グラフィクスとCAD 68－2
（１９９４．５．２０）

並列グラフィクスアルゴリズムのサーベイ

安部 毅　大野 義夫　西田 友是　近藤 邦雄　中嶋 正之
慶應義塾大学　福山大学　埼玉大学　東京工業大学

マルチプロセッサ方式の並列計算機を用いたレンダリングアルゴリズム, Z バッファ法, スキャンコンバージョン, スキャンライン法, レイトレーシング法, ラジオシティ法の並列化手法について紹介する. 並列計算機を用いる際に考えなくてはいけないのは, すべてのプロセッサが効率良く処理をするように, 負荷を分散させることである. このサーベイでは各アルゴリズムの並列化手法をまとめるとともに負荷の分散, 特にレイトレーシング法の負荷分散手法について, 従来のアプローチをまとめてみる.

A Survey of Parallel 3D Rendering Algorithms

Tsuyoshi ABE　Yoshio OHNO
Keio University

Tomoyuki NISHITA
Fukuyama University

Kunio KONDO
Saitama University

Masayuki NAKAJIMA
Tokyo Institute of Technology

A survey on parallel 3D rendering algorithms, z-buffer, scan-conversion, scan-line, ray-tracing and radiosity method on multiprocessor computer is here. We need to take account of load balance so that each processor works efficiently. In this survey, we discuss various parallelization methods for 3D rendering algorithms, and also discuss load balancing scheme which are especially important for ray-tracing.

1 はじめに

近年, コンピュータグラフィクス (CG) は映画, コマーシャル, プレゼンテーションなど, さまざまな分野で使用されており, その応用範囲は多岐にわたっている. しかし, これら CG の製作には多くの時間とコストがかかるのが問題である.

CG の製作は大きく分けて, モデリングとレンダリングに分けられるが, 多大な演算処理能力を必要とされるのがレンダリング処理である. レンダリング処理も求められる画像のクオリティによって, 使用される手法はさまざまであり, クオリティの高い画像を求めればそれだけ演算量は多くなる. また, スーパーコンピュータのような計算処理能力の高い計算機を用いれば, コストが高くなる.

これらの問題を解決する一手法として, 並列計算機を用いることが考えられてきた. 本稿では, 並列計算機を用いたコンピュータグラフィクスのレンダリングアルゴリズムとして, Z バッファ法, スキャンコンバージョン, スキャンライン法, レイトレーシング法, ラジオシティ法の各アルゴリズムに対する並列化手法について紹介する.

2 並列計算機のアーキテクチャ

並列計算機上でプログラミングをする際には, アルゴリズムに着目しそのアルゴリズムの中から並列性を抽出しなくてはならない. しかし, 並列性の出し方として, 誰が並列性を構築するのかという問題がある. 現在, 逐次処理プログラムから並列性を抽出し自動的に並列化するコンパイラなどの研究がなされているが, コンパイラのレベルから考えても, アルゴリズムから直接自動抽出するということは難しい. 実際にはプログラマがそのアルゴリズムから並列化方法を考え, 並列言語でプログラミングする方法がより多く使われている.

また, 並列処理の並列単位の大きさを示す粒度 (granularity) についても考えなくてはならない. 細粒度になればなるほど, 高い並列性を抽出することができるが, 一つの処理単位が小さくなる. その結果アーキテクチャによっては, 通信, 同期, 負荷分散, スケジューリングが必要となる.

2.1 マルチプロセッサ方式

マルチプロセッサ方式は計算機機構内に存在するデータの流れ (data stream) と, 命令の流れ (instruction stream) に注目した場合, 単一命令流 (single instruction) の SIMD (Single Instruction Multiple Data stream) 型と, 複数命令流 (multiple instruction) の MIMD (Multiple Instruction Multiple Data stream) 型に大別される.

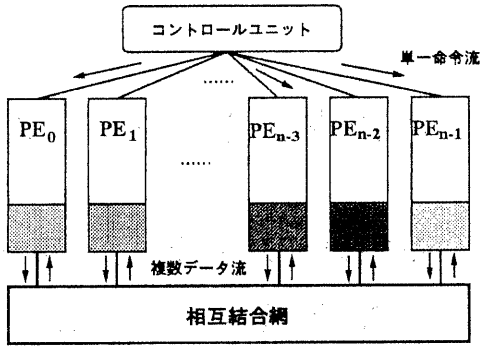


図 1: SIMD 型の概念図

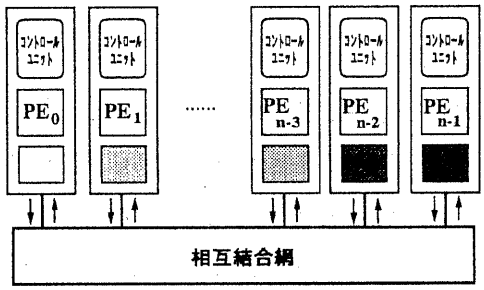


図 2: MIMD 型の概念図

SIMD 型の計算機は, 一つのコントロールユニットで複数の要素プロセッサを実行させる. すべてのプロセッサは共通のグローバルクロックをもち, 同期して動作する. 条件分岐をおおく含まず制御の流れが単純な細粒度の並列演算に向いている.

一方 MIMD 型の計算機は, 各要素プロセッサが独自のプログラムカウンタを持つため, 他のプロセッサとは異なった命令を実行できる. また, MIMD 型計算機は共有メモリ型と分散メモリ型に分けることができる. 共有メモリ型は, すべてのプロセッサがバス結合により共有メモリ区間のすべてのメモリを参照することができる. 分散メモリ型は, それぞれのプロセッサがローカルにメモリを保持して, プロセッサ間の通信はメッセージ交換で行なう. アルゴリズムの設計は, プロセッサ間の結合形態 (2 次元格子, ハイパーキューブ結合など) に大きく依存する.

図 1, 2 に SIMD 型と MIMD 型の概念図を, 表 1 に SIMD 型と MIMD 型の比較を示す.

表 1: SIMD 型 と MIMD 型の比較

	SIMD 型	MIMD 型
制御方式	一つのプログラムカウンタで全プロセッサを制御	各プロセッサは各々独立したプログラムカウンタで制御
動作	同期	非同期
プロセッサ粒度	細粒度	中流度
スケーラビリティ	大規模	中規模
応用分野	特定用途	汎用
負荷分散	不必要	必要
アルゴリズムの設計	比較的容易	困難
アルゴリズムの解析	容易	困難

2.2 パイプライン

一連の仕事を, 処理の順序にしたがって数段の処理に分解し, それぞれに対して独立したハードウェアを割り当て, 各段での処理が終るたびにその処理結果を次段に引き渡す方式である. 例えば, 画像生成処理全体は巨視的にみて, 幾何学処理, レンダリング処理, 表示処理に分けことができる [25, 26, 34]. また, Z バッファ法や, スキャンライン法のようなアルゴリズムの場合, 幾何学処理はさらに座標変換とクリッピング処理に, レンダリング処理はスキャンコンバージョン, スムーズシェーディング, 隠れ面処理に分解できる. これらの処理を直列につなぐことでパイプラインを実現できる.

2.3 シストリックアレイ

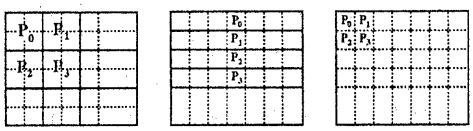
幾何学的に規則正しく配置された演算ユニットに, 同期クロックに従って定期的に入力ポートからデータを流し込む. 各演算ユニットは, 前段の演算ユニットの出力結果を入力とし, 全体で同期して演算を実行し, 演算結果を次段の演算ユニットに出力する. 出力データは同期クロックに合わせて次々に出力ポートから取り出される. この方式は, データと制御の流れが規則的な演算レベルの並列処理に使用されることが多い.

3 レンダリングアルゴリズムの並列処理

ハードウェアによる並列化に頼るのではなく, アルゴリズム自体の特徴を生かした並列化手法である. それぞれのアルゴリズムの中で並列に処理できる部分を抽出する. ここでは特にマルチプロセッサの場合について紹介する.

4 Z バッファ法の並列アルゴリズム

Z バッファ法はアルゴリズムが簡単であり, また Z 値の比較, 更新がピクセル毎に行なえるため, これを生かした並列化が行なえる. 原則的にはきわめて大きな並列度を得ることができる. 現在のハードウェア技術では, 数台から数百台のプロセッサに並列に処理をさせるものが商品化されている.



(a) ブロック単位　(b) スキャンライン単位　(c) ピクセル単位

図 3: プロセッサへのピクセルの割り当て方式

負荷の分散方法としては, 図 3 にあるように, (a) ブロック単位, (b) スキャンライン単位インターリーブ, (c) ピクセル単位インターリーブがある.

(a) の方式は, 画面を水平, 垂直に分割してできる小領域をプロセッサに割り当てる手法である. 各プロセッサでは, 独立に全データをスキャンコンバージョンして, Z 値と輝度を計算していく. MIMD 型計算機を用いれば高速計算が可能だと考えられるが, 実際には各プロセッサ毎に計算負荷が異なるため, 次に処理すべき小領域を割り当てる機構が必要になる [46, 47].

(b) の方式は, n 個の各プロセッサに n 本間隔にスキャンラインを割り当てていく手法である. この手法は DDA (Digital Differential Analyzer) によるスキャンコンバージョンが可能であり, なおかつプロセッサの負荷がほぼ均等であるという利点がある [28, 29, 38].

(c) の方式は, ピクセル単位で割当をするため, コヒーレンシーが使用できず, 効率が悪いという欠点がある. そこで, 各プロセッサが形状データを通常の方法でスキャンコンバージョンをするのではなく, 担当するピクセルが現在処理中の投影領域内にあるかどうかを直接判定する方法が考えられる [1, 34].

5 スキャンコンバージョンの並列アルゴリズム

透視投影によりスクリーンに投影された形状プリミティブを, スキャンコンバージョンしてピクセル毎の輝度やZ値を算出し, フレームメモリに書き込む一連の動作の並列化を考える. この並列化に関しても4つの段階に分けて考えることができる.

ピクセルレベルの並列処理 一つのピクセルに対して x, y, z, R, G, B の計算とメモリアクセスを, 複数の演算器で行なう.

形状プリミティブレベルの並列処理 一つのプリミティブをスキャンコンバージョンする際, 複数のスパンを生成し, そのスパンのピクセル処理を複数の演算器に並列に処理させる方法. 辺関数により平面を分割して, 頂点の輝度より線形補間する方法が使われる [31].

物体レベルの並列処理 形状のプリミティブに対し, 複数のプロセッサを割り当てる方法 [29].

スクリーン分割による並列処理 全画面を分割して, 複数のプロセッサに割り当て, 並列に処理させる方式.

6 スキャンライン法の並列アルゴリズム

スキャンライン法は, 一つのyバケットリストを使用して上から順にスキャンライン毎に順次操作を行なうアルゴリズムである. 画面をy方向に分割しておのおのの領域の走査を複数のプロセッサに担当させることにより並列化することが可能である [16, 40]. また, 一つのスキャンラインの処理を考えたとき, この処理は各スパンに対する可視セグメントの決定と各ピクセルの輝度値の補間計算であり, スパン毎に独立して実行することが可能であるので, 複数のプロセッサを用いてスパン単位で分担して処理することにより並列処理をすることができる [26].

7 レイトレーシング法の並列アルゴリズム

スクリーンを通して視点に入ってくる光を, 光の逆進性を利用して逆に視点から探索しその光の振舞いを調べ, スクリーン上に結像するアルゴリズムである. 光の反射・屈折が表現できるが, このため探索する視線が計算時間がかかる. 視線と物体との交点計算が全体の計算時間の 75 ～ 95% を占めるといわれ, その計算を削減する研究が行なわれてきた.

7.1 画素・小画面単位の並列

各画素を通る光線は互いに独立である, という性質を利用して各画素または画面をいくつかに分割したその小画面に対してプロセッサを割り当てる手法が考えられる. またその分割方法には, スキャンライン単位 [50], 任意の大きさの小画面単位 [6, 7, 13, 23, 40, 41] などの分割方法が考えられが, これらはいずれも割り当てられるプロセッサの数よりも十分多く分割することで, 負荷の分散を行なっている.

各画素または小画面を割り当てられたプロセッサはその担当した領域により, 物体数, 物体の属性 (反射物・透過物), 複雑度により負荷が異なる. 負荷の分散方法には静的な負荷分散と動的な負荷分散がある.

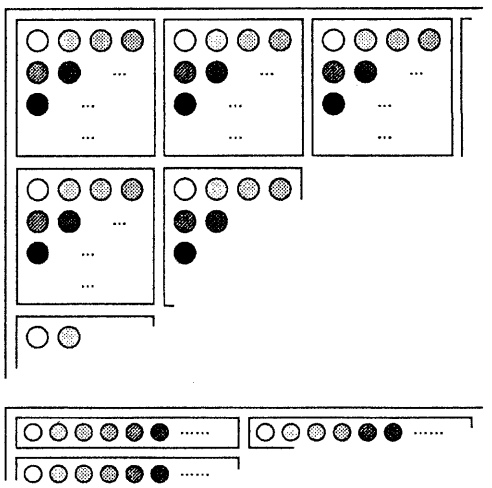


図 4: 画素単位の静的負荷分散

図 4 のように, 画面全体をプロセッサ数と同じ画素数を含む小画面に分ける. 小画面内の各画素に対してプロセッサを割り当て, 描画処理の終わったプロセッサは次の小画面の画素を処理する. 全体としては飛び飛びの画素を処理することになる. これは分割された小画面の負荷がほぼ均一という考えに基づき, あらかじめ静的にプロセッサを割り当てている [23].

動的負荷分散では, 小画面を割り当てられたプロセッサは, その処理が終了したものからホストに対して終了した旨を伝え, 新たに小画面を割り当てられる [6, 7, 13]. 分割する小画面が多くなればそれだけ終了時間のばらつきが少なくなり負荷を分散できるが, ホストプロセッサとの通信が多くなりオーバーヘッドとなる.

またマルチプロセッサによる並列化手法だけでなく, 従来の高速化手法である交差判定を削減するバウンディングボリューム, オクトリー, 空間分割や [6, 7, 23], 探索する視線の数を減らす画素選択法 [40] などの手法を同時に用いることでそれぞれのプロセッサの処理時間を短縮する.

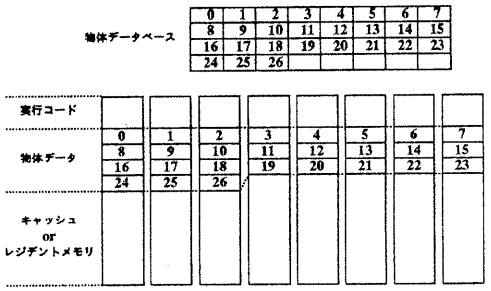


図 5: データの分割とメモリの使用法

各プロセッサのローカルメモリが十分で, 物体データがすべて入るとき, 小画面単位のレイトレーシング法は非常に効果的である. しかし物体データが多くローカルメモリに入り切らないような場合, ホストから必要なデータを随時ロードしなくてはならなくなる. そこで図 5 のように物体データベースを分割して, 各プロセッサのローカルメモリ上に配置する. メモリの残りの領域はキャッシュ, レジデントメモリにあてる [2, 11]. 必要な物体データがローカルメモリ上にない場合は, データを持つプロセッサから通信によりデータを転送し, キャッシングする. このアルゴリズムはプロセッサが多くなればそれだけ各ローカルメモリのキャッシュを多く採ることができるので, 通信のオーバーヘッドを削減できるという利点がある.

7.2 小空間単位の並列

物体空間を小空間 (ボクセル, オクトリーなど) にわけ, その小空間に対してプロセッサを割り当てる. 各プロセッサ間では光線 (レイ) をメッセージとしてやりとりし, 最終的に全光線を回収して描画する, という手法である.

静的負荷分散手法 [2] では, 物体空間をあらかじめ任意の大きさの小空間に分ける. 次に少ない光線で視線探索を行ない, どの小空間にどのくらい光線が入ったかをカウントし, 物体空間全体での大まかな負荷を計算する (サブサンプル). これは, 隣合うピクセルを通過する光線は, ほぼ同じ領域を通過するというレイ・コヒーレンシーによる. 図 6 のよ

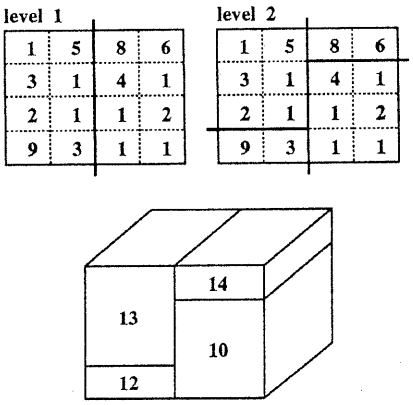


図 6: サブサンプリングによる静的負荷分散の例

うに, ほぼ同じ負荷になるように小空間をプロセッサの台数分にまとめ, まとめた領域に対してプロセッサを割り当て, 光線のメッセージ交換により描画する. 問題点としては, サブサンプルをどのくらい行なえば良いのか, 負荷が必ずしも均一に分散されない, などがある.

また, 離れたピクセルを通過する光線は画像空間内でも離れた小空間を通過するので, 互いに離れたピクセルから計算するようにし, 互いに近い二つのピクセルは違う時間に計算することにより, 静的に負荷を分散する手法も考えられている [32].

動的負荷分散では, 小空間を割り当てられたプロセッサの負荷が, 通信バッファのキューに光線がたまるなどで重くなった場合に適宜負荷を分散させる. 具体的には, ホストとなるプロセッサに対し負荷の分散を要求し, 新たに処理の軽いプロセッサに処理を分散する手法 [39] や, 周りのプロセッサと自分を比べもっとも負荷の軽いプロセッサに対して処理を分散する [8] などの手法が考えられる. このとき光線データとともにそのプロセッサが所有している小空間のデータも転送しなければならない. そのため, 各プロセッサでのデータのキャッシングは不可欠である. また空間の分割に際して, ボ

クセル分割法は, オクトリー表現に比べデータベースが大きくなるが, 光線のトラバース処理を 3DDDA (3D Digital Differential Analyzer) により高速に計算ができるという利点があるため, 小空間単位の並列レイトレーシングではよく使われるようである [18, 38].

小空間単位の並列レイトレーシングは, 通信によるオーバーヘッドのために, プロセッサの台数が増えたときその台数効果があまりあらわれず, 頭うちになるようである. また, 影処理の際に, 光源を含む小領域を持つプロセッサにシャドウレイが集中し, 通信バッファが溢れデッドロックを起こすなどの問題点がある [2, 32].

7.3 その他

交差判定の行列計算やベクトル計算の各要素を, SIMD 型計算機の各プロセッサに割り当て計算する手法 [48] や, 交点計算中に含まれるベクトル計算を専用の並列演算プロセッサを用いて [50] 高速化する試みがなされている. また, 物体探索, 交点計算, 輝度計算の 3 ステージでパイプライン処理する [24, 25] 手法がある.

8 ラジオシティ法の並列アルゴリズム

相互反射も考慮に入れるラジオシティ法では, 実用化の段階で問題になるのが計算時間である. その中でも特にフォームファクタの計算はボトルネックとなる. そこで, ヘミキューブを用いたフォームファクタの計算が並列に実行できることからフォームファクタ計算の並列化などが試みられてきた [20, 43].

また, フォームファクタを求めラジオシティ方程式を解く手法として, ガウスザイデル法があげられるが, この手法ではすべてのフォームファクタをメモリ上に保持しておかなければならないためパッチの数が非常に多いときには効率が良くない. そこでこの問題に対しては, 連立方程式を解く過程で必要になった時点でフォームファクタを計算する効率の良い方法が提案されている [5]. この手法を使った並列化ではパッチ, エレメントをプロセッサに分散し, ブロードキャストによってエネルギーのやりとりをするという方法をとっている [19, 30]. また, 動的に負荷の分散を行なうための手法も紹介されている [30].

9 まとめ

コンピュータグラフィクスを並列計算機上で実装する際には, 並列処理特有の通信オーバーヘッドやアイドル等を考慮する必要がある. また, ノードプロセッサのローカルメモリのサイズによってもアルゴリズムは大きく変わってくる. この点の工夫なしには高い並列度を引き出せず, したがって台数効果による性能向上は望めない.

参考文献

[1] Apgar, B., Bersack, B. and Mammen, A., "A Display System for the Stellar Graphics Supercomputer Model GS1000," *Proc. of SIGGRAPH*, Vol. 22, No. 4, pp. 255–262, 1988.

[2] Badouel, D., Bouatouch, K. and Priol, T., "Ray Tracing on Distributed Memory Parallel Computers: strategies for distributing computations and data," *SIGGRAPH '90 Parallel Algorithms and Architecture for 3D Image Genaration Course Notes*, No. 28, pp. 185–198, 1990.

[3] Baum, D. R. and Winget J. M., "Real Time Radiosity Through Parallel Processing and Hardware Acceleration," *Computer Graphics(1990 Symposium on Interactive 3D Graphics)*, Vol. 24, No. 2, pp. 67–75, 1990.

[4] Bishop, G., Monger, M. and Ramsey, P., "A Visualization Programming Environment for Multicomputers," *IEEE Computer Graphics & Applications*, pp. 50–58, July 1990.

[5] Cohen, M. F., Chen, S. E., Wallace, J. R. and Greenberg, D. P., "A Progressive Refinement Approach to Fast Radiosity Image Generation," *Proc. of SIGGRAPH*, pp. 75–84, August 1988.

[6] 出口, 西村, 吉村, 河田, 白川, "コンピュータグラフィックスシステム LINKS-1 における画像生成の高速化手法," 情報処理学会論文誌, Vol. 25, No. 6, pp. 944–952, 1984.

[7] 出口, 西田, 西村, 河田, 白川, 大村, "視線探索法による画像生成のための木構造並列処理システム," 電子通信学会論文誌, Vol. J69-D, No. 2, pp. 170–179, 1986.

[8] Dippé, M. and Swensen, J., "An Adaptive Subdivision Algorithm and Parallel Architecture for Realistic Image Synthesis," *Proc. of SIGGRAPH*, Vol. 18, No. 3, pp. 149–158, 1984.

[9] Franklin, Wm. R. and Kankanhalli, M. S., "Parallel Object-Space Hidden Surface Removal," *Proc. of SIGGRAPH*, Vol. 24, No. 4, pp. 87–94, 1990.

[10] Ghosal, D. and Patnaik, L. M., "Parallel Polygon Scan Conversion Algorithms: Performance Evaluation on a Shared Bus Architecture," *Computer & Graphics*, Vol. 10, No. 1, pp. 7–25, 1986.

[11] Green, S. A. and Paddon D. J., "Exploiting Coherence for Multiprocessor Ray Tracing," *IEEE Computer Graphics & Applications*, pp. 12–26, November 1989.

[12] 日高, 平井, 中瀬, 浅原, 鷲島, "マルチコンピュータ画像生成システム MC-1," 情報処理学会計算機アーキテクチャ研究会, Vol. 58, No. 5, 1985.

[13] 日高, 平井, 中瀬, 浅原, 鷲島, "マルチコンピュータ画像生成システム MC-1 における画像生成手法," 情報処理学会グラフィクスと *CAD* 研究会, Vol. 18, No. 4, 1985.

[14] İşler, V. and Özgüç, B., "Fast Ray Tracing 3D Models," *Computer & Graphics*, Vol. 15, No. 2, pp. 205–216, 1991.

[15] 河合, 山下, 大野, 吉村, 西村, 下條, 宮原, 大村, "並列画像生成システム LINKS-2 のアーキテクチャ," 情報処理学会論文誌, Vol. 29, No. 8, pp. 729–740, 1988.

[16] Kelley, M., Winner, S. and Gould, K., "A Scalable Hardware Render Accelerator using a Modified Scanline Algorithm," *Computer Graphics*, Vol. 26, No. 2, pp. 241–248, 1992.

[17] Kobayashi, H. and Nakamura, T., "A Massively Parallel Processing Approach to Fast Photo Realistic Image Synthesis," *Proc. of CGI '93*, pp. 497–507, 1993.

[18] 窪田, 西村, 小林, 中村, 重井, "光線追跡法のための空間分割型並列処理の負荷分散法," 電子情報通信学会全国大会, Vol. 6, No. 318, pp. 1640, 1987.

[19] 国谷, 近藤, 佐藤, 島田, "ラディオシテイ法のための高速並列演算手法," 情報処理学会全国大会, Vol. 5U-7, No. 2, pp. 375–376, 1994.

[20] Lamotte, W., Reeth, F. V. and Vandeurzen, L., "Parallel Processing in Radiosity Calculations," *Proc. of CGI '93*, pp. 485–496, 1993.

[21] Levinthal, A. and Peter, T., "Chap - A SIMD Graphics Processor," *Proc. of SIGGRAPH*, Vol. 18, No. 3, pp. 77–82, 1984.

[22] 水谷, 中嶋, "超並列ビジュアライゼーションに関する研究," 電子情報通信学会春季大会, Vol. D-142, No. 6, pp. 144–145, 1994.

[23] 村上, 佐藤, 広田, "セルラアレイプロセッサ CAP によるレイトレーシング," 情報処理学会グラフィクスと *CAD* 研究会, Vol. 22, No. 2, 1986.

[24] 村田, 村上, 富田, "「熱視線」: 視線探索法を高速処理する専用並列レンダリングマシン –メモリ構成およびその評価–," 情報処理学会計算機アーキテクチャ研究会, Vol. 97, No. 13, pp. 97–104, 1992.

[25] 村田, 村上, 富田, "レイトレーシング法を高速処理する専用並列レンダリング・マシン「熱視線」の要素プロセッサ・アーキテクチャ," 情報処理学会論文誌, Vol. 34, No. 7, pp. 1650–1662, 1993.

[26] Niimi, H. Imai, Y. Murakami, M. Tomita, S. and Hagiwara, H., "A Parallel Processor System for Three-Dimensional Color Graphics," *Proc. of SIGGRAPH*, Vol. 18, No. 3, pp. 67–76, 1984.

[27] 牧, "実用化が進む CG 向け並列コンピュータ," 日経 *CG*, No. 8, pp. 10–20, 日経マグロウヒル社, 1987.

[28] 西尾, 西村, 峰久, 平井, 中瀬, "画像生成におけるマルチプロセッサシステム管理の一手法," 電子情報通信学会研究会資料, Vol. CPSY90-63, pp. 151–156, 1990.

[29] 大西, 青木, 戸村, 吉良, "ハイビジョン CG 用高速画像生成装置の並列処理手法," 電子情報通信学会技術研究報告, Vol. IE89, No. 61, pp. 1–6, 1989.

[30] 大谷, 米田, 日高, 浅原, 鷲島, "ラジオシティ法の一並列化," グラフィクスと *CAD* シンポジウム論文集, pp. 95–104, 1993.

[31] Pineda, J., "A Parallel Algorithm for Polygon Rasterization," *Proc. of SIGGRAPH*, Vol. 22, No. 4, pp. 17–20, 1988.

[32] Pitot, P., "The Voxar Project," *IEEE Computer Graphics & Applications*, pp. 27–33, January 1993.

[33] Potmesil, M. and Hoffert, E. M., "FRAMES: Softoware Tools for Modeling, Rendering and Animation of 3D Scenes," *Proc. of SIGGRAPH*, Vol. 21, No. 4, pp. 85–93, 1987.

[34] Potmesil, M. and Hoffert, E. M., "The Pixel Machine: A Parallel Image Computer," *Proc. of SIGGRAPH*, Vol. 23, No. 3, pp. 69–78, 1989.

[35] 鷲島, 西澤, 浅原, "並列図形処理," コロナ社, 1991.

[36] Sato, H., Ishii, M., Sato K., Ikesaka, M., Ishihata, H., Kakimoto, M., Hirota, K. and Inoue, K., "Fast Image Generation of Constructive Solid Geometry Using A Cellular Array Processor," *Proc. of SIGGRAPH*, Vol. 19, No. 3, pp. 95–102, 1985.

[37] 佐藤, 石畑, 波形, "高並列計算機 CAP-II による三次元グラフィクス," 電子情通信学会研究会資料, Vol. CPSY 90-60, pp. 133–138, 1990.

[38] 鈴木, 青木, 宮崎, "ハイビジョンCG用高速画像生成装置による画像生成," 電子情報通信学会技術研究報告, Vol. IE89, No. 61, pp. 7–14, 1989.

[39] 竹安, 河合, 青木, 大西, "空間分割による並列視線探索法の画像生成装置MAGGへの実装," 情報処理学会グラフィクスと *CAD* 研究会, Vol. 60, No. 8, pp. 57–62, 1992.

[40] 玉邑, 秋本, "高速画像生成装置 -MAGIC- のCG処理環境と応用," 情報処理学グラフィクスと *CAD* 研究会, Vol. 35, No. 7,, 1988.

[41] 寺西, 河合, 小沢, 青木, 大西, "画像生成装置MAGGのための並列双方向視線探索法," 情報処理学会グラフィクスと *CAD* 研究会, Vol. 60, No. 7, pp. 49–56, 1992.

[42] Torborg, J. G., "A Parallel Processor Architecture for Graphics Arithmetic Operations," *Proc. of SIGGRAPH*, Vol. 21, No. 4, pp. 197–204, 1987.

[43] 上嶋, 山崎, 渡部, 得丸, "トランスピュータによるラジオシティ法の並列化," 情報処理学会全国大会, Vol. 5U-7, No. 2, pp. 373–374, 1994.

[44] 梅尾, "SIMD上の並列アルゴリズム," 情報処理, Vol. 33, No. 9, pp. 1042–1055, 1992.

[45] Wang, Y., Mangaser, A. and Srinivasan, P., "A Processor Architecture for 3D Graphics," *IEEE Computer Graphics & Applications*, pp. 67–75, September 1992.

[46] Whitman, S., "Computer Graphics Rendering on a Parallel Processor," *SIGGRAPH '90 Parallel Algorithms and Architecture for 3D Image Genaration Course Notes*, No. 28, pp. 167–183, 1990.

[47] Whitman, S. and Parent, R., "A Survey of Parallel Hidden Surface Removal Algorithms," *SIGGRAPH '90 Parallel Algorithms and Architecture for 3D Image Genaration Course Notes*, No. 28, pp. 153–166, 1990.

[48] 山田, 安浦, "FMPPによるレイトレーシングの高速化手法について," 情報処理学会計算機アーキテクチャ研究会, Vol. 97, No. 14, pp. 105–112, 1992.

[49] 安浦, "並列計算機と並列計算モデル," 情報処理, Vol. 33, No. 9, pp. 1024–1032, 1992.

[50] 吉田, 成瀬, 高橋, "画像生成計算機SIGHT-2," 情報処理学会計算機アーキテクチャ研究会, Vol. 77, No. 6, pp. 43–50, 1989.

[51] 吉嶋, 佐藤, "高並列コンピュータにおける可視化システム," *NICOGRAPH* 論文集, pp. 104–111, 1993.