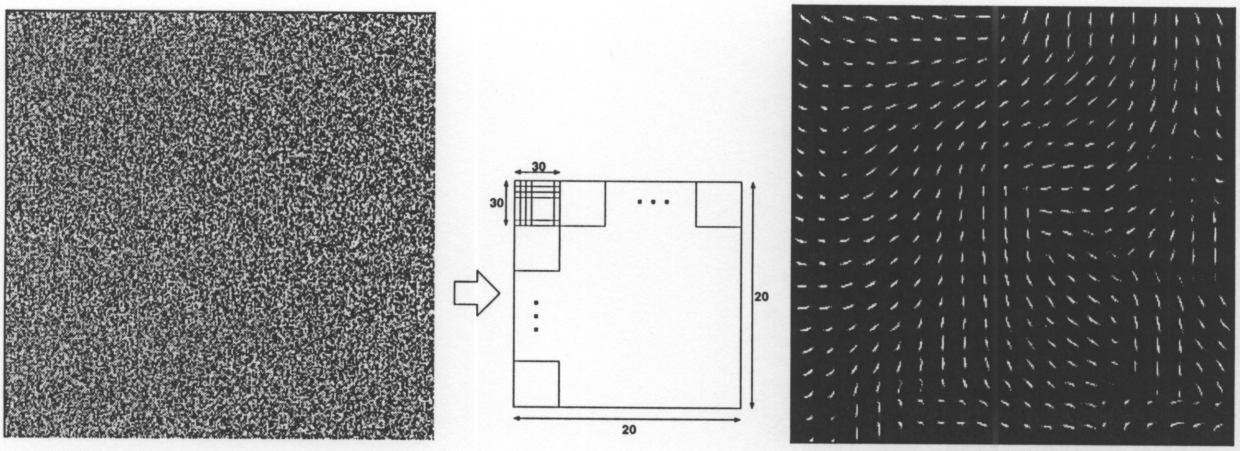
4.4 V1 野細胞の自己組織化

4.4.1 自己組織化シミュレーションと重みベクトルの可視化

図4.2(a)に示すV1野モデルを用い、V1野における方位選択性細胞の自己形成を行なう。シミュレーションでは、入力層に 30×30 個のノード（$n = 30$）、競合層に 20×20 個のノード（$m = 20$）を配置する。

各ノードが持つ結合重みベクトルの各要素は、学習の初期状態において0.0〜1.0までの乱数値で初期化される。このように構成したネットワークに対して、中心位置、傾き共にランダムな値を持つスリットパターンを25000回入力層に呈示し、先に述べた自己組織化アルゴリズムを用いて学習シミュレーションを行なう。

学習前の結合重みベクトルの初期状態を図4.4(a)に示す。結合重みベクトルは、図4.4(b)に示すように、競合層の 20×20 の各ノードに対して、そのノードが持つ結合重みベクトルの各要素の値を 30×30 の2次元平面に表示されている。各要素は0.0〜1.0の値をとり、要素の値が大きい点ほど明るくなるように表示している。また、学習率の初期値$\alpha_{\mathrm{ini}} = 0.2$に対して近傍の初期値 $d_{\mathrm{ini}} = 10$ の時の学習終了後の結合重みベクトルを図4.4(c)に示す。図4.4(c)より、競合層の各ノードにおける結合重みベクトルは、30×30 の配列の中でほぼスリット状の形状を示していることがわかる。競合層の一つのノードにおける重み結合の一つ一つは、入力層のノードと一対一に対応している。学習において結合重みベクトルは



(a) 初期状態　(b) 各ノードの結合重みベクトルの表示方法　(c) 学習終了後の結合重みベクトル

Figure 4.4: 結合重みベクトルの状態変化

入力ベクトルとの差分が小さくなるように各要素の更新を受ける。したがって入力層に与えられたスリット状のパターンが、結合重みの 2 次元パターンに反映されている。

4.4.2 V1 野細胞活性パターンの可視化

出力関数の定義

学習時においては、各ノードの出力関数は特に定義していなかった。各ノードの活性値を調べるに当たり、その出力関数を以下のように定義する。

$$f_j(u) = \frac{1}{1 + exp\{\beta(u - \theta)\}} \tag{4.4}$$

$$u = \sum_j (W_j \cdot E) \tag{4.5}$$

ここで、$f_j(u)$ を競合層のノード jの活性値、W_jは競合層のノード jが持つ結合重みベクトル、Eは入力ベクトルである。式 (4.4) はシグモイド関数と呼ばれる。βはシグモイドの立ち上がりの急峻度を決めるパラメータであり、θは閾値である。β、θを適当に調節することによって式 (4.4) は閾値関数として用いることが出来る。すなわち、ここで競合層の各ノードは典型的な人工ニューロンモデルとして定義される。

スリットパターンに対する神経細胞活性パターン

競合層ノードの活性パターンを調べるためのテストパターンは、中心の座標が入力層の中心にあり、スリットパターンを 0 度〜150 度まで 30 度ずつ回転させて作成する。学習終

了後のV1野モデルの入力層にテストパターンを呈示した時の競合層ノードの活性パターンを図4.5(a)に示す。各図の上段には呈示したスリットパターンを示した。下段は20×20の競合層ノードの活性値を立体的に表している。活性値の大きなノードほど赤く着色し、立体的にも標高を高く表示している。

図4.5(a)より、特定の傾きに対して選択的に活性化するノードは、競合層の中でまとまって存在していることが分かる。また、傾きの連続的な変化に対して競合層における活性化ノードの位置もほぼ連続に変化している。すなわち、互いに傾きの近いスリットパターンに対して活性化するノードは、競合層内でも互いに近い位置に存在している。このことから、V1野モデルの入力層と競合層との間で、スリットパターンの傾きに関してトポロジカルマッピングが学習によって形成されたことが分かる。

以上のことから、V1野モデルはKohonen自己組織化モデルに基づく学習アルゴリズムにより、特定の傾きに対して選択的に活性化する、すなわち方位選択性を持つノードを競合層上に形成できることが分かる。

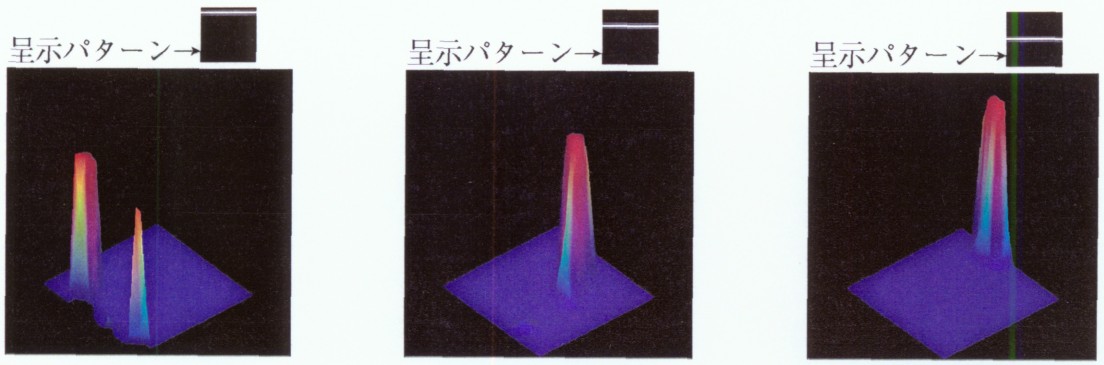
図4.5(b)に、傾き0度で、位置の変化するテストパターンを呈示した時の、競合層ノードの活性パターンを示す。傾きの異なるスリットの呈示実験と同様、スリットの中心位置に対応して、活性化するノードが競合層内で移動している。このことから、V1野モデルは、位置情報にも選択的に活性化するノードを競合層に形成できる事が分かる。

4.4.3 生理学的実験結果との比較検討

Blasdel と Salama[20] は、サルの脳の視覚皮質において、スリットパターンの傾きに対する細胞の活性分布を生理学実験によって詳しく調べた。図4.6(a) は、サルの視覚皮質におけるスリットパターンに対する活性分布を、電圧感受性のある色素を用いて可視化したものである。図4.6(a) より、特定の傾きに選択的に活性化する細胞は皮質表面において帯状あるいは斑点状に分布していることが分かる。

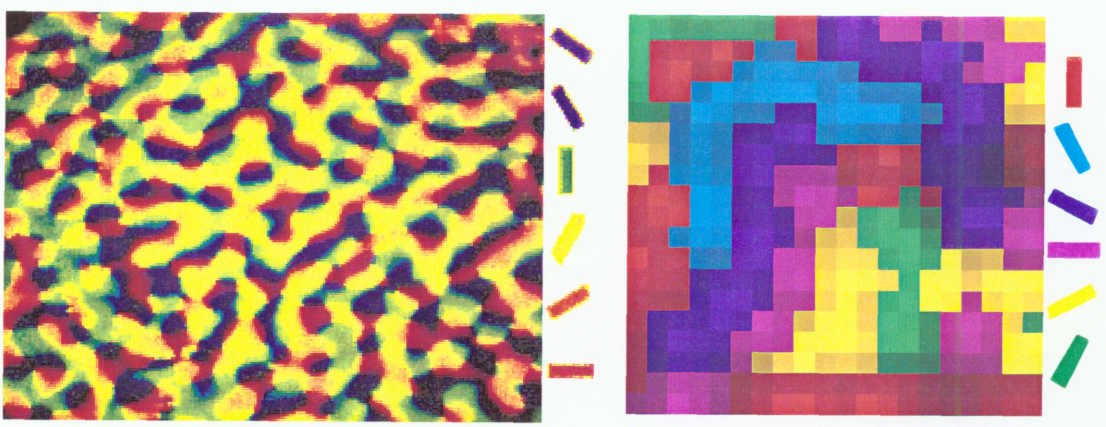
そこで、スリットパターン学習後のV1野神経結合モデルにおいて、Blasdel らと同様の観点からスリットパターンの傾きに対する競合層ノードの活性パターンを色分けし、各ノードにおける最適方位の分布を調べた。図4.6(b) が可視化の結果である。図4.6(b) の右側に示したスリットの色は、それぞれのスリットが持つ傾きを最適方位とする競合層ノードの色と対応している。図4.6(b) より、それぞれの傾きに対して選択的に活性化するノードの集合は、競合層内でほぼ帯状あるいは斑点状に分布している。この活性分布は生理学実験により得られた図4.6(a) の結果と定性的に一致することが分かる。

(a) 傾きの変化に対する活性パターンの変化



(b) 呈示位置の変化に対する活性パターンの変化

Figure 4.5: スリットパターンに対する神経細胞活性パターン



(a) 生理学実験による方位選択生細胞の分布 [20]

(b) 競合層における最適方位の分布

Figure 4.6: 生理学実験結果との比較

4.5 IT 野細胞の自己組織化

4.5.1 V1 野-IT 野の階層型神経結合モデル

V1 野モデルは自己組織化による学習により、入力層に呈示された図形に含まれる線分要素の位置と傾きを、競合層ノードの活性パターンで表現できる。ここで、方位選択性を学習した V1 野モデルの競合層（第 1 競合層）に階層的に第 2 競合層を付加した神経結合モデルを考える。第 2 競合層の各ノードは、学習済みの第 1 競合層の活性パターンを用いて自己組織化による学習を行なう。したがって、第 2 競合層の各ノードは、線分の組合せで表現された幾何学図形に対して選択的に活性化する可能性がある。そこで、図 4.2(c)に示す V1 野-IT 野モデルを用いた第 2 競合層の自己組織化学習について詳しく検討する。

4.5.2 自己組織化学習シミュレーション

入力層に呈示する学習パターンは図 4.7(a)～4.7(k)に示す幾何学図形をビットマップ化したものとする。それぞれ菱形、平行四辺形、四角形、長方形、直角三角形、二等辺三角形、三角形である。これらの図形をそれぞれ 0～350 度まで（ただし、180 度の回転で元の図形に戻るものは 0～170 度まで）、10 度ずつ回転させ合計 288 の図形パターンを作成した。288 パターンのうち 144 を学習に用い、残りの 144 は、学習終了後のテストパターンとして用いる。

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k)

Figure 4.7: 第 2 競合層における学習パターン

144 の学習パターンを用いて、第 2 競合層の自己組織化学習を行なう。学習の初期状態において、第 2 競合層の結合重み w_{jk} は 0.0～1.0 の範囲の乱数値で初期化される。第 1 競合層の結合重み w_{ij} は V1 野モデルの方位選択性の学習によって得られた結合重みベクトルを用いる。V1 野-IT 野モデルの学習では、第 1 競合層の結合重み w_{ij} は一定であり、第 2 競合層のみで自己組織化学習を行なう。

第 1 競合層の結合重みは、V1 野モデルの学習シミュレーション（学習率の初期値 $\alpha_{\mathrm{ini}} = 0.3$、近傍半径の初期値 $d_{\mathrm{ini}} = 10$）の結果を用いた。第 2 競合層の学習パラメータも同じく $\alpha_{\mathrm{ini}} = 0.3$、$d_{\mathrm{ini}} = 10$ とする。学習回数は 25000 回である。

4.5.3 IT 野モデルにおける神経細胞の活性パターン

144 個の学習パターンを用いて学習を行なった第 2 競合層に対して、学習に用いないテストパターンに対する競合層の活性パターンについて詳しく検討する。

図 4.9(a)～図 4.9(c)に、テストパターンに対する第 2 競合層の活性パターンを示す。各図の上段には入力層に呈示したテストパターンを示している。図 4.9(a)～図 4.9(c)の下に記した数字の組は、それぞれのテストパターンに対して最も強く活性化するノードの第 2 競合層内の座標である。

図 4.9(a)～図 4.9(c)より、特定の図形パターンに対して活性化するノードは、競合層内の特定の位置にかたまって存在していることが分かる。このことから、提案した階層構造を有する V1 野-IT 野モデルは、幾何学図形パターンを識別するノードを学習によって自己形成することが確認できる。ただし、複数の図形に対して最大の活性値を示すノードも形成されている。従って、最大の活性値を示すノードの座標のみで図形の識別を行なうことはできない。

また、図 4.9(b)と図 4.9(c)に示す競合層における活性領域は、いずれも競合層の左上方である。従って、図形特徴（四角形と三角形）に対する競合層の活性領域の住み分けが完全に行なわれていない事が分かる。これは、提案した V1 野- IT 野モデルが Fujita[14] らの生理学実験の結果を完全に再現できていない事を示している。

さらに、第 2 競合層における活性パターン同士の類似性は、入力層に呈示する幾何学図形同士の客観的な類似性と必ずしも一致していない。原因としては、第 1 競合層において、呈示された図形に対する活性パターンが、図形の特徴を正確に反映していないことが考えられる。入力図形の特徴をより正確に第 1 競合層の活性パターンに反映させるため、競合層の大きさ、学習率や近傍半径の減少関数、結合重みの変更方式等、学習アルゴリズムの再検討が必要である。また、V1 野-IT 野モデルの学習において、用いる図形サンプルが 144 種では少なすぎることも考えられる。従って、学習に用いる図形サンプルについても詳しい検討が必要である。

4.5.4 IT 野モデルにおけるブルドン効果の再現性に関する検討

図形の見え方に関する視覚現象の一つにブルドン効果 [17] がある。ブルドン効果とは、図 4.8(a)、図 4.8(b) に示すように、円周上に等間隔に 4～5 点を配置した場合には多角形が知覚され、図 4.8(c) に示すように点の数が 7～8 点以上になると円が知覚される現象である。

ここでは、提案した V1 野-IT 野モデルにおいてブルドン効果が再現可能であるか否かを詳しく検討する。そのために、頂点のみの図形を呈示した場合と、頂点を結んだ図形を呈示した場合の第 2 競合層ノードの活性パターンを比較する。

頂点のみの図形パターンに対する競合層ノードの活性パターンを図 4.10(a)～4.10(c) に示す。それぞれ、その活性パターンに対する呈示図形パターンを上に示している。これに対し、図 4.10(a')～図 4.10(b') は頂点を結んだ幾何学図形に対する活性パターンである。また、4.10(c') は正円を呈示した場合の活性パターンである。

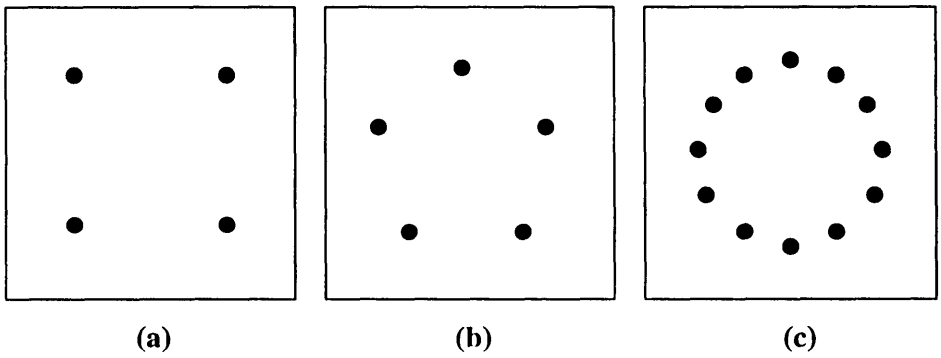


Figure 4.8: ブルドン効果

図 4.10(a) と図 4.10(a')、図 4.10(b) と図 4.10(b') をそれぞれ比較すると、頂点の情報だけを与えた場合でも、第 2 競合層は完全な幾何学図形が呈示された場合と非常に類似の活性パターンを示している。このことから、第 2 競合層は図形パターンを構成する点列ではなく、図形の見え方に対応した活性パターンを示すことが分かる。

さらに、図 4.10(c) と図 4.10(c') を比較すると、非常に類似した活性パターンであることが分かる。このことから、図 4.10(c) を呈示した場合、第 2 競合層は図 4.10(c') に示す正円を知覚していると言える。以上のことから、本モデルは人間の視覚現象の一つであるブルドン効果を再現可能であることが分かる。

4.6 まとめ

本報告では、大脳視覚野において特定の線分や図形パターンに対して選択的に活性化する刺激選択性細胞に着目し、階層型神経細胞結合モデル上での自己組織化シミュレーションを行なった。その結果、V1 野モデルにおいて、スリットパターンに対して方位選択性を持つ神経細胞を自己形成できることを確認した。また、階層構造の V1 野-IT 野神経結合モデルにおいて、単純な幾何学図形を用いた自己組織化学習シミュレーションを行ない、幾何学図形に対して選択性のある神経細胞活性パターンが現れることを明らかにした。さらに、V1 野-IT 野神経結合モデルが人間の視覚現象の一つであるブルドン効果を再現可能であることを示した。

今回提案した神経結合モデルは、第 1 競合層の学習後に第 2 競合層の学習を行なう自己組織化プロセスを仮定した。これに対して、実際の脳ではさまざまな領野が同時並列的に自己組織化を行なっている。このことを考慮したネットワークモデルとその学習アルゴリズムに関しては今後の課題である。

Figure 4.9: テストパターンに対する第 2 競合層の活性パターン

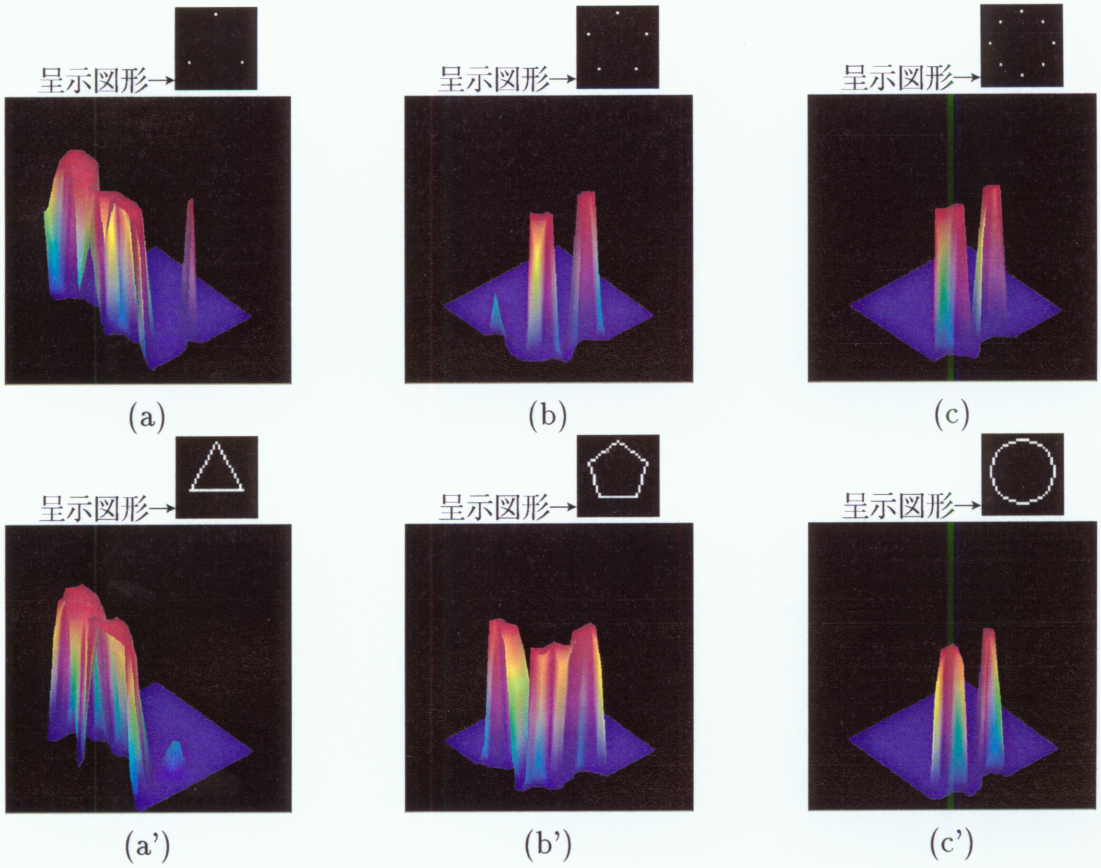


Figure 4.10: 頂点及びそれらを結んだ図形に対する活性パターンの比較

第 5 章

自己組織化ニューラルネットワークの並列学習シミュレーションと可視化

5.1 はじめに

Kohonen により提案された自己組織化マップ（Self-Organizing Map,SOM）は、ベクトル量子化や制御などの分野での応用が提案されつつある [21]。しかしながら、SOM は平面上に広がった入力層、競合層を用いており、各層間は完全結合で結ばれるため、ユニット数を増すと計算量の爆発が起こる。そのため、並列処理による高速化手法が提案されている。Obermayer ら [22] は、脳内で見られる側抑制を実現するために興奮層、抑制層の 2 層を用いたモデルを提案し、Thinking Machines 社製の超並列計算機 Connection Machine 2 上での並列学習実験を行っている。Myklebust ら [23] は、SOM のユニットを各プロセシングエレメント（PE）に割り当てるだけでなく、学習に用いる学習セットを分割して並列化する手法を提案している。また、Chwan ら [24] は直鎖網と 2 次元トーラス網への SOM の実装法を検討し、異なった領域のユニットを同一の PE に割り当てることにより負荷の均一化を図っている。しかしながら、これらの並列学習法は競合層を分割して各 PE に割り当てる競合層分割法を基本としており、SOM の学習アルゴリズムが持つ特性のために学習後半で PE の負荷が不均一となり効率的ではない。また、画像データなどの巨大な入力層を必要とする問題では、競合層分割法では十分な学習速度を得ることができない。そこで、本研究では入力層分割法による自己組織化ニューラルネットワークの並列学習法を提案する。

次節では Kohonen による SOM とその学習法について説明する。5.3節では、従来行われてきた競合層の分割ではなく、入力層を分割して並列化を行う入力層分割法について検討し、並列学習時間の解析を行う。また、競合層分割法による SOM の並列学習時間を同様に解析し両者の比較検討を行う。さらに、入力層分割法を並列計算機 nCUBE/2 に実装し、PE 間の負荷について議論する。

5.2 SOM

自己組織化マップ（SOM）は、入力信号のもつ位相を教師信号を用いることなく競合層上へ写像することができるニューラルネットワークの一種である。SOM は入力層と競合層の 2 層から成り、各層間は完全結合で結ばれている。層間の結合には重みが与えられており、学習前に乱数で初期化される。SOM を以下に述べる競合学習で学習させることにより、入力信号の位相を写像したトポグラフィックマップを得ることができる。

ある入力信号$\boldsymbol{x}$が与えられたとき、競合層のユニット i は自分が持つ重み$\boldsymbol{w}_i$と入力信号間の距離を計算する。この距離が最小の競合層ユニット c を最整合ユニット（ウィナー）として式（5.1）で定義する。

$$\begin{aligned} c &= argmin_i\{||\boldsymbol{x}-\boldsymbol{w}_i||\} \\ ||\boldsymbol{x}-\boldsymbol{w}_c|| &= min_i\{||\boldsymbol{x}-\boldsymbol{w}_i||\} \end{aligned} \tag{5.1}$$

競合層上で c の近傍に存在するユニットは、c からの距離に反比例した強度で発火し、結合重みを式（5.2）により更新する。

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + h_{ci}(t)(\boldsymbol{x}(t) - \boldsymbol{w}_i(t)) \tag{5.2}$$

式（5.2）において、t は離散時間座標であり、$h_{ci}(t)$ は近傍関数と呼ばれ式（5.3）により定義される。

$$h_{ci}(t) = h(||\boldsymbol{r}_c - \boldsymbol{r}_i||, t) \tag{5.3}$$

ここで、$\boldsymbol{r}_c, \boldsymbol{r}_i$はそれぞれ競合層におけるユニット c と i の位置ベクトルであり、$||\boldsymbol{r}_c - \boldsymbol{r}_i||$が増加するにつれて $h_{ci} \to 0$ とする。

必要な時間、十分な数の入力信号$\boldsymbol{x}$を用いて競合学習を行うことにより、競合層の各ユニットが持つ重みを参照ベクトルとするボロノイ・モザイク領域が形成される。図 5.1に 200 個の乱数を学習パターンとして、1000 回の学習を行ったときのトポグラフィックマップの生成過程を示す。図 5.1中、緑色の点は競合層ユニットの重みを 2 次元上にプロットしたものであり、ユニット間の空間的隣接関係を青色の線で示している。重み空間に一様に分布する乱数を用いて SOM の学習を行った場合、競合層ユニットの重みは重み空間に一様にひろがり、最終的に格子状のトポグラフィックマップが得られる。図 5.1(a) に示すように、学習初期では重みはランダムに初期化されているため、重み空間における重みの分布とユニットの空間的な配置には規則性が見られない。しかし、(b)→(c) と学習が進むにつれて、重みは重み空間に一様に拡散していき、同時に空間的隣接関係を反映したトポグラフィックマップが形成され始める。1000 回の学習後では、図 5.1(d) のようなトポグラフィックマップが得られる。

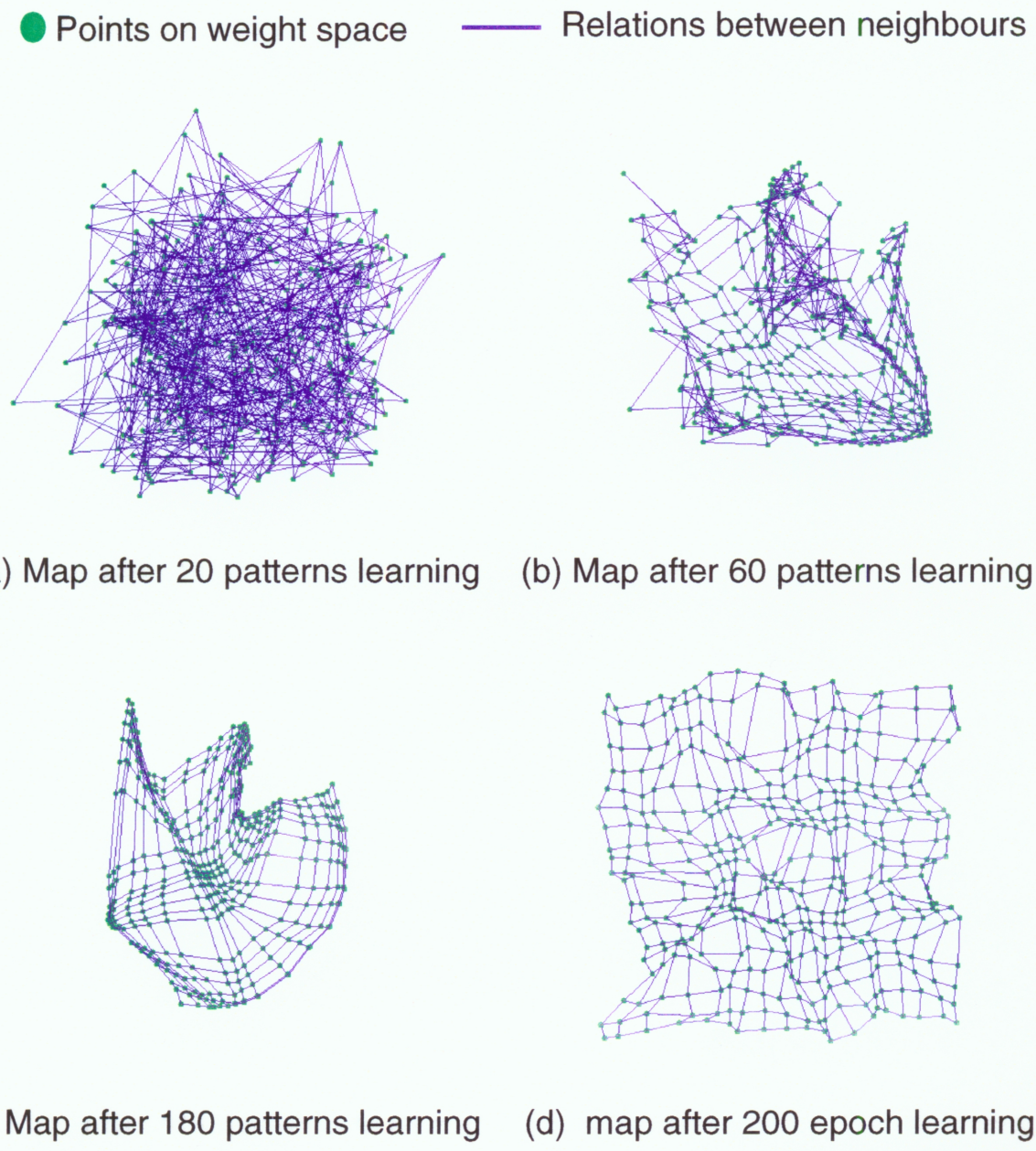


Figure 5.1: トポグラフィックマップの生成過程

5.3 入力層分割による並列学習

5.3.1 入力層分割法

SOM は入力層、競合層を 2 次元上に配置するため、ユニット数の増加は大幅な計算量増加をもたらす。そこで並列化による処理の高速化が求められているが、従来行われてきた競合層分割による並列学習では PE 間の負荷が不均一であったり、画像などの高次元入力では十分な高速化が行えないなどの問題が存在する。以下では、PE 間の負荷の不均一が原理的に発生しない入力層分割による並列化を提案し、その性能を評価する。

図 5.2に、入力層分割の概念を示す。各 PE は入力層のユニット一つを担当し、競合層の各ユニットへの重みを持つ。したがって、競合層のどのユニットがウィナー、あるいは近傍となって重み更新を行っても、すべての PE が同等の処理をするため負荷の不均一は発生しない。

対象とする SOM は入力層、競合層の 2 層とし、それぞれ $M \times M$、$N \times N$個のユニットからなり、ある時刻 t における入力ユニット i と出力ユニット j間の重みを W_i^j $(i = 1, \cdots, M^2,\ j = 1, \cdots, N^2)$ とする。この SOM を $M \times M$個の PE に分割し、各 PE は N^2 次元の重みベクトルを持つ。学習パターン $T = \{t_1, \cdots, t_{M^2}\}$ が入力層ユニットに入力された場合を考える。各 PE は学習パターンで対応する要素を用い、入力ベクトル要素 t_iと重みベクトル W_i^jとの間のユークリッド距離 d_i^jを式 (5.4) に基づき計算する。

$$d_i^j = \sqrt{(t_i - W_i^j)^2},\ j = 1, \cdots, N^2 \tag{5.4}$$

各 PEi が d_i^jを計算した後で、i について d_i^jを式 (5.5) により集計し、結果を全 PE にブロードキャストする。

$$d^j = \sum_{i=1}^{M^2} d_i^j \tag{5.5}$$

その結果、競合層ユニット jと学習パターン間の距離 d^jをすべての PE が持つことになる。各 PE は独立に d^jが最も小さいものをウィナー c として選び、式 (5.6) などの近傍関数 h_{cj}を用いて c とその近傍 $N_c(t)$ の重みを更新する。

$$h_{ci} = \begin{cases} \alpha(t) & i \in N_c \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

したがって、各 PE は同一個数の重み要素を更新することになり、負荷の不均一は原理的に発生しないことになる。図 5.3に、入力層分割法による SOM の並列学習アルゴリズムを示す。

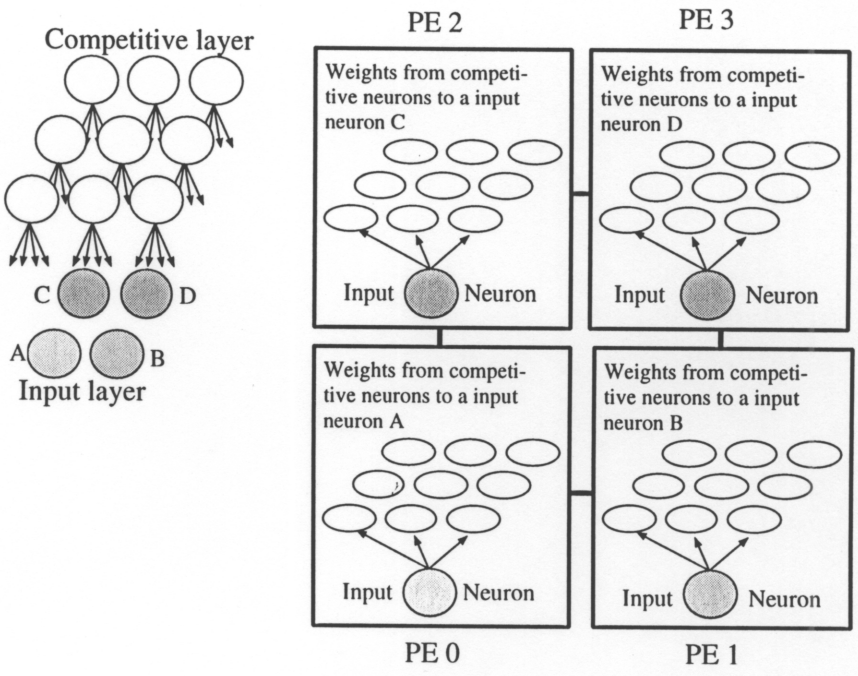


Figure 5.2: 入力層分割法の概要

5.3.2 入力層分割法の学習時間

入力層分割を用いた並列 SOM の学習時間について考察する。入力層、競合層はそれぞれ $M \times M$、$N \times N$の 2 次元格子を構成しており、$N_{PE} = M^2$個の PE を持つ並列計算機に実装するとする。近傍関数$\alpha(t)$、近傍範囲 $N_c(t)$ はそれぞれ式 (5.7)、式 (5.8) とした。

$$\alpha(t) = \alpha(0)(1.0 - \frac{t}{T}) \tag{5.7}$$

$$N_c(t) = N_c(0)(1.0 - \frac{t}{T}) \tag{5.8}$$

ここで、Tは最大学習回数とし、$\alpha(0) = 0.4$、$N_c(0) = \frac{N}{2}$とした。また、学習時間を導出するにあたり、加減算に要する時間を t_{add}、乗除算に要する時間を t_{multi}、平方根を計算するのに要する時間を t_{sq}、ウィナーを決定するのに要する時間を T_s、PE 間の通信時間を t_{comm} とした。

学習パターンは各要素に分解され、各要素が PE に分配される。各 PE では学習パターン要素と競合層ユニットへの重みとの 2 乗誤差を計算する。これに必要な時間 T'_dは次式で表される。

$$T'_d = N^2(t_{add} + t_{multi}) \tag{5.9}$$

次に競合層中のユニットで最も学習パターンに近いユニットを決定するが、これには各 PE が持つ誤差要素を集計する必要がある。この集計には簡単化のため、各 PE が一つのホ

Algorithm 5.1 入力層分割法

1 Initialize $W_i^j(0)$.
2 Distribute t_i to each PE i.
3 **do** on each PE i,
4 **for** $j = 0$ to N^2,
5 $d_i^j = ||W_i^j(t) - T_i||$.
6 broadcast d_i^j and calclate $d^j = \sum_i d_i^j$, then send it to PE i.
7 decide winner $c = \arg\min_j d^j$ and neighbours $N_c(t)$ on each PE.
8 $W_i^j(t+1) = W_i^j(t) + h_{cj}(t)(W_i^j(t) - t_i)\ j \in N_c(t)$
9 **while** $d^c < LIMIT$, or learn for max iterations.

Figure 5.3: 入力層分割学習法のアルゴリズム

ストに誤差要素を送り、ホストで加算処理後各 PE にブロードキャストするというモデルを用いる。すべての PE がホストにデータを送るのに必要な時間が $M^2 t_{comm}$、ホストが送られたデータを加算する時間が $(M^2-1)t_{add}$、加算した結果の平方根を計算するための時間が $N^2 t_{sq}$、計算結果を全 PE に送信するのに必要な時間が $M^2 t_{comm}$ であるので、誤差集計に必要な時間 T_c' は次式のようになる。

$$T_c' = (M^2-1)t_{add} + N^2 t_{sq} + 2M^2 t_{comm} \tag{5.10}$$

各 PE はホストから受信した集計後の誤差をもとにウィナーと近傍を決定する。近傍 $N_c(t)$ 内の競合層ユニット数は $N_c^2(t)$ なので、重み変更に必要な時間 T_{uw}' は式 (5.11) で表される。

$$T_{uw}' = (2t_{add} + t_{multi})N_c^2(t) \tag{5.11}$$

以上より、学習パターン一つを入力層分割 SOM で学習するのに必要な時間 $T_{total}^{'som}$ は式 (5.9)、式 (5.10)、式 (5.11) と T_s との総和であり、式 (5.12) で表される。

$$\begin{aligned} T_{total}^{'som} &= T_d' + T_c' + T_n + T_s + T_{uw}' \\ &= \left\{M^2 + N^2 + 2N_c^2(t) + 1\right\} t_{add} + \left\{N^2 + N_c^2(t) + 4\right\} t_{multi} \\ &\quad + N^2 t_{sq} + T_s + 2M^2 t_{comm} \end{aligned} \tag{5.12}$$

入力層分割法を用いて、N_p 個の学習パターンを N_{epoch} 回学習するのに必要な時間は、式 (5.12) に N_p を乗じ、それを N_{epoch} まで加算した式 (5.13) で表されることになる。

$$T^{'som} = \sum_{t=1}^{N_{epoch}} N_p T_{total}^{'som} \tag{5.13}$$

5.3.3 競合層分割法の学習時間

本節では従来の競合層分割法による学習時間について議論する。入力層分割法と同じ条件を用い、1PE に競合層の1ユニットを割り当てる。学習パターンは1パターン毎にすべての PE にブロードキャストされ、並列に競合層ユニットの誤差を計算する。これに必要な時間 T_d''は式（5.14）で表される。

$$T_d'' = M^2(t_{add} + t_{multi}) + t_{sq} \tag{5.14}$$

各 PE は単一の競合層ユニットを割り当てられているので、競合層全体でのウィナーを決定するためには一旦計算した誤差をホストに送る必要がある。ホストはウィナーを決定後、各 PE にウィナーをブロードキャストする。これに必要な時間 T_c''は式（5.15）となる。

$$T_c'' = 2N^2 t_{comm} + t_s \tag{5.15}$$

ウィナー決定後、近傍半径および学習率の決定には、他のモデルと同様に T_nの時間が必要である。重みの更新は近傍範囲内のユニットを割り当てられた PE のみが行い、他の PE はアイドル状態で処理の終了を待つとすると、重み更新に必要な時間 T_{uw}''は式（5.16）で表される。

$$T_{uw}'' = M^2(2t_{add} + t_{multi}) \tag{5.16}$$

以上より、競合層を分割したときの並列モデルにより学習パターン1つを処理するのに必要な時間 $T_{total}''^{som}$は式（5.17）となる。

$$\begin{aligned} T_{total}''^{som} &= T_d'' + T_c'' + T_n + t_s T_{uw}'' \\ &= (3M^2 + 2)t_{add} + (2M^2 + 4)t_{multi} \\ &\quad + t_{sq} + T_s + 2N^2 t_{comm} \end{aligned} \tag{5.17}$$

以上より N_p個の学習パターンを N_{epoch}回学習するのに必要な時間 T''^{som}は式（5.18）で表される。

$$T''^{som} = N_{epoch} N_p T_{total}''^{som} \tag{5.18}$$

5.3.4 入力層分割法と競合層分割法の比較

逐次処理 SOM と、式（5.12）による入力層分割、および式（5.17）による競合層分割による並列 SOM の処理時間を比較する。$M = N = n$、$N_{epoch} = 1000, N_p = 100$、$T_s = n \times t_{add}$とし、理想並列計算機上での各学習法による学習時間を図 5.4に示す。並列学習における学習時間の計算では、ユニット数と同数の PE を使用すると仮定している。また、各学習時間の計算における各処理時間は、nCUBE/2 での実測により $t_{add} = t_{multi} = 1\mu sec., t_{sq} = 10\mu sec., t_{comm} = 100\mu sec.$とした。

図 5.4に示すように、逐次処理と比較して並列処理を用いた学習は大幅な高速化が達成できる。これは、逐次処理では通信時間は各層の軸上のユニット数 n に対して $\mathcal{O}(n^4)$ で処理

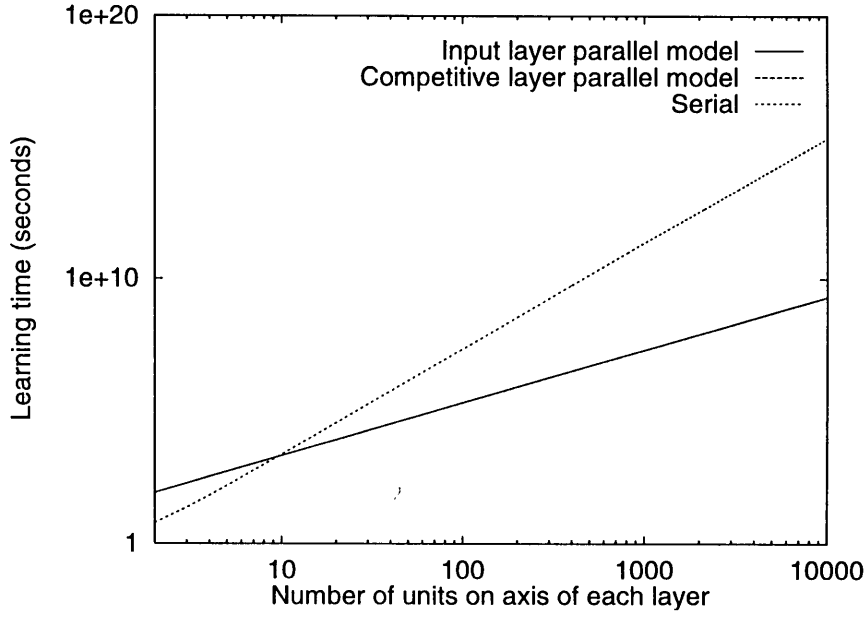


Figure 5.4: 逐次処理および理想並列計算機上での入力層分割と競合層分割学習法の学習時間

時間が増加するのに対し、並列化時には $\mathcal{O}(n^2)$ であり、ネットワークのサイズが大きくなるにつれ並列化による効果が大きくなる。モデルによるシミュレーションでは、100×100 の SOM では 1 万個の PE を用いた並列処理により約 100 倍の、1000×1000 の SOM では約 10000 倍の高速化が達成できる。入力層分割法と競合層分割法を比較した場合、競合層分割法の方がわずかに高速である。これは競合層分割法での 2 乗誤差の計算が並列で t_{sq} で行うことができるのに対し、入力層分割法は $N^2 t_{sq}$ となることが一因と考えられる。

図 5.5に、入力層分割法を nCUBE/2 に実装したときの学習時間とモデルによる計算時間を示す。nCUBE/2 における配列加算通信に要する時間はメッセージ長を $Mlen$、使用する PE 数を N_{PE} とすると式（5.19）で近似できる。

$$t'_{comm}(N_{PE}, Mlen) = (0.0005 + 3.0 \times 10^{-6} M_{len}) \log N_{PE} \tag{5.19}$$

入力層分割法での通信時間は、配列加算通信を用いることにより $2M^2 t_{comm}$ から $t'_{comm}(N_{PE}, Mlen)$ となる。t'_{comm} において、$N_{PE} = M^2, Mlen = N^2$ である。図 5.5に示した SOM の規模は、競合層 10×10、入力層 $2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16$、$N_p = N_{epoch} = 100$ である。図 5.5より、モデルによる計算時間は実装による実験結果と誤差 10%以内で一致している。この結果から、SOM の学習時間検討の手段として式（5.13）、式（5.18）を用いることは妥当と考えられる。

さて、図 5.6に PE 数とメッセージ長を考慮した通信時間を用いた場合の並列学習時間を示す。図 5.6より、理想並列計算機上での比較では差が見られなかったが、メッセージ長や PE 数による通信遅延を考慮すると競合層分割法に比べて入力層分割法の学習時間が長く

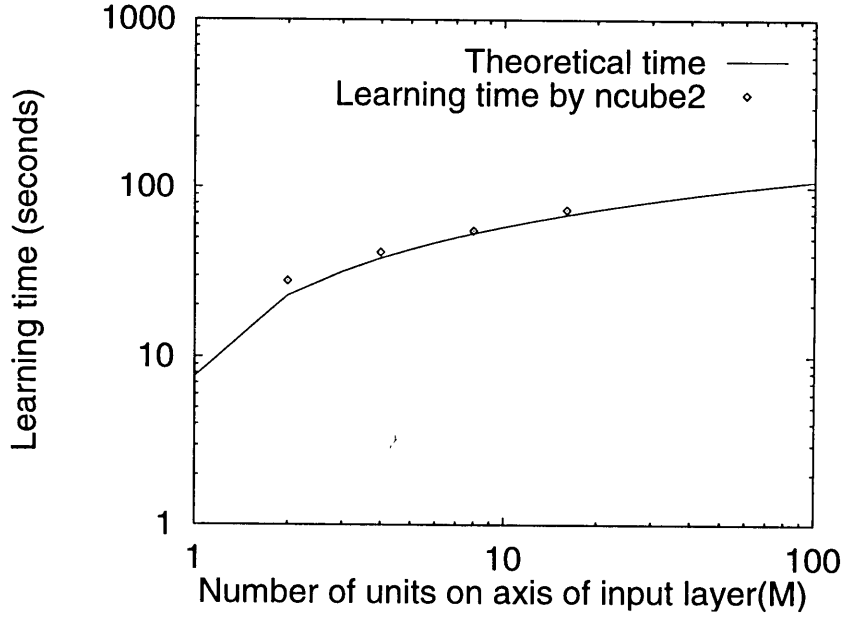


Figure 5.5: 並列 SOM 学習法のモデルと実装結果の比較

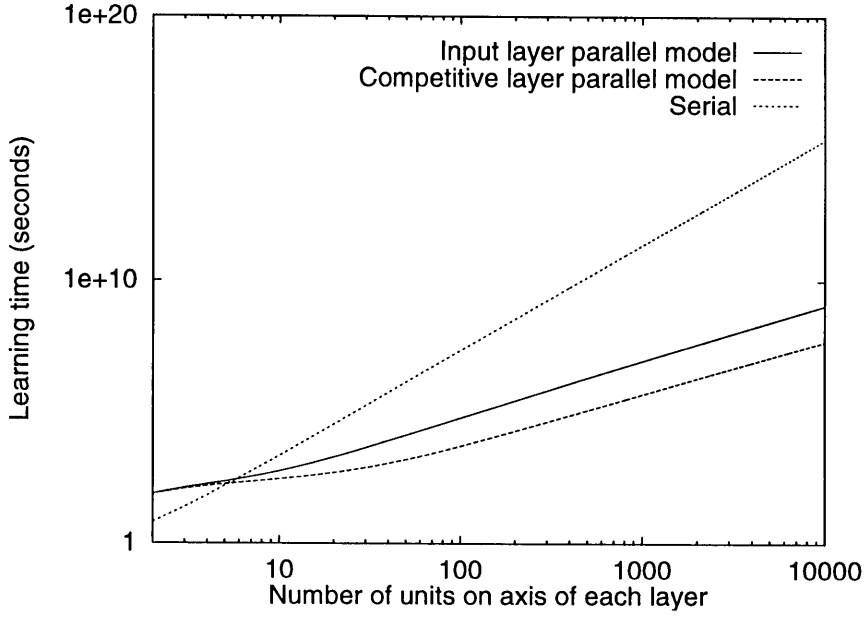


Figure 5.6: 通信遅延を考慮した並列 SOM の学習時間の比較

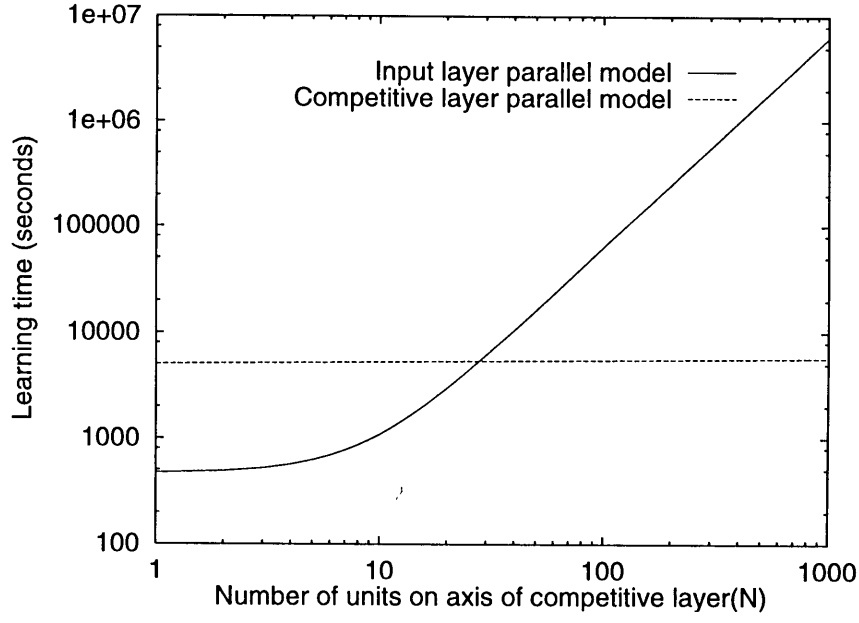


Figure 5.7: 入力層を 100×100 としたときの学習時間

なっている。競合層分割法では通信で送られるメッセージ長は一定であるが、入力層分割法では競合層のユニット数に比例してメッセージ長が増加するため、ネットワークの規模が大きくなるにしたがって一回の通信時間が増大する。このことから、各層で同一規模のユニットを持つ SOM では、競合層分割法の方がやや高速となる。

次に、入力層と競合層のユニット数が異なる場合について検討する。図 5.7に入力層を 100×100 として競合層のユニット数を変化させたときの学習時間を、図 5.8に競合層を 100×100 に固定して入力層のユニット数を変化させたときの学習時間を示す。このとき、$N_p = 100, N_{epoch} = 1000$ である。

入力層を 100×100 に固定した場合、図 5.7より競合層ユニット数が少ない間は入力層分割法の方が高速である。しかし、競合層のユニット数が増加するにしたがって入力層分割法の学習時間は長くなり、競合層分割法の方が高速になる。入力層分割法では、式 (5.12) より学習時間が競合層サイズ N^2により主として支配されているため、競合層のサイズが大きくなると学習時間が長くなる。これに対し、式 (5.17) で示される競合層分割法の学習時間では Nを含む項はなく、通信時間が $t'_{comm}(N^2, 1.0)$ で Nを影響を受けるのみで、競合層サイズが変化しても学習時間はほとんど変化しない。

図 5.8では競合層を 100×100 に固定し、入力層のサイズを変化させたときの学習時間を示す。入力層分割法では式 (5.12) から分かるように t_{add} の係数と通信時間 $t'_{comm}(M^2, N^2)$ に Mが含まれているため、入力層サイズが大きくなると学習時間が徐々に長くなっていく。しかし、競合層分割法では式 (5.17) より学習時間は Mに支配されるため、入力層分割法よりも学習時間の増加割合が大きく、入力層のサイズがおよそ競合層の 4 倍以上では入力層分割法よりも学習時間が長くなる。

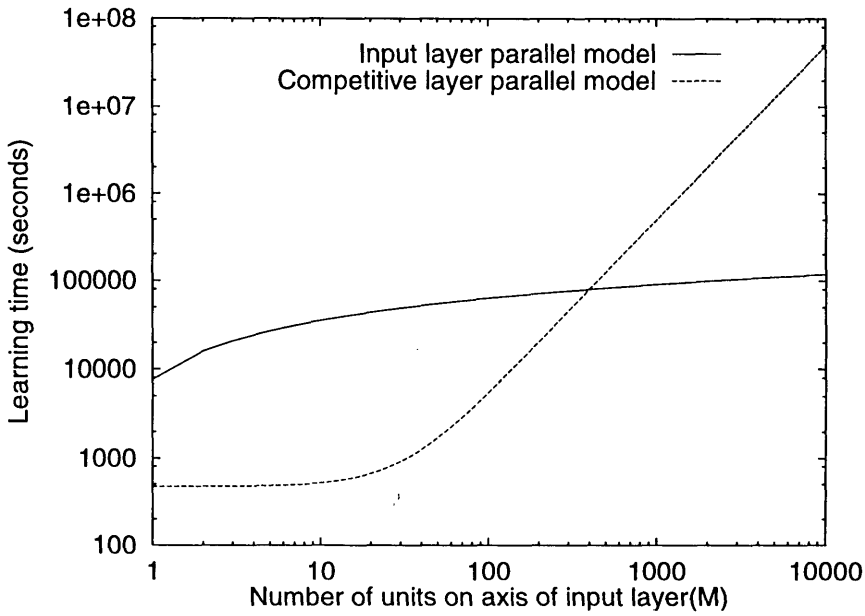


Figure 5.8: 競合層を 100×100 としたときの学習時間

以上より、提案した入力層分割法は画像符号化などで多く用いられている大規模な入力層を持つ SOM に適した学習法であるといえる。

5.4 並列学習シミュレーション

5.4.1 学習時間

並列計算機 nCUBE/2 を用いて入力層分割法の並列学習時間について検討する。並列学習シミュレーションに用いた SOM は、入力層を $2 \times 1, 2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16$、競合層を 10×10 とし、学習パターン数を 50, 100, 150、最大学習回数を 100 として学習を行った。このときの学習時間を図 5.9に示す。

図 5.9より、学習時間は学習パターン数に比例して増加しており、式 (5.13) で示したモデルと同様の結果となっている。PE 数の変化による学習時間の増加を見ると、PE 数の log、すなわちハイパーキューブ網の次元数に比例して学習時間が増加している。これは、入力層分割法での通信時間が式 (5.19) のようにハイパーキューブ網の次元数に比例して増加するためである。

図 5.10に、入力層を 4×4 に固定して競合層のサイズを変化させたときの学習時間を示す。式 (5.12) より、入力層分割法では四則演算の処理時間は N^2 に比例し、通信時間もメッセージ長の増加といった形で競合層サイズの影響を受ける。図 5.10より、競合層の軸上のユニット数の変化に対してほぼ軸上の競合層ユニット数の 2 乗で学習時間が増加している

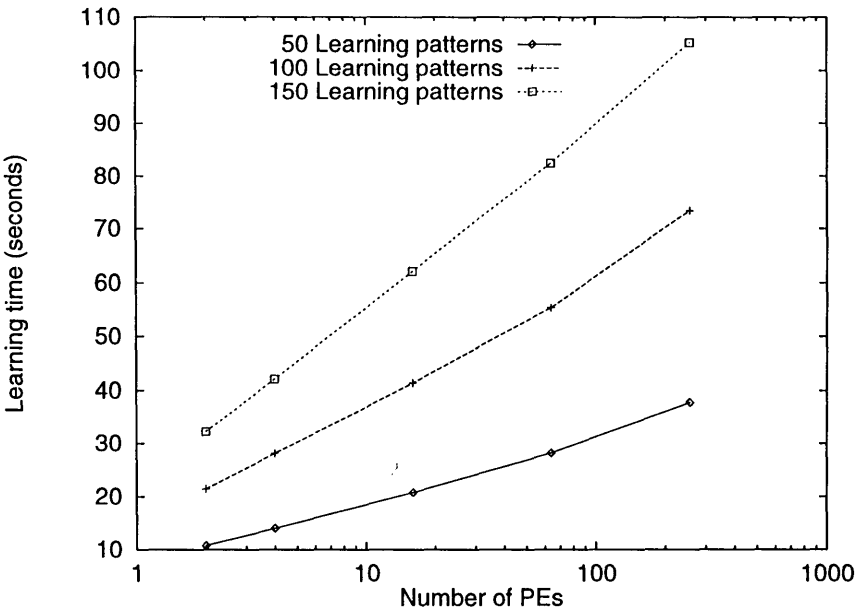


Figure 5.9: 競合層サイズを 10×10 として入力層のサイズを変化させたときの入力層分割法の並列学習時間

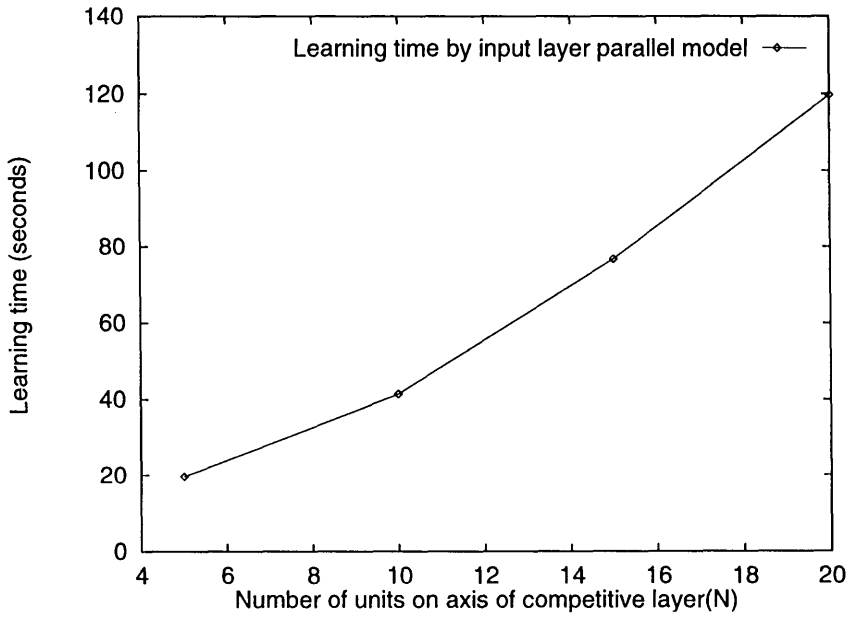


Figure 5.10: 4×4 の入力層で競合層サイズを変化させたときの並列学習時間

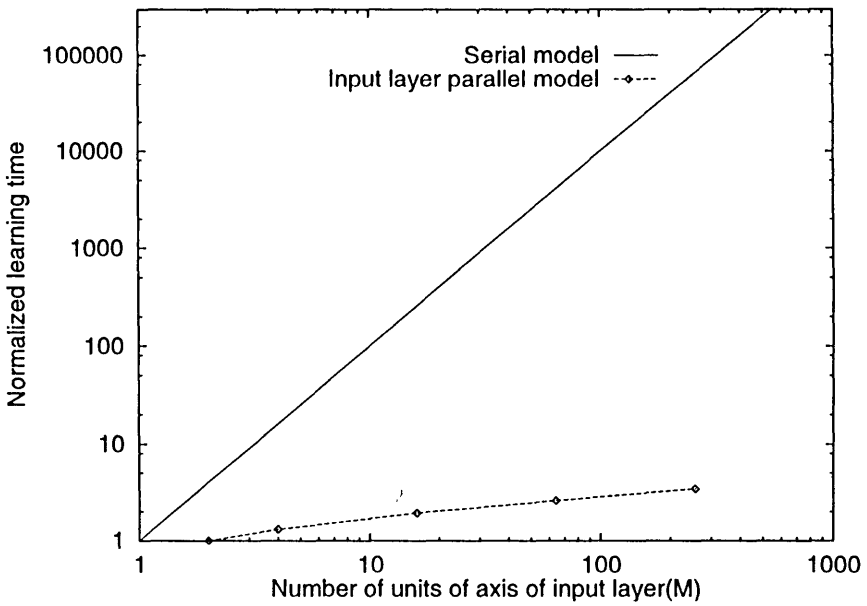


Figure 5.11: 入力層ユニット数の変化に対する相対学習時間

ことが分かる。

図 5.11に、入力層のユニット数が変化したときの逐次計算モデルと入力層分割法での相対学習時間を示す。入力層分割法の相対学習時間は 10×10 の競合層で nCUBE/2 で学習したときの値を使用した。図 5.11から、逐次処理では入力層サイズの増加に比例して学習時間が長くなることが分かる。これに対して入力層分割法では、入力層のサイズが約 100 倍となっても学習時間の増加は約 3.5 倍に留まっており、並列化により大幅な学習時間の短縮が実現されていることが分かる。

5.4.2 並列学習における負荷分布

入力層分割法及び競合層分割法での各 PE 間の負荷分布について検討する。入力層分割法、競合層分割法ともに入力層、競合層のサイズを同一とし、100 個の学習パターンを 100 回学習した。入力層分割法については nCUBE/2 上での各 PE の学習時間を用いて負荷を検討する。競合層分割法については、重み更新対象ユニット以外を受け持つ PE はアイドルとなるため、各 PE の学習時間は重み更新回数に比例する。したがって、学習時間に代えて各 PE での重み更新回数を元に各 PE での負荷分布について検討する。

図 5.12に、SOM のサイズを 4×4 としたときの入力層分割法および競合層分割法での各 PE の学習時間と重み更新回数を示す。入力層分割法では、PE0 を除きほぼ同一の学習時間となっている。配列加算通信では、各 PE が持つ誤差を PE0 にいったん集めたあとその結果をブロードキャストするため、PE0 での学習時間がやや長くなる。入力層分割法において、負荷の差はおよそ 3%程度であった。これに対して重み更新回数がそのまま学習時間

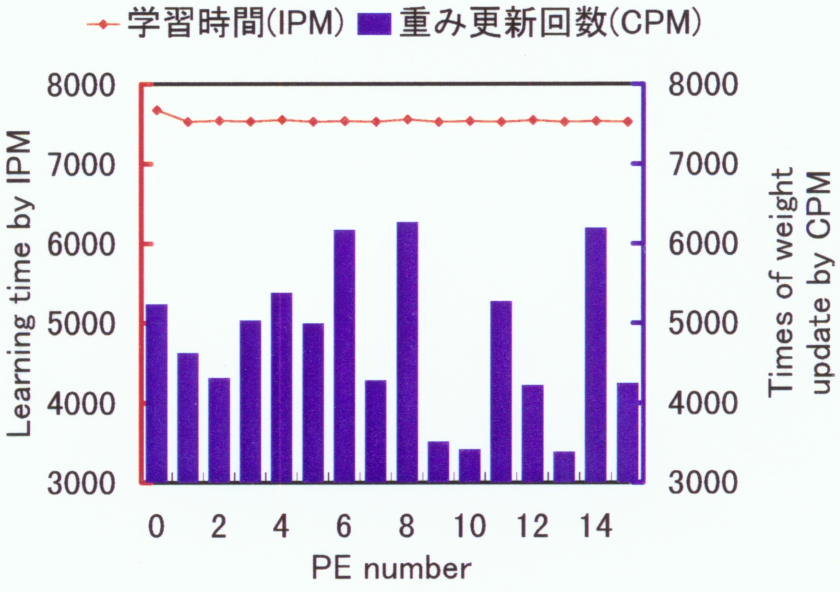


Figure 5.12: 各 PE の学習時間と重み更新回数（16PEs）

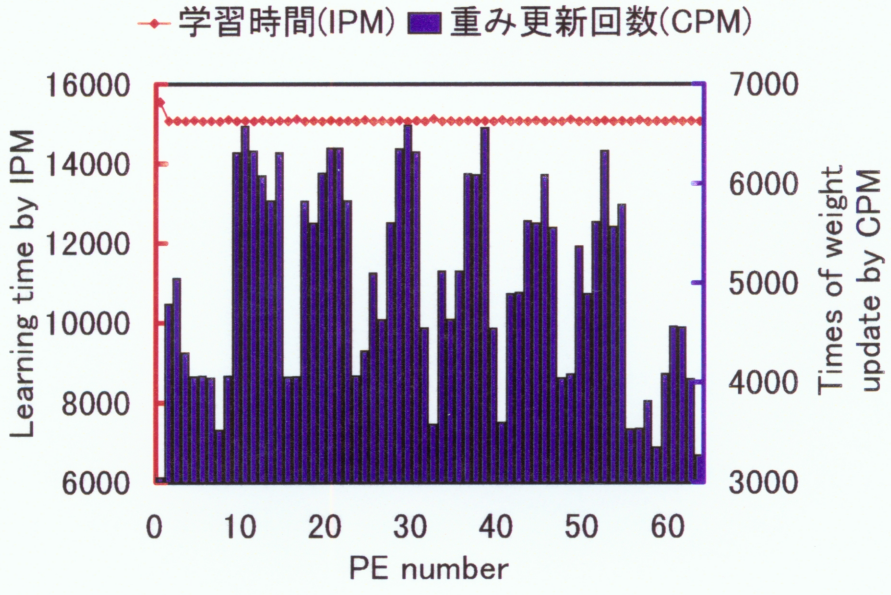


Figure 5.13: SOM での各 PE の学習時間と重み更新回数（64PEs）

の差としてあらわれる競合層分割法では、負荷の差はおよそ2倍となっている。図5.13で示した8×8のSOMでもこの数値はほぼ同一であり、入力層分割法では事実上負荷の不均一が発生していないことが分かる。

以上より、入力層分割法ではPE間の負荷の不均衡は発生せず、並列学習法として有効であることが分かった。

5.5 まとめ

KohonenによるSOMは従来の階層型ニューラルネットワークと異なりユニットが2次元マップ状に配置されており、類似した入力に対して近傍のユニットが反応するというトポグラフィックマッピングを行うことができるという特徴がある。こうした特徴は、ベクトル量子化などの分野で応用され、相応の成果をあげつつある。しかし、多数のユニットとその相互結合を用いるため、ネットワークが大きくなると学習時間が膨大になるという、ニューラルネットワークに共通した大きな問題がある。

こうした問題を解決するため、並列処理によるいくつかの高速学習法が提案されている。しかしながら、SOMの学習アルゴリズムのため従来の競合層分割による並列学習ではPE間の負荷が不均衡になり、学習後半にアイドルとなるPE数が増えてしまう。画像などのユニット数が膨大になる問題や、多数の入力ユニットを必要とする多次元情報の分類を行う問題では十分な高速化が行えないという問題があった。

本研究では、PE間の負荷の不均一が原理的に発生しない入力層分割法によるSOMの並列学習法を提案し、その学習速度について検討した。その結果、256PE（16×16の入力層）を使用した時の各PEの負荷のばらつきは約3%となり、優れた負荷分散性を示した。これに対し、従来の競合層分割法では、PE間の負荷は最大で約2倍のばらつきとなることを示した。

学習時間の解析では、100×100程度のSOMの学習においてユニット数と同数のPEを用いることにより、逐次処理の場合と比較して約3000倍の高速化が達成できることを示した。また、ネットワークの規模が大きくなればなるほど、並列化による高速化の効果は大きくあらわれる。ここで提案した入力層分割法は、入力層と競合層のユニット数が同程度では従来の競合層分割法ほぼ同じ程度の学習時間である。競合層のユニット数が入力層のユニット数よりも多くなると、入力層分割法の方が高速となる。したがって、入力層分割法は膨大な数の入力層ユニットを必要とする画像符号化などの分野や、多数の入力情報を分類・統合するようなクラスタリングなどの問題に対して効果的であると考えられる。

第 6 章

ウェーハスタック型超並列システムの熱分布シミュレーションと可視化

6.1 はじめに

超並列システムへの課題の 1 つとして、故障回避の問題がある。少数の並列処理システムでは、単一の PE の故障率が十分低ければ、システムとしての信頼性を確保することができる。しかしながら、数百万規模の PE から成る超並列システムでは、たとえ一つ一つの PE の故障率が低くても、全体としての故障率がかなり大きくなってしまうという問題がある。このため、故障した PE を切り離し、この機能を別の PE で代替する必要がある。

故障した PE の機能を配線切替により別の PE で代替する故障回避方式は、あからじめ予備の PE を用意しておき、スイッチを切り替えることにより、故障 PE の機能を予備 PE に振り替えることによって行なわれる。配線切替による故障回避の問題は、まず網の中核となる PE(Core PE) と、故障時に Core PE の機能を代替する予備 PE(Spare PE) をどのように配置し、代替が可能となるような配線をどのように行なうかという故障回避アーキテクチャの問題と、実際に故障が起った時に、どの予備 PE でどの Core PE の機能を代替するかという代替アルゴリズムの問題の 2 つがある。

故障回避アーキテクチャとして、格子結合に関して、$N \times N$の PE アレーに、1 行 1 列または 2 行 2 列の冗長 PE(予備 PE) を付加し、スイッチを切り替えることにより、論理的な $N \times N$の故障の無い PE アレーを得る方法が Kung らによって提案されている [25]。これは PE と PE の間に、それぞれ 1 本または 2 本のトラック及びスイッチ回路があり、故障の分布によりスイッチを切り換えシフトを行なうことにより、故障 PE を救済する。また、循環型ハイパーキューブ網 (CCC) の故障回避アーキテクチャについて、故障回避が可能であり、かつ面積の使用効率の良い実装方式が求められる [26]。しかしながら、現在までに、RDT や PEC などのような、トーラス網にバイパスリンクを付加して構成される相互結合網に関する故障回避アーキテクチャは、提案されていない。

一方、故障した PE をどの PE で置き換えるかという、代替アルゴリズムについて、格子網では古くから研究がなされ、グラフ理論を用る方法 [25] や、ローカル情報を用いる方

法 [27] が提案され、高い欠陥救済率を得ている。

超並列システムでは、平面の実装だけでは、基板面積やウェーハ面積が非常に大きくなる。このため、3 次元的にシステムを実装する必要がある。3 次元実装の際には、内部の発熱をどのように放熱するかが問題となる。ウェーハスタックシステムの試作例 [28] では、ウェーハ間の信号を接続するスプリングブリッジによって、高さ方向 (ウェーハ面に直角方向) に、放熱している。しかしながら、スプリングブリッジによる高さ方向の放熱は、スプリングブリッジの位置や数に大きく影響される。

本章では、最初に 1 次元リニアアレイ配置の 1D-SRT の故障回避方式として、スイッチ束を用いた故障回避アーキテクチャを提案する。この 1D-SRT の故障回避アーキテクチャを 2 次元に拡張し、2D-SRT の故障回避アーキテクチャについて議論する。2D-SRT はトーラス網に基づき構成されるので、格子結合網の故障回避アルゴリズムを 2D-SRT に適用する。

3 次元のウェーハスタック実装の放熱問題に対しては、故障回避アルゴリズムに、方向性を持たせることにより、ウェーハ周囲からの放熱特性が良い実装方式を提案し、ウェーハスタック温度分布とシステム歩留まりについて評価を行なう。また温度分布について可視化を行なう。

6.2 故障回避アーキテクチャ

6.2.1 1D-SRT の故障回避アーキテクチャ

図 6.1 (a) に再構成可能な 1D-SRT のアーキテクチャを示す。直鎖上に並べられた N 個の PE が中央部に置かれ、これを挟むように両端に 1 つまたは複数のスペア PE が置かれる。中央部の PE と両端のスペア PE は同等の機能を有する。各 PE は使用中の状態と未使用の状態の 2 つの状態をとることができる。故障 PE は未使用の状態にされ、隣接する PE によって故障 PE の機能が代替される。代替を繰り返し行うことをシフトと呼び、シフトにより故障 PE を回避する。

シフトを行なうために、各ノードはレベル 0 からレベル l_{max} までのリンクを束ねたリンクバスに、スイッチ束を介して接続される。スイッチ束中には level-1 から level-l_{max} までの l_{max} 個のスイッチがある。リンクバスには level-0 のリンクも含まれるが、level-0 のスイッチはスイッチ束とは独立して扱う。各スイッチは Cross モードと Connect モードの 2 つの状態をとることができ、スイッチ束中のスイッチは、0 または 1 つのスイッチだけが Connect モードをとり、他のスイッチは Cross モードにセットされる。一方、level-0 のスイッチは、スイッチ束の状態に拘束されずに 2 つの状態をとることができる。

故障 PE の機能を補うために、横方向 (図では右方向) にシフトするが、このためには、まず故障 PE を切離し、次にノードのレベルを再定義する。故障 PE の切離しは、level-0 のスイッチ及びスイッチ束の全スイッチを Cross モードにすることにより行なわれる。次に、シフトによりノードのレベルの再定義を行なう。故障 PE の機能を隣接する PE で代替するために、隣接 PE のレベルが故障 PE のレベルにセットされる。同様に、代替 PE に隣接する PE のレベルは代替 PE が保持していたレベルにセットされる。この手続きを再帰的

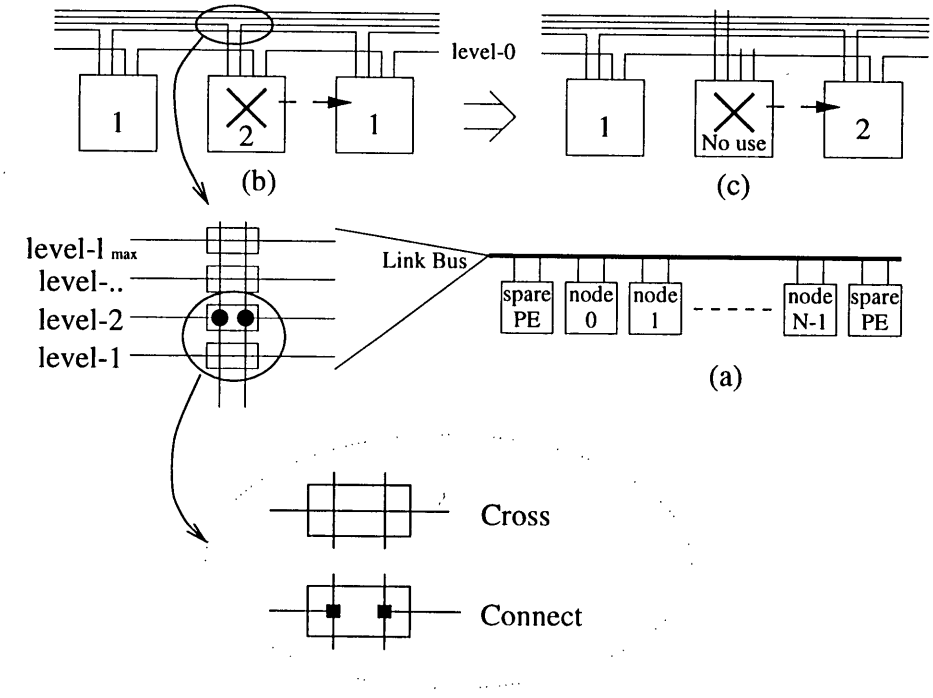


Figure 6.1: 再構成可能な 1D-SRT アーキテクチャ

に行なうことにより、レベルの再定義が行なわれる。レベルの再定義前のレベル l の PE の状態は、スイッチ束のうちレベル l のスイッチのみが Connect モードになっている (図 6.1 (b))。レベルの再定義後、この PE がレベル k にセットされるとすると、スイッチ束のレベル l のスイッチは Cross モードにセットされ、代わってレベル k のスイッチが Connect モードにセットされる (図 6.1 (c), *level* $1 \Rightarrow 2$)。

6.2.2 リンク故障の回避

リンクバスとスイッチ束の使用により、リンクバス中の任意の配線を任意のレベルのリンクとすることができる。そこで、リンクバス中の配線の故障 (断線や短絡) を回避するために、リンクバスに、本来必要な本数以上の配線を、予め用意しておき、これを予備リンクとする。リンクバス中の配線の何本かが故障している場合、スイッチ束の故障しているレベルのスイッチを Cross モードにし、代わりに予備リンクのなかの 1 つを Connect モードにすることによって、リンク故障を回避することができる。

6.2.3 2D-SRT の故障回避アーキテクチャ

図 6.2に再構成可能な 2D-SRT のアーキテクチャを示す。中心部に $N \times N$のコア PE を置き、スペア PE がコア PE の周囲に配置される。PE と PE の間にはリンクバス、故障回

避に使用される補償リンクバス、及びリンクバスを切り替える3モードスイッチが置かれる。1D-SRTの場合は横方向だけのシフトにより故障を回避したが、2D-SRTは縦横の両方向のシフトが考えられる。上下左右どちらの方向にシフトするかを決定するアルゴリズムは、WSI全体の歩留まりに大きく影響し、いくつかの方法が考えられるが、Kungらはグラフ理論を用いた方法[25]、Numataらはローカル情報のみを用いてヒューリスティックに再構成を行なう方法(HS法)[27]を提案している。

最初に図に示すような横(右)方向のシフトについて考える。横方向のリンクバスは1D-SRTと同様にシフト可能である。縦方向のリンクバスは、図6.2に示すような、3モードスイッチを用いてシフトされる。このスイッチは、上下と左右の結線をスルーにする状態と、上-右、左-下をつなぐ状態、及び上-左、右-下をつなぐ状態の、3つのモードをとることができる。縦方向のリンクバスのシフトを可能とするために、横方向に補償リンクバス(図中の破線)を置く。

右シフトは3ステップの動作により成し遂げられる。ステップ1では、故障PEの縦横方向のリンクバスに接続する全スイッチ束のスイッチ及びlevel-0スイッチをCrossモードにし、故障PEの切離しを行なう。ステップ2では、補償リンクバス及び3モードスイッチを用いて、縦方向のリンクバスの経路を曲げる。ステップ3では、レベルの再定義を行なう。1D-SRTの再構成と同様に、ノードのレベルが$l \rightarrow k$に再定義される場合、縦横両方向のリンクバスに接続するスイッチ束のレベルlのスイッチがConnectモードからCrossモードに変更され、代わってレベルkのスイッチがCrossモードからConnectモードにセットされる(図中破線円)。縦方向にシフトする場合も同様の手続きによりシフト可能である。これらの手続きにより、故障PEは上下左右どの方向にもシフトでき、故障の回避が可能である。

図6.3, 6.4に、2D-SRTの故障回避を行ない、正常PEだけを使用するようにWSI上での2D-SRTの結合を再構成した例を示す。図6.3は再構成前の故障及び各PEの使用/不使用の状態、図6.4は再構成後の結合状態である。

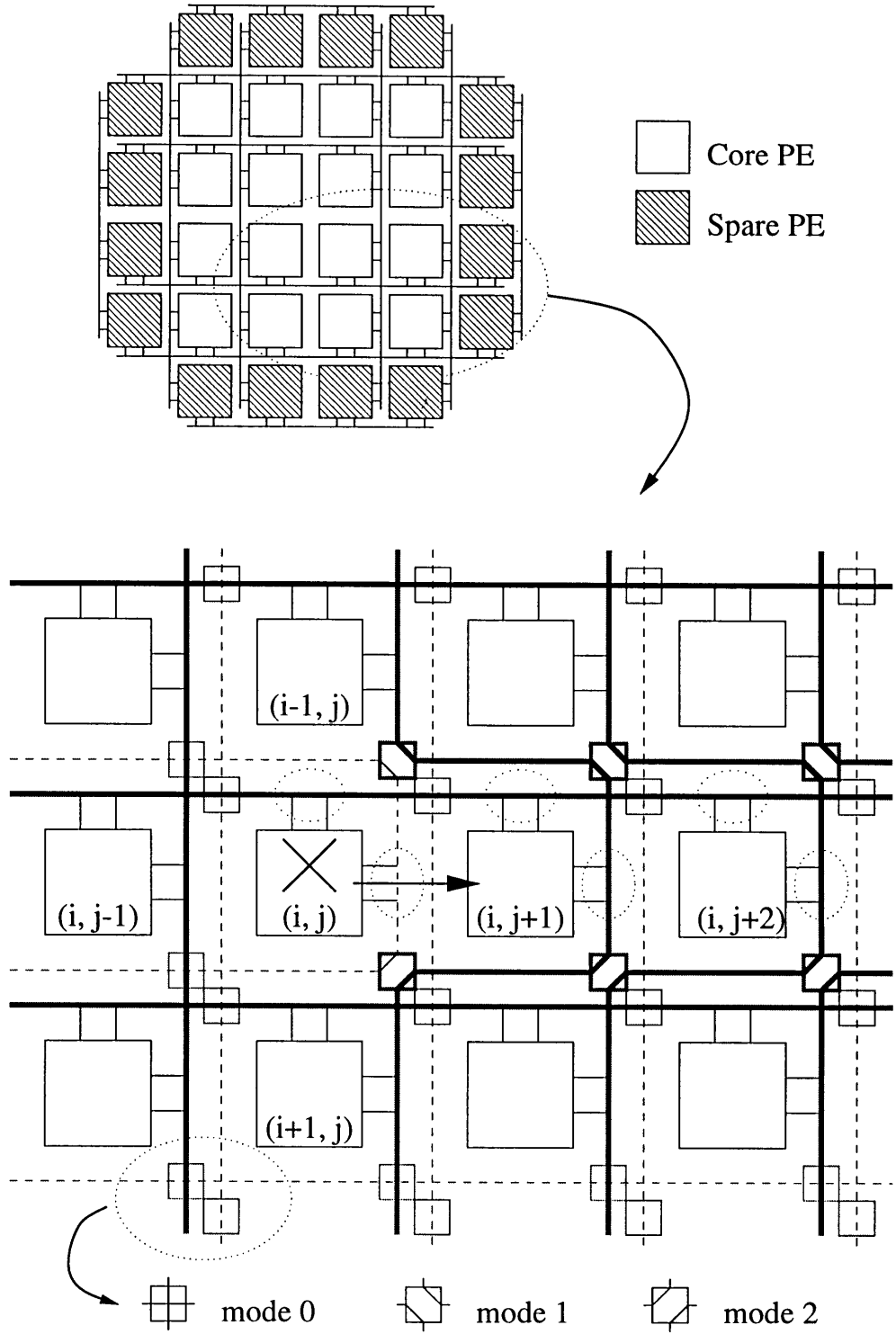
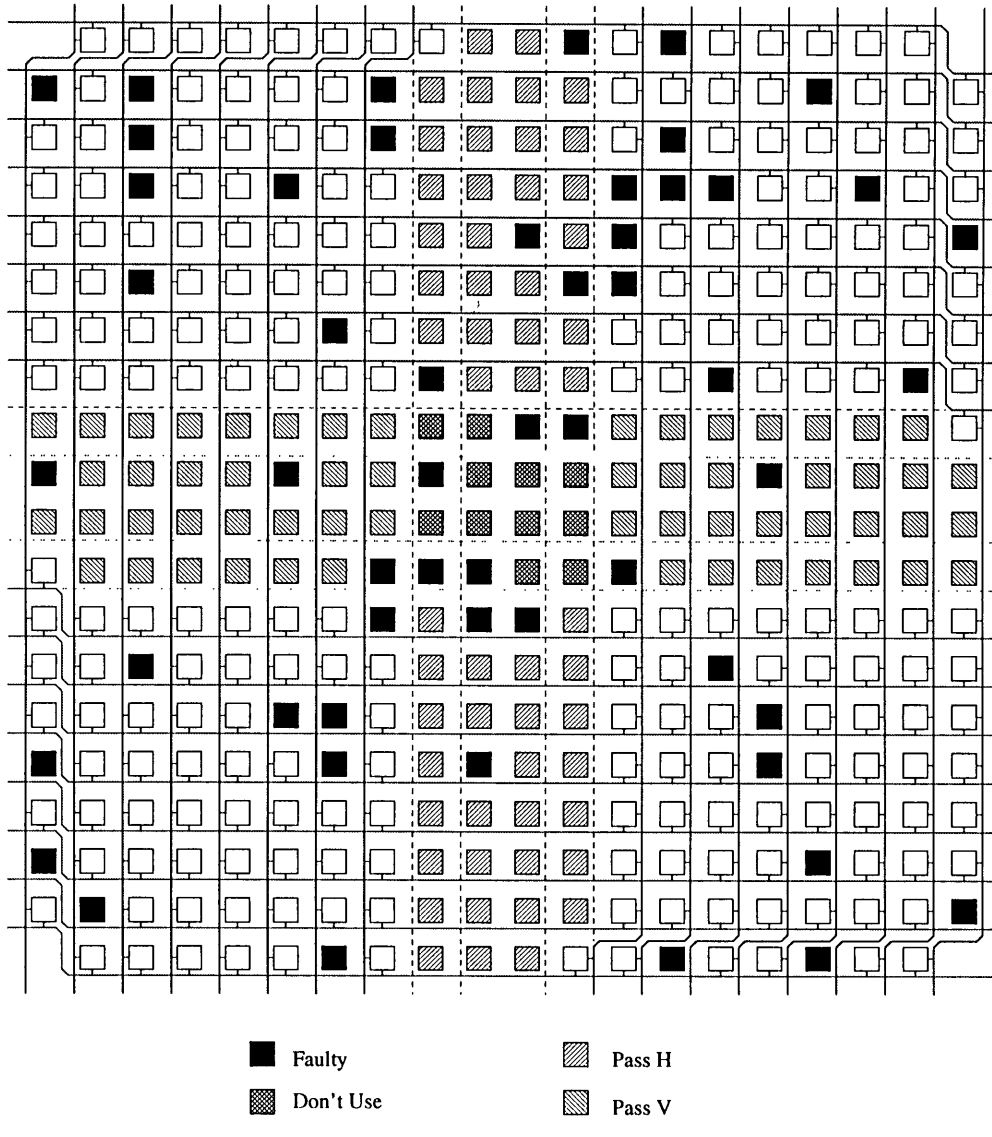
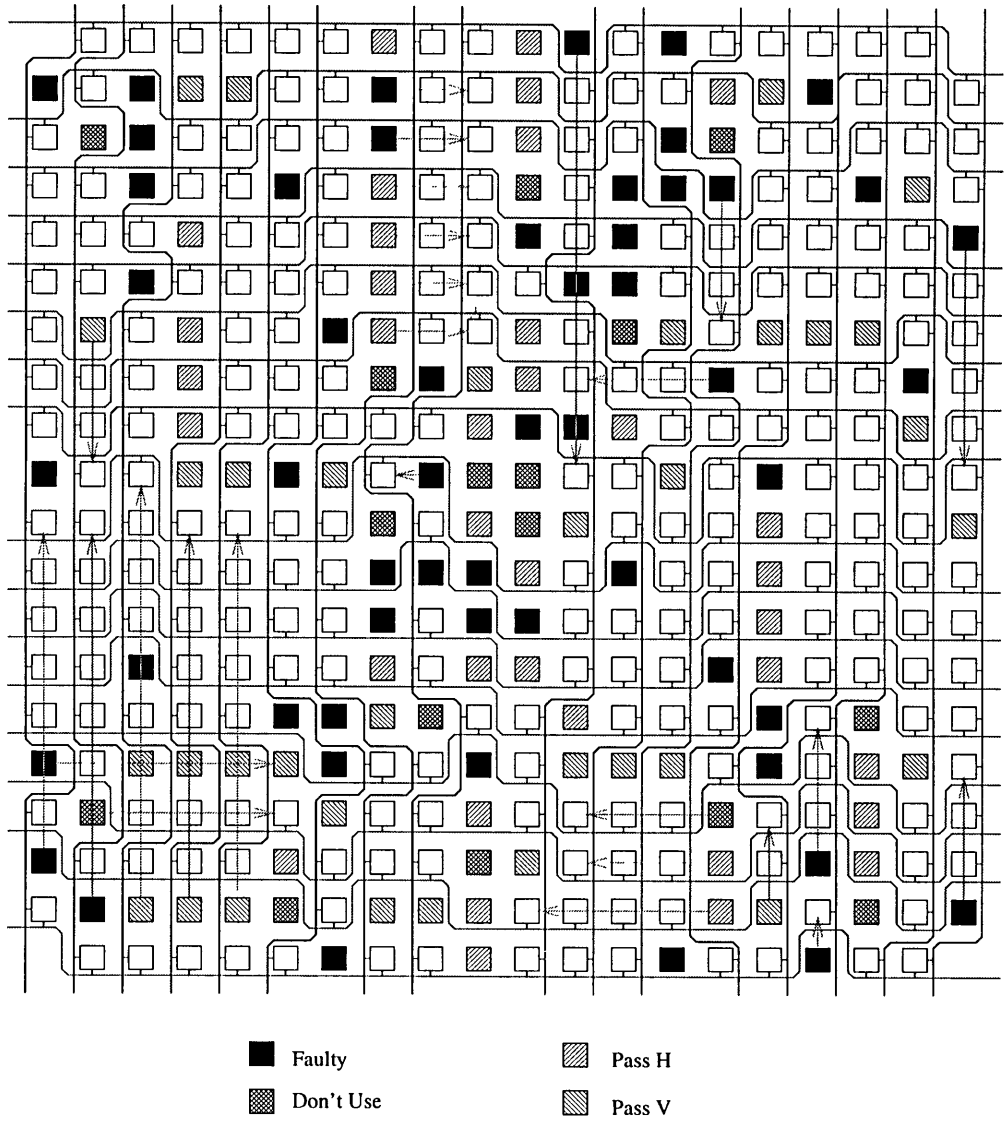


Figure 6.2: 再構成可能な 2D-SRT アーキテクチャ



Core PE on Edge, nodirect shift, B=0.0, yield=0.84

Figure 6.3: 再構成前の WSI の状態



Core PE on Edge, nodirect shift, B=0.0, yield=0.84

Figure 6.4: 再構成後の WSI の状態

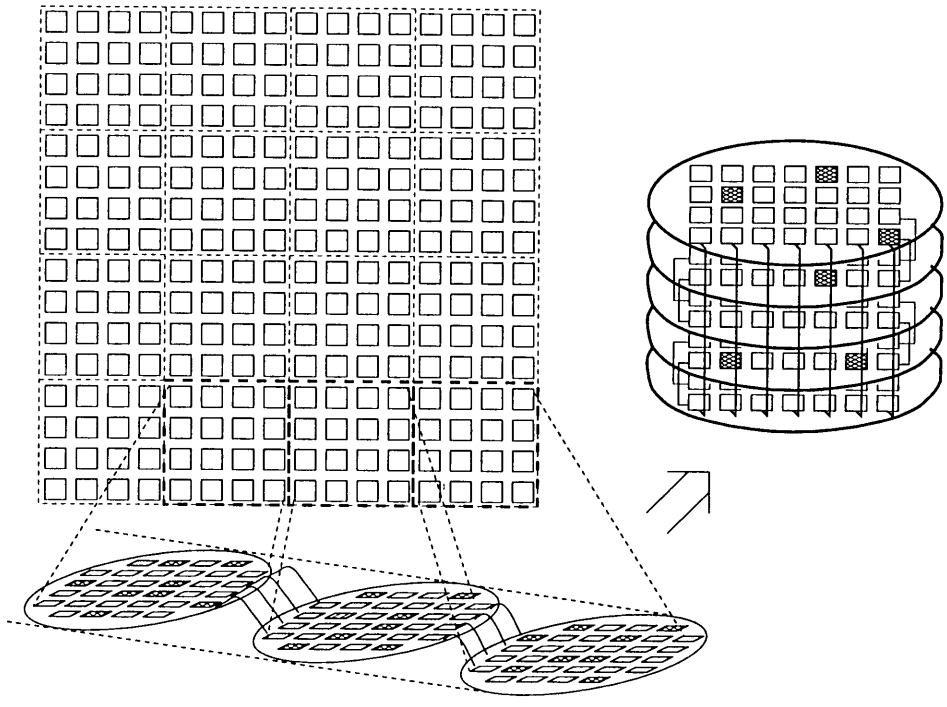


Figure 6.5: 2D-SRT のウェーハスタック実装

6.3 ウェーハスタック構造を用いた SRT 実装

6.3.1 ウェーハスタック構造

大規模な 2D-SRT を構成する場合、複数のウェーハ上に 2D-SRT の一部を実装し、このウェーハを層状に重ねることにより、WSI スタックによる実装が考えられる。この概念図を図 6.5に示す。ウェーハ上には実装すべき 2D-SRT のサブセットに加え、冗長な PE を配置しておく。ウェーハ上の故障 PE は、ウェーハ内部で再構成され、各ウェーハは、論理的に欠陥の無い 2D-SRT のサブセットを構成する。欠陥回避のためのシフトは、ウェーハ内で完結し、ウェーハ間にまたがるシフトは行わないものとする。

6.3.2 ウェーハスタック実装における発熱モデル

ウェーハ上の PE は、もし動作状態ならば発熱し、非動作状態の場合は発熱しない。放熱を WSI スタックの周囲から行なうとすると、ウェーハ内の発熱量が同じ場合、発熱部分がなるべく周囲に存在する方が、発熱部分が中心部に集中する場合よりも、ウェーハ内の最高温度を低くすることができる。

一方、故障回避が成功した場合、ウェーハ上には動作 PE、故障 PE、正常だが動作しない休止 PE ができる。故障率が十分低い場合、冗長 PE の殆どが休止 PE となる。故障 PE は

用いたウェーハにより決定されるが、休止 PE は動作 PE と交換可能であるため、休止 PE をウェーハ中心部に集めることにより、システムの最高温度を低下させることができる。

休止 PE を移動させることによる温度への寄与について、1 次元の場合について考察する。このモデルを図 6.6に示す。図中、(a), (c) は発熱部分、(b) は発熱せず熱伝導のみを行うとする。(a) の左端は、ウェーハ中心とし、反対側も同様に発熱するものと考え、断熱境界とする。

(a) の右端は、(a) で発熱した熱は全て系の右端から放熱されるので、熱流束が分る。従って (a) の境界条件は、

$$\begin{aligned} x &= 0 : (\dot{q})_{x=0} = -\lambda \left(\frac{dT}{dx}\right)_{x=0} = 0 \\ x &= h : (\dot{q})_{x=h} = h\dot{q}_v \end{aligned}$$

また内部発熱のある 1 次元の熱伝導方程式の一般解は、

$$T = -\frac{\dot{q}_v}{2\lambda}x^2 + C_1 x + C_2$$

である。

(b) は内部発熱が無いため、境界条件及び温度分布は以下の様になる。

$$\begin{aligned} (\dot{q})_{x=h} &= (\dot{q})_{x=h+l} = h\dot{q}_v \\ h\dot{q}_v &= \lambda \frac{T_{x=h} - T_{x=h+l}}{l} \end{aligned}$$

(c) の左端は、(a) の右端と同様に熱流束一定の条件、右端は温度一定 (外部温度) とする。

$$\begin{aligned} x &= h + l : (\dot{q})_{x=h} = h\dot{q}_v \\ x &= 1 : T_{x=1} = T_0 \end{aligned}$$

これらを解くと、(a), (b), (c) それぞれの温度分布は、

$$\begin{aligned} T_a &= -\frac{\dot{q}_v}{2\lambda}x^2 + \frac{\dot{q}_v}{\lambda}\left\{l(h + \frac{1}{2}l) + \frac{1}{2} - l\right\} + T_0 \\ T_b &= \frac{\dot{q}_v}{\lambda}hl + \frac{\dot{q}_v}{\lambda}\left\{\frac{1}{2}(l^2 - h^2) + \frac{1}{2} - l\right\} + T_0 \\ T_c &= \frac{\dot{q}_v}{\lambda}\left\{-\frac{1}{2}x^2 + lx + \frac{1}{2} - l\right\} + T_0 \end{aligned}$$

ウェーハ中心で温度が最高と仮定すると、ウェーハ中心温度 $T_{x=0}$ は、

$$T_{x=0} = \frac{\dot{q}_v}{\lambda}\left\{l(h + \frac{1}{2}l) + \frac{1}{2} - l\right\} + T_0$$

となり、非発熱領域 (b) を動かす (h を動かす) ことにより、線型的に変化することが分る。

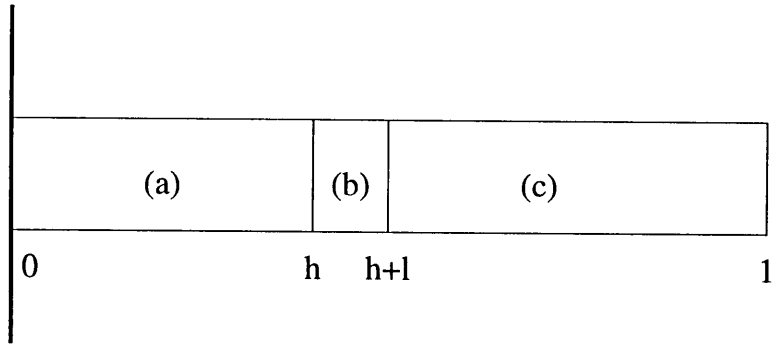


Figure 6.6: 1 次元の発熱モデル

6.3.3 放熱を考慮した故障回避アルゴリズム

冗長 PE の初期配置

再構成を開始する前の状態として、冗長 PE が周囲にある場合と、中心部にある場合が考えられる。周囲にある場合 (図 6.7) は、内部が均質であり、再構成後の歩留まりが高いことが期待できる。しかしながら、故障 PE が少なく、PE の配置が初期配置からあまり変化しない場合、周囲に休止 PE が多数存在し、内部温度は高くなる。

冗長 PE が中心にある場合 (図 6.8) は、外周近くの故障 PE は内部に向ってしかシフトできないため、ウェーハの歩留りは低下する傾向があるが、発熱 PE が周囲に配置されるため、内部温度は低くできる。

シフト方向の重み付け

HS 法の再構成アルゴリズムでは、シフトの方向について乱数を用いているが、方向については均質である。放熱を考慮した場合、なるべく外周に動作 PE が配置されるように、乱数に外に向って重み付けを行ない、シフトの方向を決定する。この様子を図 6.9に示す。

6.3.4 放熱を考慮した故障回避アルゴリズムの評価

歩留まりの評価

冗長 PE を外周に配置しシフト方向の重み付けが無い場合、冗長 PE を内部に配置しシフト方向の重み付けが無い場合、及び冗長 PE を内部に配置しシフト方向の重み付けを行なった場合について、ウェーハの歩留まりを評価した。

冗長 PE の配置による歩留まりを図 6.10に示す。冗長 PE を内部に配置しシフト方向の重み付けを行なった場合の歩留まりが低い原因は、冗長 PE が内部にあるにもかかわらず、シフトの方向は外向きに重み付けされているため、冗長 PE のある方向にシフトが行なわれにくいためと考えられる。いずれの方式も極端な歩留まりの低下は見られない。

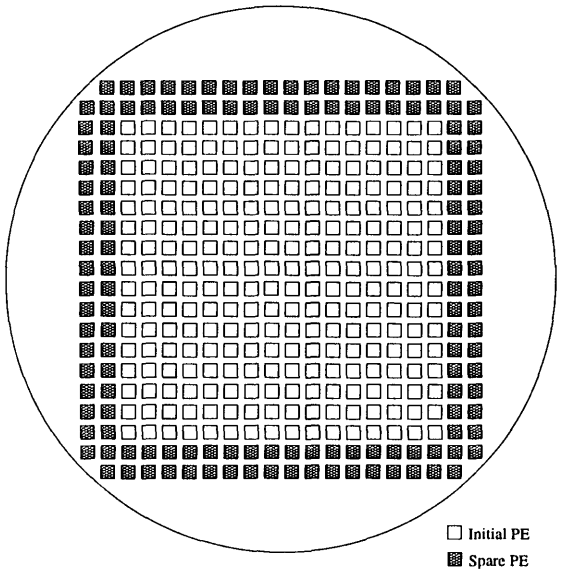


Figure 6.7: スペア PE を周囲に配置した WSI

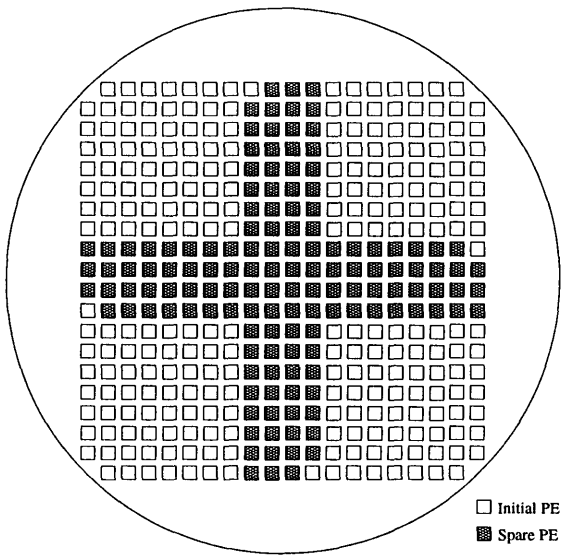


Figure 6.8: スペア PE を中心部に配置した WSI

温度評価

冗長 PE を外周に配置しシフト方向の重み付けが無い場合、冗長 PE を内部に配置しシフト方向の重み付けが無い場合、及び冗長 PE を内部に配置しシフト方向の重み付けを行なった場合について、ウェーハ内の最高温度をシミュレーションにより求めた。このグラフを図 6.11に示す。

シミュレーションの条件として、ウェーハの素材は Si とし、熱伝導率などの物理定数は Si と同じ値を用いた。PE は $(16+4) \times (16+4)$ で構成される。また、PE1 つ当りの面積を $25mm^2$、ウェーハの直径は $25cm$ とする。周囲温度は 25 ℃、1PE 当りの発熱量を 0.5W とした。

冗長 PE が外周に配置される場合、初期状態のとき放熱が最も悪い。故障 PE が増加すると、故障 PE を回避するために、内部の故障 PE が動作せず、外周部の冗長 PE が用いられるので、故障 PE が増加するに従い最高温度が低下する。

冗長 PE が内部に配置された場合、故障 PE が少ないとき放熱が最も良い。故障 PE が増加すると、内部に向って動作 PE が移動するため、最高温度は増加する。

故障 PE が 0 のとき、PE は初期配置のまま使用される。この時、外周に冗長 PE を配置した場合、137 ℃となり、ほぼ半導体の動作温度の限界となる。一方、冗長 PE が内部に配置される場合は、内部の最高温度が 88 ℃と、外周に冗長 PE を配置した場合に比べ、かなり最高温度を低くすることができる。この時の最高温度のグラフとウェーハ内の温度分布を図 6.11に示す。温度分布は、色別の図で直観的に観察できるが、各部の温度を詳しく観察できる等温線による温度分布を図 6.12に示す。

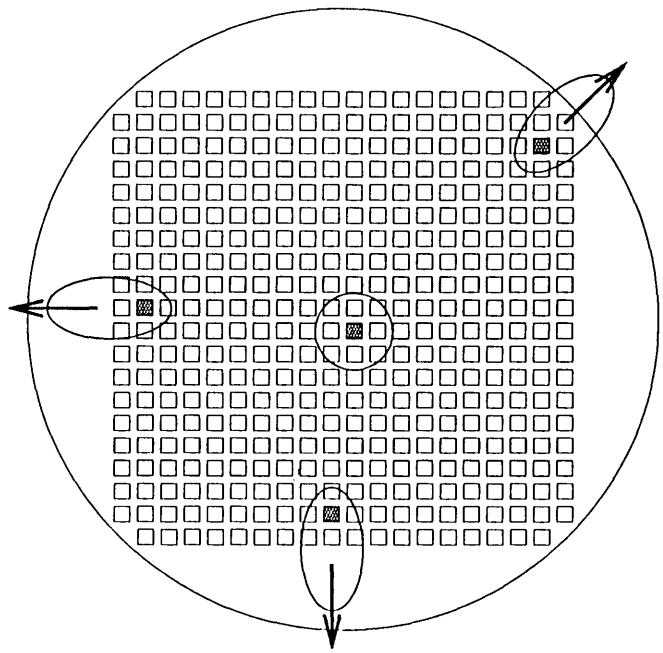


Figure 6.9: 方向の重み付けシフト

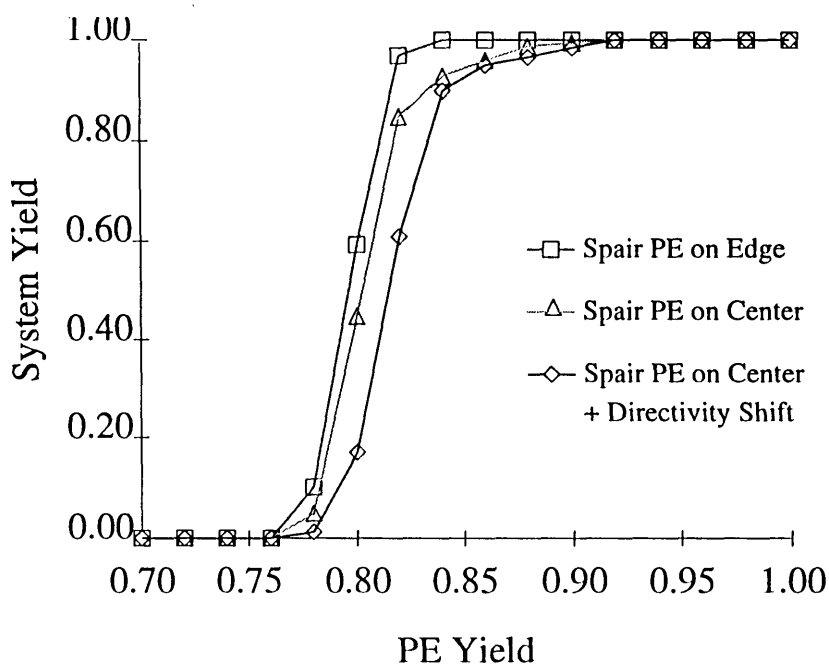


Figure 6.10: ウェーハスタック構造 SRT の歩留まり

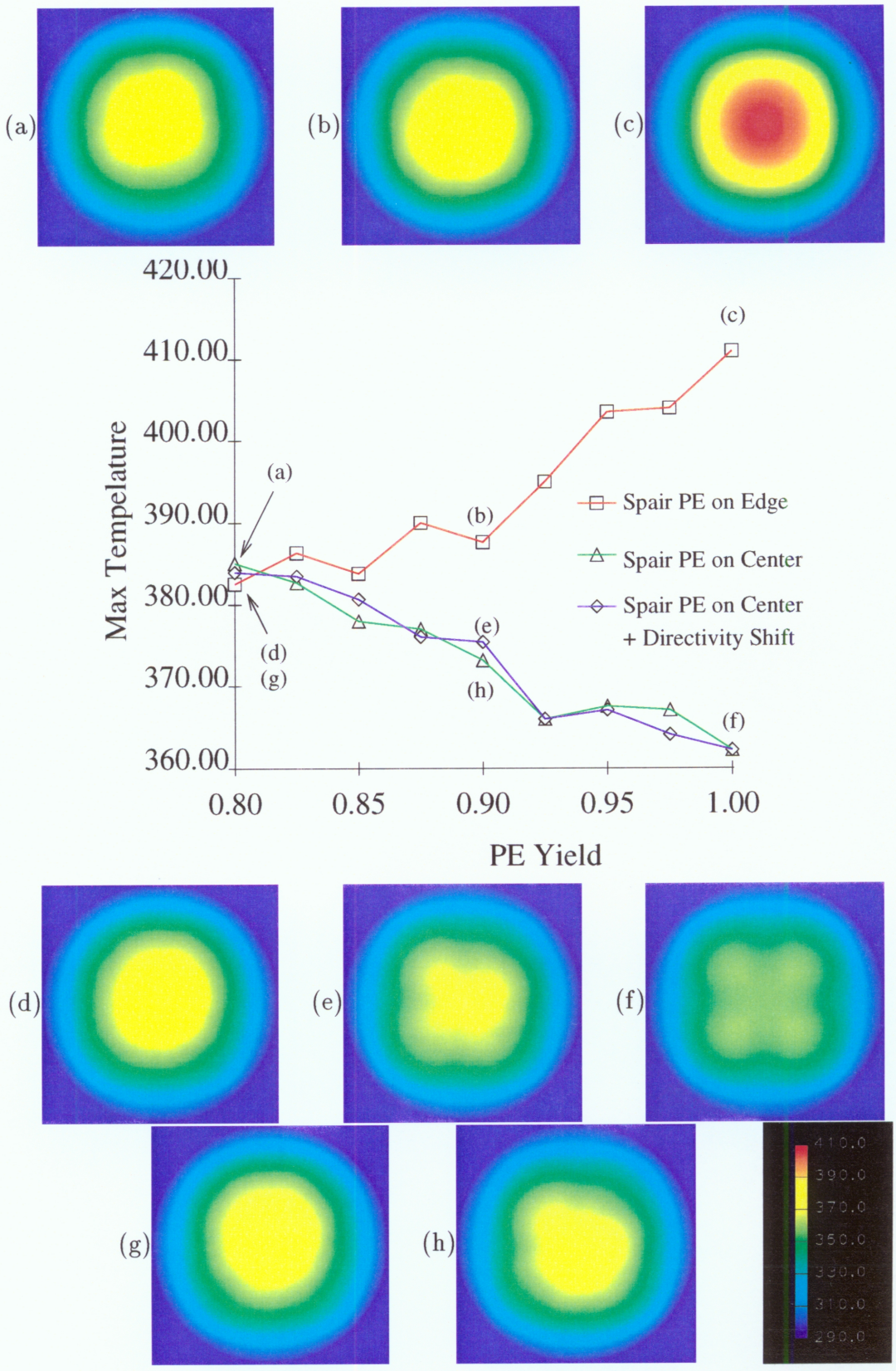


Figure 6.11: ウェーハスタック内の最高温度

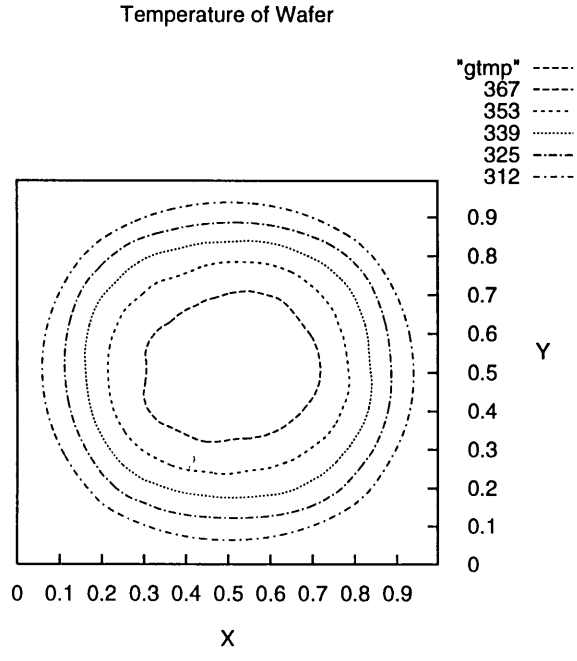


Figure 6.12: ウェーハスタック内の温度分布

シフトの方向に重みを持たせた方式では、PE の歩留りが低い場合には、他の方式に比べ最高温度を抑えることができる。しかしながら、その差は大きくはない。

6.4 まとめ

本章では、実装における諸問題として、SRT の故障回避アーキテクチャと温度分布について議論した。

SRT の故障回避アーキテクチャは、スイッチ束を用いた 1D-SRT の再構成アーキテクチャ、及び補償リンクバスを用いた 2D-SRT の再構成アーキテクチャを提案した。この故障回避アーキテクチャは、補償リンクバスやスイッチ束に、予備のリンクやスイッチを設けることによって、PE の故障だけでなく、リンクの故障も回避することができることを示した。

3 次元の実装方式として、SRT のウェーハスタックによる実装を提案した。3 次元の実装時に問題となる、内部の発熱をどのように放熱するかという問題に対し、従来の格子結合網の故障回避アルゴリズムを基本として、冗長 PE の配置法と、シフト方向の重み付けを行なった。この放熱を考慮した故障回避アルゴリズムについて、ウェーハ歩留まり及び

内部温度を評価した。その結果、歩留まりは、冗長 PE を外部に配置した場合が一番良いが、冗長 PE を内部に配置しても、歩留まりの低下は低いことが分った。また温度評価では、PE の歩留まりが高い場合は冗長 PE を内部に配置することにより、内部温度を大幅に低下できることが分った。PE の歩留まりが低い場合、シフト方向の重み付けにより、更に内部温度を低下させること可能であることが分った。

第 7 章

並列ラジオシティ法による室内照明シミュレーションと可視化

7.1 はじめに

3 次元 CG のレンダリング技法の一つであるラジオシティ法は、伝熱工学を応用した大域照明モデルによる画像生成手法であり、室内照明の高品位 CG を生成する際などによく用いられている方法である。ラジオシティ法では、光源からの直接光だけでなく物体間の相互拡散反射も考慮に入れて画像を計算するため、線光源・面光源が作る不均一な影や、間接照明が多い室内などの表現に適し、非常に現実感の高い画像を生成できるのが特徴である。

ラジオシティ法では、パッチ間で光が到達する割合を表す、フォームファクタと呼ばれる値の計算が重要となる。これは 2 重積分を解くことによって求められるが、パッチ間に障害物がある場合などに積分計算で求めるのは非現実的である。そのため、近似値を求める方法としてヘミキューブ法 [29] が提案された。また、Cohen 等 [30] により発表された漸進法は徐々に解に近づけていくため、途中の計算でも画像が得られる点、およびメモリを大幅に節約できる点が有利である。

一般にラジオシティ法では、フォームファクタの計算時間が全計算時間の大部分を占める。この処理には膨大な時間が必要であり、画像生成をリアルタイムに行うことが要求される現在では、その高速化が実用化に向けての大きな問題点になっている。したがってフォームファクタの並列計算による高速化が重要である。

従来の並列化手法として、ポリゴンやパッチ単位で分割し並列化を行なう方法が提案されている。さらにパッチを階層的に分割して精度を上げる方法も提案されているが、並列化が難しく研究されてこなかった。

本章では、動的負荷分散を考慮に入れたラジオシティ法の並列計算モデルを提案する。基本となるアルゴリズムは、フォームファクタの計算の際にヘミキューブを分割し、漸進法のラジオシティエネルギーを放射するパッチを複数にして、並列計算を行なう。

このアルゴリズムを超並列計算機 Parsytec GC-PowerPlus に実装し、室内照明シミュレー

ションを行い、並列アルゴリズムの有効性を示す。

7.2 ラジオシティ法

7.2.1 ラジオシティ方程式

光のエネルギーの放射と反射について、すべての放射エネルギーの放射と反射の過程は、理想的な拡散である (入射光がすべての方向に同じ強度で反射する) と仮定する。このとき、ある 1 枚の面から発する光 (ラジオシティ) は、自己放射光 (面自身が発光している) および他の面から入射して拡散反射する光で構成されている。1 つの面から発する光の量を決定するためには、すべての面相互間の幾何学的な関係および各面から発する光の量を記述することが必要となる。そこでまず、あるパッチ i とパッチ j のラジオシティに関する関係を式 (7.1) で表す。

$$B_i A_i = E_i A_i + \rho_i B_j F_{ji} A_j \tag{7.1}$$

ここで、

B_i: パッチ i から発するエネルギー (ラジオシティ) の総量 [エネルギー/単位時間/単位面積]
E_i: パッチ i から自己放射されるエネルギーの総量 [エネルギー/単位時間/単位面積]
A_i: パッチ i の面積
A_j: パッチ j の面積
ρ_i: パッチ i の反射率
F_{ji}: パッチ j から発したエネルギーがパッチ i に到達する割合 (フォームファクタ)

である。ここでフォームファクタとは、あるパッチから他のパッチへ光のエネルギーが到達する割合である。

7.2.2 フォームファクタ

フォームファクタ F_{ij} は、パッチ i の放射するエネルギーがパッチ j に到達する割合を示しており、パッチ相互の幾何学的関係によって決定される。このフォームファクタを求めると、各パッチのラジオシティが得られる。

パッチ i、j の面積がそれぞれ A_i、A_j であるとする。パッチ i に含まれる微小領域 dA_i からパッチ j に含まれる微小領域 dA_j へのフォームファクタ $F_{dA_i dA_j}$ は次式で与えられる。

$$F_{dA_i dA_j} = \frac{cos\phi_i \cdot cos\phi_j}{\pi r^2} \tag{7.2}$$

式(7.2)を領域A_jについて積分することにより、微小領域dA_iからパッチjへのフォームファクタ$F_{dA_i j}$は次式で表される。

$$F_{dA_i j} = \int_{A_j} \frac{cos\phi_i \cdot cos\phi_j}{\pi r^2} dA_j \tag{7.3}$$

さらに、式(7.3)を領域A_iについて積分することにより、パッチiからパッチjへのフォームファクタF_{ij}は領域A_iにおける平均として次式で表される。

$$F_{ij} = \frac{1}{A_i} \int_{A_j} \int_{A_j} \frac{cos\phi_i \cdot cos\phi_j}{\pi r^2} H(dA_i, dA_j) dA_j dA_i \tag{7.4}$$

ただし、$H(dA_i, dA_j)$は、

$$H(dA_i, dA_j) = \begin{cases} 1 & (dA_i と\ dA_j の間に遮蔽物体が存在しない場合) \\ 0 & (dA_i と\ dA_j の間に遮蔽物体が存在する場合) \end{cases} \tag{7.5}$$

で定義される。

フォームファクタは式(7.4)を用いて解析的に求める必要がある。しかし、積分を含むため非常に多くの計算時間を要し、実用的には計算しにくい。そこで、簡単にフォームファクタを計算する手法が提案された。

7.2.3 漸進法

Cohen[30]らは、パッチ数に比例する程度のメモリ量で、漸近的に画像を生成するラジオシティアルゴリズムを提案した。従来法では、パッチiのラジオシティは式(7.6)に示すように、パッチi自身の自己放射エネルギーと他のパッチjからパッチiが受けとるエネルギーの和である。

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \tag{7.6}$$

あるシーンで、パッチiが放射する光量を決めているのは、残りの環境から収集する光である(図7.1)。式(7.6)の$\sum$の中の項は、パッチiのラジオシティに影響を与えるパッチjの寄与の度合を示している。

$$B_j による\ B_i = \rho_i B_j F_{ij} \tag{7.7}$$

パッチjの最終的なラジオシティB_jは、他のすべてのパッチからの影響の合計である。影響の大きなパッチから順にその影響度を加算していけば、パッチjに対する他のすべてのパッチからの影響の合計は、より早く最終値に収束する。ここで、影響の大きなパッチとは、最も大きなエネルギーを放射する、すなわち最大の$B_i A_i$を持つパッチ群を指す。直観的に言って、最も強い光のエネルギーを放射するパッチは、普通、環境照明に最も強く影響する。このようなパッチは最初に扱うべきである。

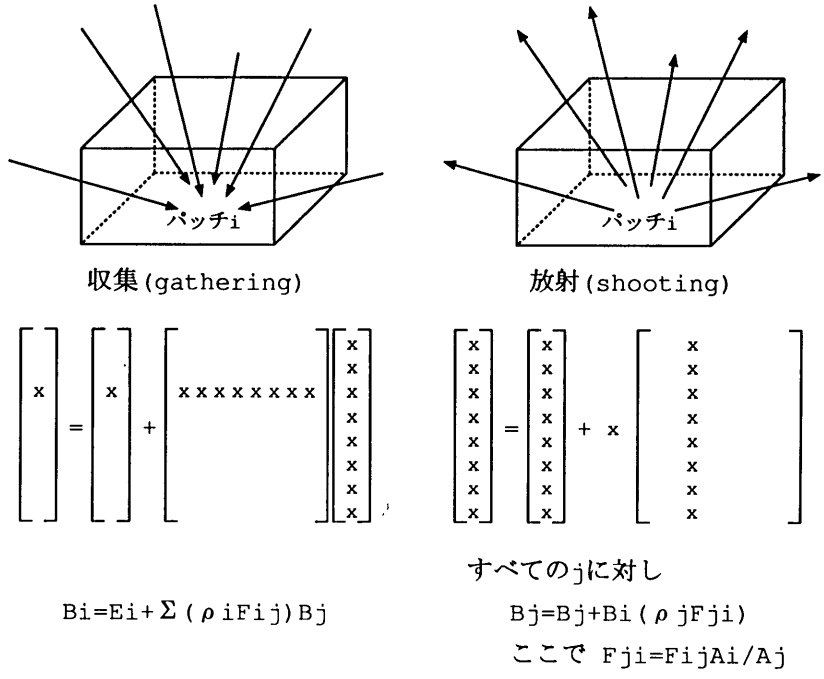


Figure 7.1: 光の収集と放射

そこで、$\Delta B_i A_i$が最大になるようなパッチは、常にエネルギーを放射するものとする。大部分の光源は、この規則に従い自動的に処理される。なぜなら、それ以外のすべてのパッチではラジオシティの初期値は 0 だからである。光源は多くのパッチにとって普通最も重要な照明源なので、光源を最初に処理した後では、多くの環境は既に十分よく明かりに照らされている。この規則に従って処理される次のパッチ群は、その光源から最も多くの光を受け取ったパッチとなる。そして、これが次々に繰り返される。

大きい順にパッチをソーティングし、この順序で処理を進めていくと、光源から光が環境へ伝搬していくのとほぼ同じ順序でラジオシティの計算が進む。一般に、こうしてパッチをソーティングすると、ラジオシティの収束を早めることができ、計算量を実質的に削減できる。

7.3 並列ラジオシティ法

7.3.1 ヘミキューブ固定分割による並列化アルゴリズム

本研究で用いた手法は、ヘミキューブを図 7.2 のようにメッシュ状に分割し、各々のセルを PE に割り当てて処理を分散させる方法である。また、漸進法において、複数のパッチから同時にエネルギーを放射することによって並列に処理することが特徴である。

この方法によって、フォームファクタの計算とエネルギーの放射の 2 つの段階で並列計

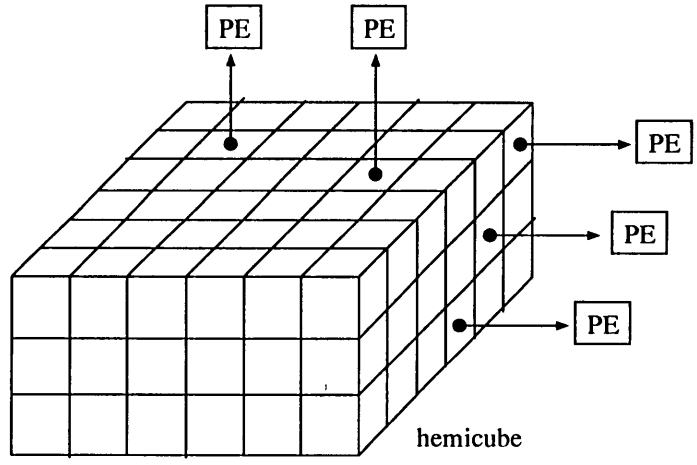


Figure 7.2: ヘミキューブの分割による並列化

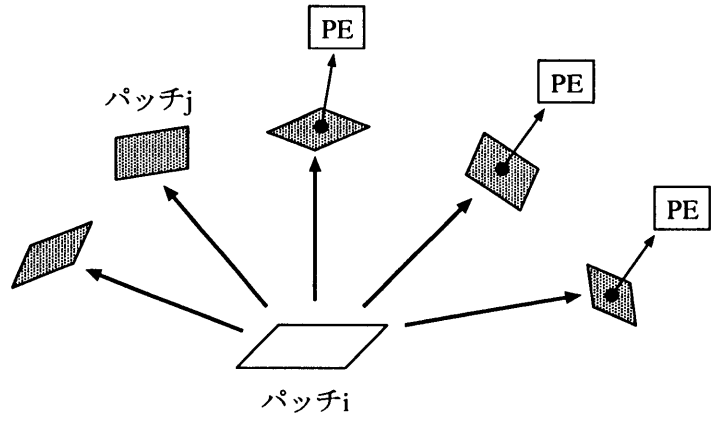


Figure 7.3: エネルギーの放出の並列化

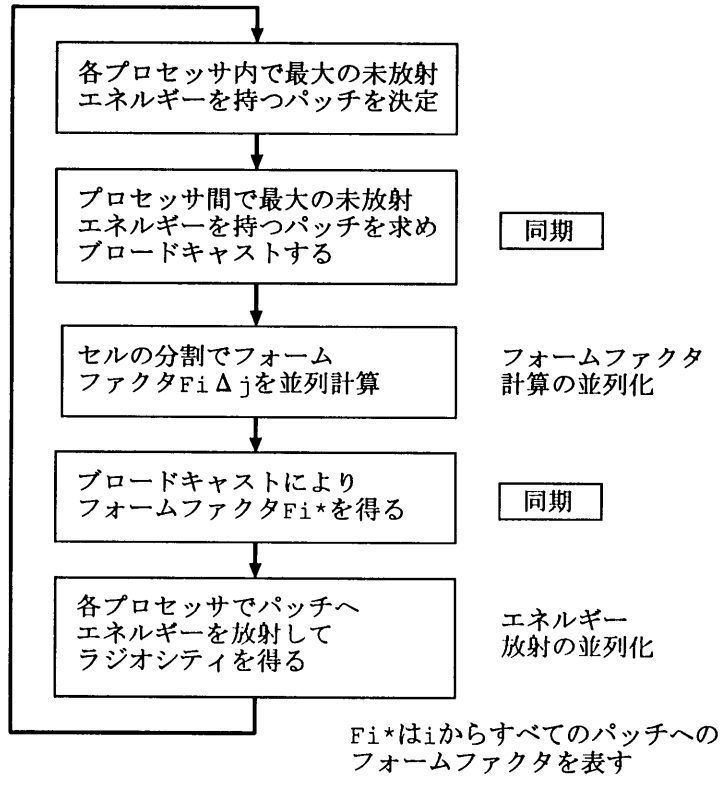


Figure 7.4: 固定分割法の流れ

算を行っている。本手法の大まかな流れを図 7.4 に示す。

7.3.2 超並列計算機 Parsytec GC への実装

並列ラジオシティアルゴリズムは図 7.5 に示した Parsytec GC に実装した。プログラミング言語は C++である。Parsytec GC では、並列処理のための API は C のライブラリ関数として用意されている。

環境データは図 7.6 のようなリンク構造を持つ構造体として、各 PE のメモリに置かれる。各 PE は 32MB のローカルメモリを持っているので、それぞれのローカルメモリに重複してデータを置く余裕がある。こうすることで PE 間の通信量を極力減らすようにした。入力データファイルは 1 つの環境定義ファイルと 1 つ以上のモジュール定義ファイルから成る。

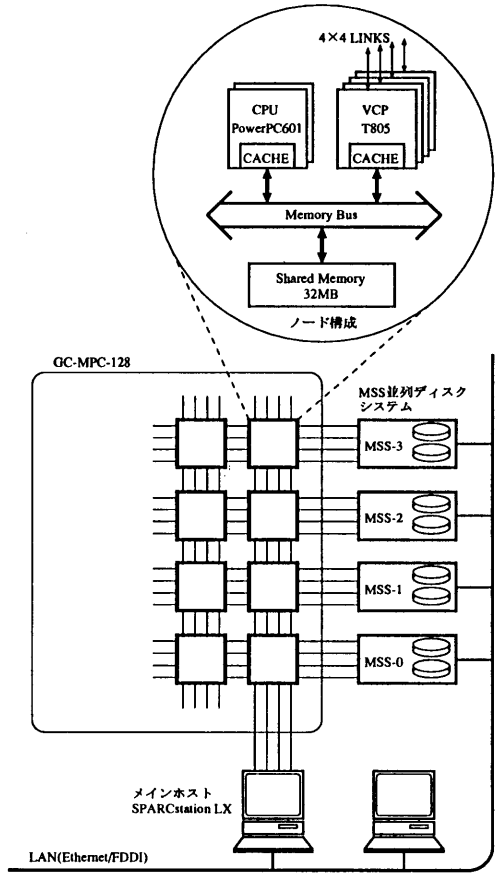


Figure 7.5: Parsytec GC-PowerPlus128/32 構成図

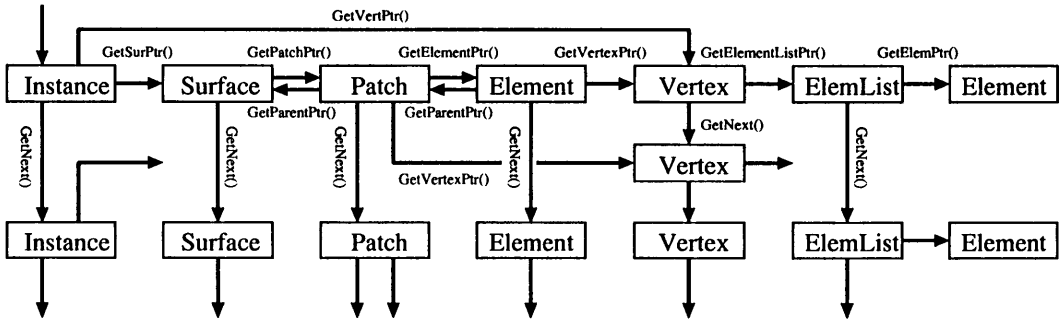


Figure 7.6: 環境のデータ構造

Table 7.1: 実験に用いた室内照明環境データ

	インスタンス数	面数	パッチ数	エレメント数	頂点数
(a) ROOM	9	22	48	197	326
(b) PICROOM	10	27	176	205	343
(c) CORNELL	8	9	112	166	274

7.4 室内照明シミュレーション

7.4.1 並列化性能

実験には3種類の室内照明シミュレーションデータを用いた。データファイルの各種パラメータは表7.1 のとおりである。この3種類のデータから生成された画像を、それぞれ図7.7 、図7.8 、図7.9 に示す。

7.4.2 ブロードキャストの高速化

ノード数が増加するにしたがって処理時間の大部分を通信時間が占めている。これを改善するにはブロードキャストを高速化する方法が考えられる。ブロードキャストそのものを高速化することにより、全処理時間に占める通信時間の割合を小さくし、高速化を図る。

ここまで述べてきた手法の実装では、次のようなアルゴリズムでブロードキャストを行っていた。

表7.2 のように、ループを繰り返すことによりエネルギーが次第に放射されていく。それにしたがって徐々にリアルな画像が得られることがわかる。これらの画像が漸進法によって変化していく過程を図7.10 、図7.11 、図7.12 に示す。

また、放射エネルギー99.9%に要した処理時間を表7.3 、図7.13 に示す。ノード数8までは、並列計算により室内照明シミュレーションを高速化できるが、16以上になるとブロードキャストに必要な計算時間が多くなり、プロセッサ数に比例した高速化が得られない。

すべてのノードからデータを収集する必要がある場合、ある1つのノード (root ノード) に対して残りのノードが順にメッセージを送信する。root ノードは同様に残りのすべてのノードに順にメッセージを送信する。このアルゴリズムを図7.14 に示す。

この方式ではデータの送受信回数は $O(N)$ のオーダーになる。それに対し、次のようなアルゴリズムでブロードキャストを高速化した。

ツリー構造を考え、末端の葉から順にデータを送信する。そうして最後には root ノードにすべてのデータが到達する。root ノードは今の逆の要領でデータを送り、末端の葉にデータが届くまで繰り返す。このアルゴリズムを図7.15 に示す。この方式ではデータの送受信回数は $O(logN)$ のオーダーになる。

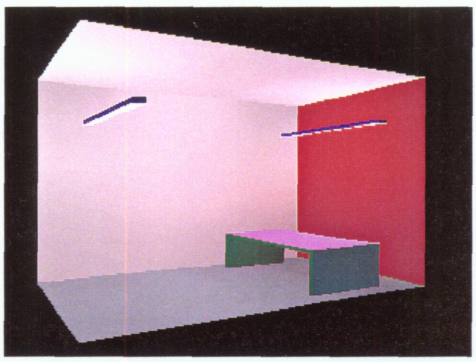


Figure 7.7: 実験データ画像 (ROOM)

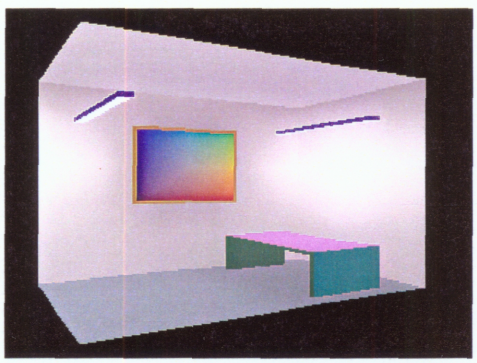


Figure 7.8: 実験データ画像 (PICROOM)

Figure 7.9: 実験データ画像 (CORNELL)

Table 7.2: 放射エネルギーと漸進法のループ回数

放射エネルギー (%)	10	20	30	40	50	60	70	80	90	95	99	99.5	99.9
(a) ROOM	4	7	10	12	16	19	25	33	41	44	54	59	73
(b) PICROOM	3	5	7	10	13	30	53	81	123	166	274	307	394
(c) CORNELL	-	-	-	-	1	4	13	30	71	104	183	216	302

Table 7.3: ノード数と計算時間

ノード数	1	2	4	8	16	32	64
(a) ROOM(sec)	320	268	221	189	257	384	633
(b) PICROOM(sec)	398	280	232	203	296	429	721
(c) CORNELL(sec)	289	237	186	144	209	336	582

この 2 種類の方法を使って、1 度のブロードキャストに要する時間を計測した。ツリー構造を用いたブロードキャストの結果を表 7.5 と図 7.16 に示す。

ノード数が多い場合、ツリー構造を用いたブロードキャストは、逐次的通信によるブロードキャストに比べて 2 倍以上高速にすることができた。

このブロードキャストルーチンを用いて、ラジオシティ(放射エネルギー 99.9%) を求めるのに要した処理時間を表 7.6 に示す。ブロードキャストアルゴリズムの変更後ではかなりパフォーマンスが改善されていることがわかる。

Table 7.4: 逐次的通信によるブロードキャストに要する時間

Nodes	Connect Only	Connect and Transfer		
		1KB	4KB	16KB
2	1.58ms	2.01ms	2.81ms	6.54ms
4	4.79ms	5.61ms	8.41ms	19.6ms
8	21.8ms	23.0ms	27.7ms	48.8ms
16	40.4ms	47.0ms	57.4ms	111ms
32	111ms	137ms	143ms	267ms
64	282ms	343ms	380ms	586ms

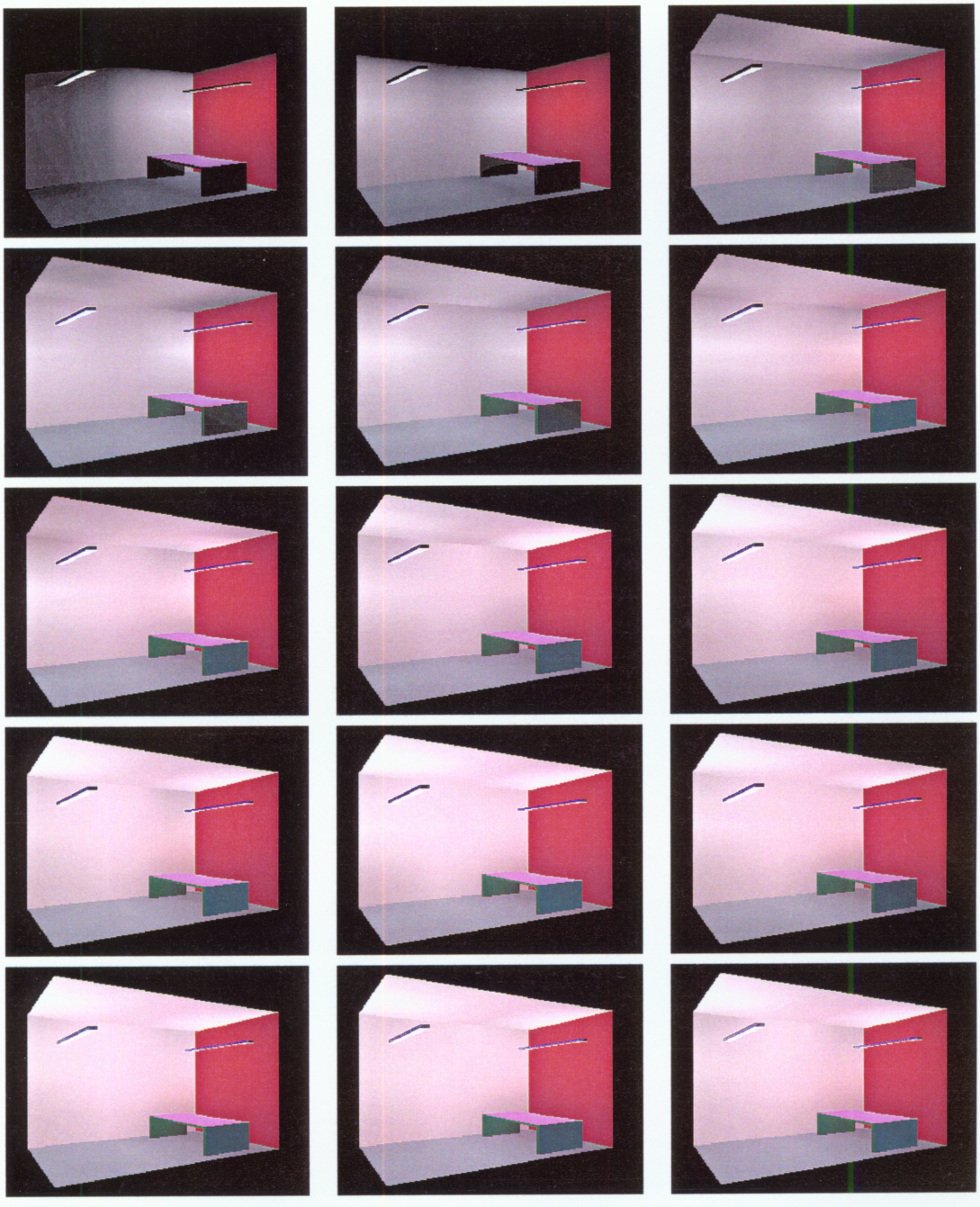


Figure 7.10: 漸進法による変化 (ROOM)

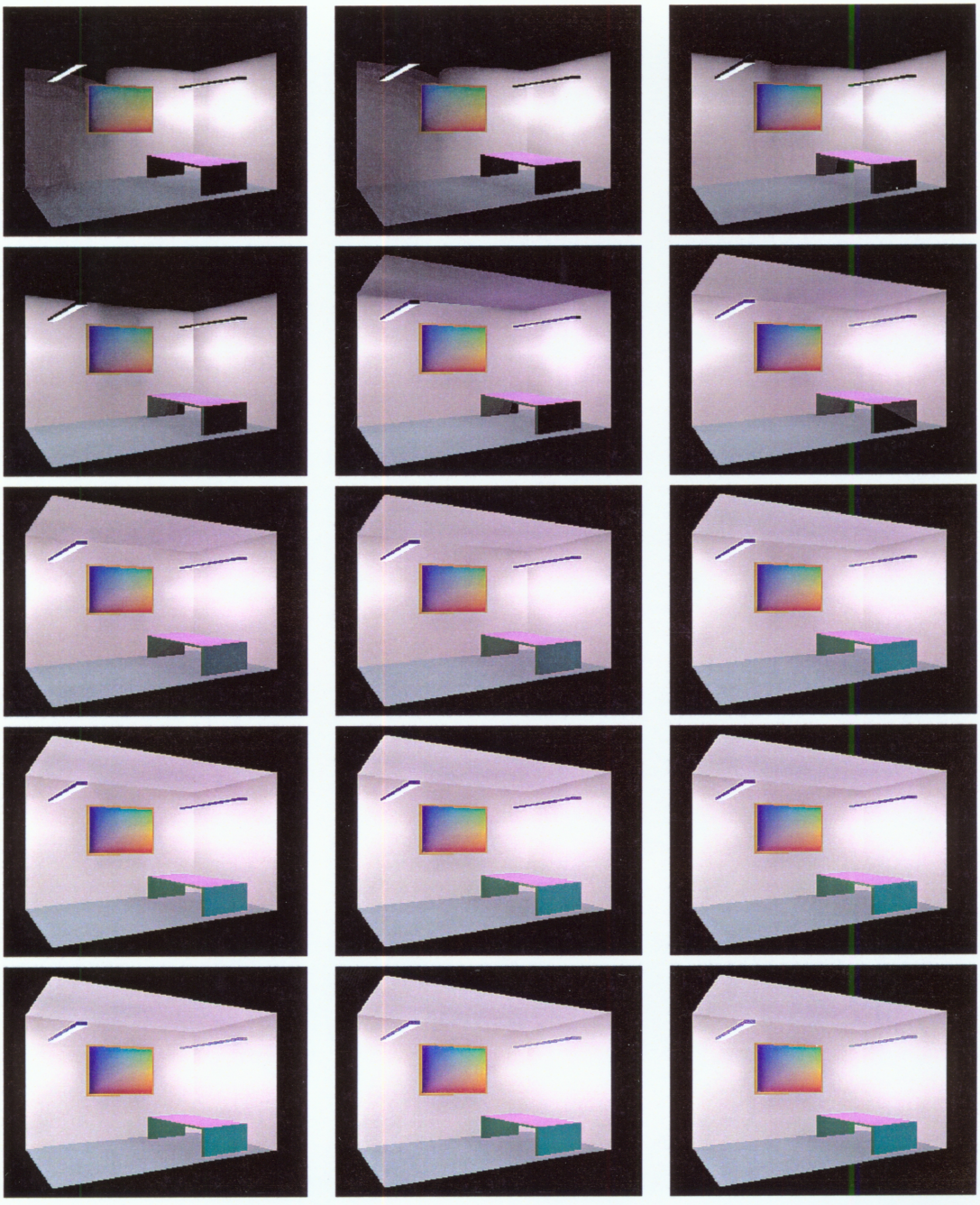


Figure 7.11: 漸進法による変化 (PICROOM)

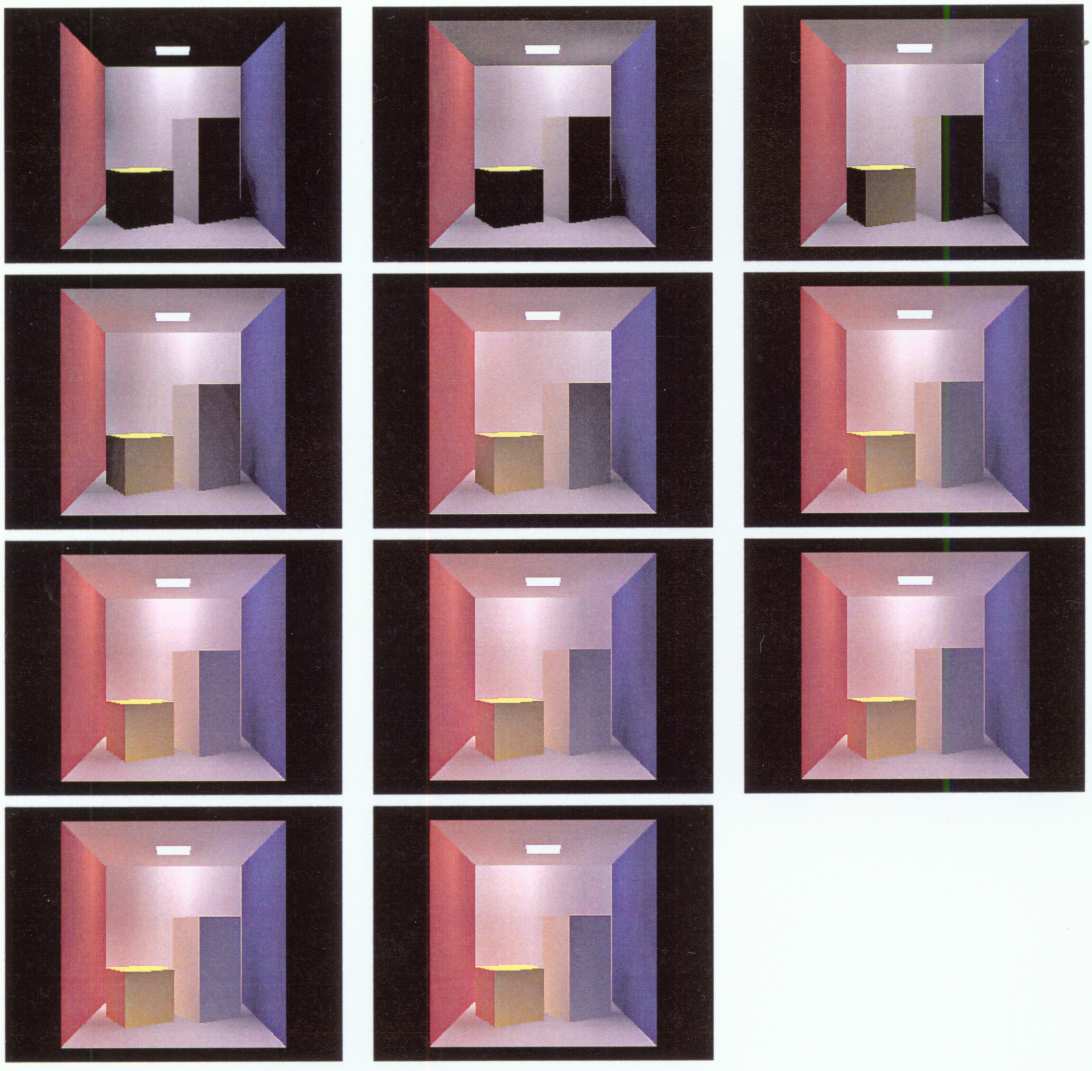


Figure 7.12: 漸進法による変化 (CORNELL)

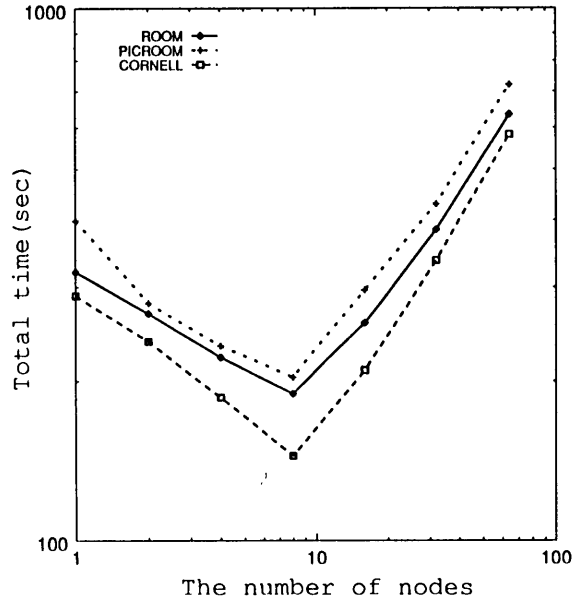


Figure 7.13: ノード数と計算時間

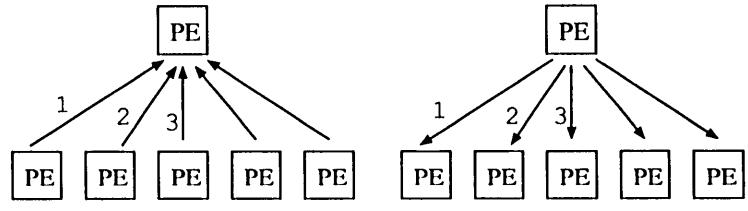


Figure 7.14: 逐次的通信によるブロードキャスト

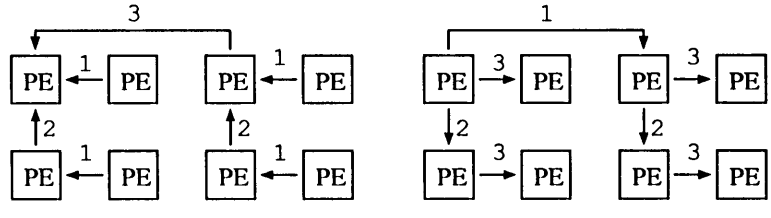


Figure 7.15: ツリー構造を用いたブロードキャスト

Table 7.5: ツリー構造を用いたブロードキャストに要する時間

Nodes	Connect Only	Connect and Transfer		
		1KB	4KB	16KB
2	1.59ms	1.88ms	2.82ms	6.54ms
4	3.89ms	4.65ms	5.93ms	14.6ms
8	19.0ms	19.9ms	22.8ms	33.8ms
16	34.9ms	36.6ms	40.4ms	54.7ms
32	75.3ms	74.7ms	78.1ms	79.1ms
64	103ms	105ms	104ms	137ms

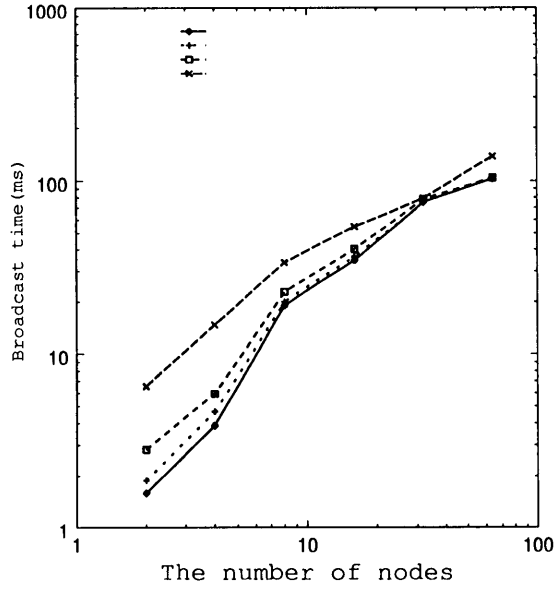


Figure 7.16: ツリー構造を用いたブロードキャストに要する時間

Table 7.6: ノード数と計算時間

ノード数	1	2	4	8	16	32	64
(a) ROOM(sec)	320	267	215	178	227	329	545
(b) PICROOM(sec)	398	279	224	191	264	362	594
(c) CORNELL(sec)	289	236	180	122	177	289	490

7.5 まとめ

従来行われてきた並列化手法を紹介した後、ヘミキューブの分割によるフォームファクタの並列化手法を提案した。さらにこの手法を Parsytec GC-PowerPlus に実装してアルゴリズムの有効性を検討した。

プロセッサ数が多くなると、通信に要する時間が非常に大きくなり、結果として逐次処理よりも速度が劣る結果となった。しかし、ブロードキャストをツリー構造を用いたアルゴリズムに改良することによって、わずかながら処理時間の短縮を図ることができた。

データをブロードキャストするのに、毎回コネクションを張り、データを送り、コネクションを切るという処理を繰り返しているが、コネクションを張るのにもかなりの時間がかかっている。

第 8 章

まとめ

コンピュータシミュレーションは、様々な科学技術分野で用いられ、実用規模の数値模擬実験には巨大なメモリ空間と膨大な計算時間が必要とされている。現在、この分野ではスーパーコンピュータに代わって多数の高速プロセッサからなる超並列コンピュータが注目され、最先端分野の超並列シミュレーション法およびデータの可視化に関する研究が切に望まれている。

本研究では、基礎物理・化学・物性材料設計・流体問題や神経回路学習などの最先端分野のにおけるコンピュータ・シミュレーションを疎結合型超並列コンピュータで実行する超並列シミュレーションとその可視化について物理、材料、計算工学、計算機科学、ソフトウェア科学などの分野から詳しく検討してきた。その結果、従来のコンピュータにあったシミュレーション対象の量的、質的な制限を大幅に緩めることが出来ることが明かになった。例えば、物理・化学分野での分子の振舞いをシミュレーションする分子動力学法では、分子数が数千個に限られていた物を数万個に容易に拡張できる。また、シミュレーションデータの可視化により気体から液体への相移転などの分子の振舞いを確認できた。

超並列シミュレーションの高速化については、プロセッサ間ネットワーク、メッセージパッシング、データ配置を考慮した動的負荷分散並列シミュレーション・アルゴリズムの提案を行ないその有用性を確認した。この分野以外では、流体シミュレーション、3 次元ウェーハスタック構造超並列コンピュータの発熱シミュレーション、脳における視覚野神経細胞の学習時の活性化シミュレーション、自己組織化の超並列シミュレーションやラジオシティ法を用いたコンピュータグラフィックスによる室内照明シミュレーションを行ないその高速性と有効性を明らかにした。

超並列シミュレーションからの膨大なシミュレーションデータは、3 次元コンピュータ・グラフィックスを用いて可視化を行ない、複雑なデータのカラー可視化や 3 次元可視化手法の有効性を示した。

参考文献

[1] D. W. Heermann, (小澤哲, 篠嶋妥 訳). "シミュレーション物理学". シュプリンガー・フェアラーク東京, 1990.

[2] D. M. Beazley and P. S. Lomdahl. "Message-passing multi-cell molecular dynamics on the Connection Machine 5". *Parallel Computing*, Vol. 20, pp. 173–195, 1994.

[3] P. S. Lomdahl, P. Tamayo, N. Gronbech-Jensen, and D.M. Beazley. "50 GFlops Molecular Dynamics on the Connection Machine 5". In *Proceeding of Supercomputing'93 IEEE*, pp. 520–527, 1993.

[4] "AVS ハンドブック Vol 1, Vol 2". 株式会社クボタ, 1993.

[5] *"AVS User's Guide Release 4"*. Advanced Visual Systems Inc., May 1992.

[6] *"AVS Module Reference Release 5"*. Advanced Visual Systems Inc., Feb. 1993.

[7] Thinking Machines Corporation. *"CM5"*, 1992.

[8] 保原充, 大宮司久昭（編）. "数値流体力学", pp. 3–13, 15–48. 東京大学出版会, 1992.

[9] 日本機械学会（編）. "流れの数値シミュレーション", pp. 56–93. コロナ社, 1988.

[10] Thinking Machines Corporation. *"CM Fortran Reference Manual"*, 1992.

[11] Thinking Machines Corporation. *"Prism Reference Manual"*, 1992.

[12] D. H. Hubel and T. N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex". *J.Physiol.,Lond.*, Vol. 195, pp. 215–243, 1968.

[13] K. Tanaka. "Neural Mechanisms of Object Recognition". *Science*, Vol. 262, pp. 685–688, 1993.

[14] I. Fujita, K. Tanaka, M. Ito, and K. Cheng. "Columns for visual features of objects in monkey inferotemporal cortex". *Nature*, Vol. 360, , 1992.

[15] 加藤聡, 堀口進. "下部側頭葉皮質IT野における神経細胞結合モデル". *JAIST Research Report*, Nov. 1996.

[16] T. Kohonen. "Self-Organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics*, Vol. 43, pp. 59–69, 1982.

[17] 乾敏郎. "視覚情報処理の基礎", p. 111. サイエンス社, 1990.

[18] K. G. Götz. "Do "d-blob" and "l-blob" hypercolumns tessellate the monkey visual cortex ?". *Biological Cybernetics*, Vol. 56, pp. 107–109, 1987.

[19] 斎藤秀昭. "知覚・認知と脳の細胞活動はどう対応するか". 実験医学, Vol. 12, 19（増刊）, pp. 167–174, 1994.

[20] G. Blasdel and G. Salama. "Voltage-sensitive dyes reveal a modular organization in monkey striate cortex". *Nature*, Vol. 321, pp. 579–585, 1986.

[21] T. コホネン. "自己組織化マップ". シュプリンガー・フェアラーク東京, 1996.

[22] H. Ritter K. Obermayer and K.Schulten. "Large-scale simulations of self-organizing neural networks on parallel computers:application to biological modelling". *Parallel Computing*, Vol. 14, pp. 381–404, 1990.

[23] G. Myklebust and J. G.Solheim. "Parallel Self-organizing Maps for actual applications ". In *Proc. Int. Conf. Neural Networks 1995*, 1995.

[24] C.-H. Wu and R. E. Hodges. "Parallelizing the Self-Organizing Feature Map on multiprocessor systems". *Parallel Computing*, Vol. 17, pp. 821–832, 1991.

[25] S. Y. Kung, S. N. Jean, and C. W. Chan. "Fault-Tolerant Array Processors Using Single-Track Switches". *IEEE Trans. on Computers*, Vol. 38, No. 4, Apl. 1989.

[26] 福田大, 堀口進. "巡回型ハイパーキューブ結合の WSI 構成法". 信学論, Vol. J76-D-I, No. 2, pp. 88–98, 1993.

[27] 沼田一成, 堀口進. "格子結合型マルチプロセッサシステムの WSI 構成法". 信学論, Vol. J77-D-I, No. 2, pp. 121–129, 1994.

[28] Michael J. Little and Jan Grinberg. "The 3-D Computer: An Intergrated Stack of WSI Wafers". *Wafer Scale Integration*, pp. 253–318, 1989.

[29] Michael F. Cohen and Donald P. Greenberg. "The hemi-cube: A radiosity solution for complex environments". In *Computer Graphics(SIGGRAPH '85)*, Vol. 19, No. 3, pp. 31–40, Aug. 1985.

[30] Michael F. Cohen, Shenchang E. Chen, John R. Wallace, and Donald P. Greenberg. "A progressive refinement approach to fast radiosity image generation". In *Computer Graphics(SIGGRAPH '88)*, Vol. 22, No. 4, pp. 75–84, Aug. 1988.

謝辞

本研究は、平成7年度～平成8年度に文部省より交付された科学研究費補助金基盤研究 (B) ならびに並列・分散処理研究推進機構の研究助成を用いて行なわれた。関係各位に深謝いたします。

本研究での超並列並列計算機でのシミュレーション技術やシミュレーションデータのビジュアル化方式が、今後ますます発展する最先端科学技術分野の大規模シミュレーションに大いに役立つことを願っている。この研究報告書が、これら関係分野の研究者各位のお役に立てば幸いである。