グラフィクスとCAD 22－2
（1986・7・11）

セルラアレイプロセッサCAPによるレイトレーシング

村上　公一，　佐藤　弘幸，　広田　克彦

㈱富士通研究所　　システム研究部

レイトレーシングによる映像生成の高速化を目的として，マルチプロセッサシステムCAPによるレイトレーシングの並列処理を実現し，性能評価を行った。空間分割とパラメトリック3次元DDAを用いた高速化アルゴリズムを提案し，実現・評価を行った結果，空間分割を使わないシステムの十～数百倍の高速化が達成された。また，以前に提案した，CSG表現から直接に画像を生成するための状態木法を，分割空間毎に適用する部分評価方式についても議論する。レイトレーシングの並列化実現に関しては，負荷の静的分散方式を提案し，オーバーヘッドの少ない良好な分散効果を得た。さらに，汎用計算機との性能比較も行い，大型機に匹敵する性能を確認した。

Parallel Ray-tracing on a Cellular Array Processor(CAP)

Kouich Murakami, Hiroyuki Sato, Katsuhiko Hirota
FUJITSU LABORATORIES LTD.
1015, Kamikodanaka, Nakahara-ku, Kawasaki

Parallel ray-tracing system on a Cellular Array Processor(CAP) is developed and analyzed. We propose the algorithm that uses space-division and parametric 3DDDA(3 Dimensional Digital Differential Analyzer) to reduce the number of ray-object intersection calculations. As a result, this system can generate images ten to several hundred times faster than conventional ones without space-division scheme. Also, we previously proposed CSG evaluation method, called status tree method that generates images directly from CSG representation. By applying this method for each subspace, boundaries are efficiently evaluated during ray-tracing phase. The program is efficiently executed on CAP with a static load distribution mechanism. The performance is comparable with a main frame computer M380.

1．はじめに

コンピュータグラフィックスにおいて，光線追跡法は高品質な画像の生成方式として期待されているが，膨大な生成時間を要するという欠点がある。本論文では，光線追跡法の高速化を実現するためのアプローチについて述べる。

アルゴリズムによる高速化のアイデアは処理の大半を占める光線と物体の交差計算の回数を減らすことである。この方法として，空間配置に依存したモデルの階層化［1］，世界空間を分割して部分空間毎に交差計算を行う方式［2］［3］［4］等がある。本論文の2章では筆者等の提案する，空間分割とパラメトリック3次元ＤＤＡを用いた高速化アルゴリズムについて述べる。本方式によって従来の十倍から数百倍程度の高速化が達成出来る。

また，筆者等はCSG 表現から B-rep表現に変換する境界評価処理［5］を行わずに，直接 CSG表現から画像を生成する状態木法［6］［7］を提案した。この方式は，物体の可視部分の処理しか行なわないので効率の良い処理ができるという特徴を持っている。本論文では，状態木法を分割空間毎に適用し，より高い効率を指向した部分評価方式についても提案する。

一方，ハードウェアからの高速化方式としてスクリーン上の画素から出射する光線の計算独立性に着目して，画素単位に並列処理を行う方式［8］がある。3章では，本システムを実現したマルチプロセッサシステムＣＡＰのアーキテクチャ［9］，その上での本システムのインプリメンテーション，および筆者等の提案する静的負荷分散方式について述べる。さらに，アンタイエイリアシングされた画像や局所的に負荷が偏った画像に対して有効なフレーム平均化静的負荷分散方式についても述べる。4章では，空間分割による高速化，静的負荷分散法の評価を行い，5章でまとめを行う。また，本システムは汎用計算機M380上にも実現された。同時に，ＣＡＰとM380上のシステムの比較・検討を行う。

2．空間分割による高速化方式

光線追跡法の重要な問題点として，画像生成時間の遅さが指摘される。これは，光線と物体の交差計算の数が膨大なためで報告［4］によると処理時間の大半は，この処理に費やされている。従って，交差計算の回数を減らすことが高速化の鍵となる。この方法をアルゴリズム－1に示す。

Step1:	空間を部分空間に分割し，そこに存在する物体を前処理で求める。
Step2:	光線が通過する空間要素を高速に求める。
Step3:	もし，その部分空間内に物体が存在すれば，その物体のみと交差計算を行う。さもなければ，Step2 の処理から繰り返す。

アルゴリズム－1　高速光線追跡法

空間の分割方法は，(1)octree分割［2］［3］と(2) voxel分割［4］の方式に分類できる。両方式とも，光線の部分空間でのトラバースコストが問題となる。一般にoctree分割は，使用記憶量の点で優れるが，トラバースコストは高い［4］。以前，筆者等はoctree分割による高速化方式［2］を実現したが，高い効果は得られなかった。ここでは，voxel 分割を用いて高速化を達成した。

2.1 パラメトリック3DDDA

3DDDA(3-Dimensional DDA)についてはすでに報告［4］があるが，本論文では光線の直線方程式

$$\vec{x} = \vec{\alpha}t + \vec{\beta} \qquad (1)$$

における直線パラメタｔを使うパラメトリック方式を提案する。octree空間内のトラバースは除算を用いるが，voxel 空間での3DDDA を使ったそれは増分計算のみで行うことができる。Δtx，Δty，Δtzをｘ，ｙ，ｚ軸間の光線に沿った距離，tx,ty,tzを光線が次に横切るｘ，ｙ，ｚグリッド上のｔパラメータの値とする。

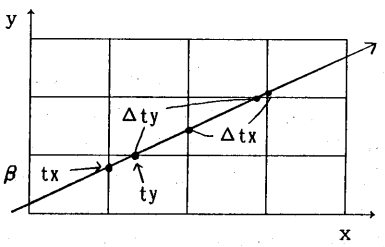


図2-1 トラバースの原理図（2次元）

図2-1 から分るように，光線が次に入る voxel要素は，現在の voxelインデクス(vx,vy,vz)と最小のtx,ty,tzから相対的に求めることができる。2次元で考えると，tx > tyの時は光線はｘ境界より先にｙ境界を横切り，voxel イ

ンデクスのｙ成分が変化する。すなわち，光線は voxel空間をｙ方向に一つ進む。tx <ty の時はこれとは逆にｘ方向へ進む。tx,ty,tzは光線が次のvoxel 境界と交差するｔパラメタであるので，各方向へ進んだ後に，対応する増分を加えて更新する。但し，一回の処理では，一つの成分しか更新されない。これをアルゴリズム－２に示す。この手続の主要部は２個の加算と比較で達成される。

```
while(空間の領域内)  {
  最小のtx,ty,tzとその軸を求める;
  switch (最小の軸)  {
    case X:
            vx +=  incre_x
            tx +=  Δtx;
            break;
    case Y:
            .
  }
}
```

アルゴリズム－２　3DDDA方式

ここで，incre_x は±１の値をとり光線の傾きから決定される。この値と初期 voxelインデクスとｔ増分は，ある光線に対して3DDDA の初期計算として求められる。

例外処理として，光線が境界のある成分と交差しない場合がある。例えば$\alpha x = 0$ の時，この光線とｘ境界とは交わらない。初期値として tx = ∞とすることで，ｘ方向に横切ることを自動的に防ぐことが出来る。

2.2 Voxel 生成方式

Voxel 生成は，表示の前処理としてモデルを空間的にソートする処理である。具体的には，表示対象物体の存在する直方体状の空間(ユニバース) を各軸等間隔に区切りそれぞれの voxelに交差する物体要素(プリミティブ) を求め記憶する。

Voxel 生成は，光線追跡法の処理に比べ充分高速に行う必要がある。ことに，光線追跡法処理が並列化され高速になると voxel生成がオーバーヘッドになる可能性が出てくる。我々は，(1)高速性，(2)高精度，(3)並列化可能なことの３点を考慮してアルゴリズムの設計を行った。

基本的なアルゴリズムを次に示す。

Step1:　ユニバースの位置と大きさを求める。

Step2:　すべてのプリミティブについて次の処理を行う。

2.1 プリミティブの外接直方体を生成し，世界座標系へ変換する。

2.2 外接直方体をX-Y 平面へ平行投影した多角形が交差するX-Y 平面上のPixel(voxel のX-Y 平面への投影) 領域を求める。

2.3 この領域内のPixel について，それぞれのPixel のX-Y の範囲で外接直方体のＺの最大値と最小値を近似的に求め，交差するVoxel の存在範囲を求める。

アルゴリズム-3　Voxel 生成法

あるPixel において，ある外接直方体のＺの最大値，最小値は，外接直方体の各面を無限平面として扱い，次式で近似的に求める。

$$Z_{max} = min_{f \in ff}(max_{x \in X, y \in Y}(z(f,x,y)))$$
$$Z_{min} = max_{f \in bf}(min_{x \in X, y \in Y}(z(f,x,y)))$$

z(f,x,y)は，面f 上の点 x,yにおけるz の値
ff:　表面(面の方向ベクタのz 成分が正)
fb:　裏面(面の方向ベクタのz 成分が負)
X,Y:　Pixel の4 角の点のx,y 座標をとる

外接直方体で voxel生成した場合，プリミティブが voxelより大きい時には，実際に交差しない voxelが発生する。外接無限平面を追加し，より多くの面で囲むことで，精度の向上を図っている。

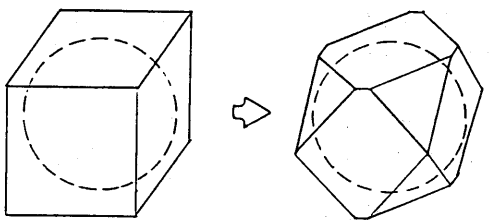


図2-2　外接無限平面の付加による精度の向上

2.3 状態木による CSG評価と部分評価方式

以上，交差計算の回数を減らすための方式として空間分割による方式について説明したが，光線追跡法の高速化に寄与する別の考慮点に物体の表現形式がある。ここでは，CSGデータ構造で記述された三次元形状データを B-rep表現に変換せずに，光線追跡法の枠組みで直接表示する方式について述べる。境界評価を行ってB-rep 表現のデータを作成して表示処理を行うのに比べ，この方法は物体の見え

る部分のみしか処理しないことや誤差を表示の精度内に保持すれば良いことなどの点で有利である。また，境界評価処理自体が不要であり会話型処理などの際の空白な時間を無くすることができる。さらに，データの個数が少ないため交差計算の回数を減少させることができ，転送量や記憶量の点でも優れる。

本方式では，光線を追跡しながら光線上の一次元的な集合演算評価を行って実際の可視点（論理演算の評価後）を決定する。集合演算の評価（以下 CSG評価）に，状態木データ構造［6］を採用する。状態木の構造は，演算子をノードに，プリミティブをリーフに持ち，リンクはボトムアップに張られている。（図2-3）

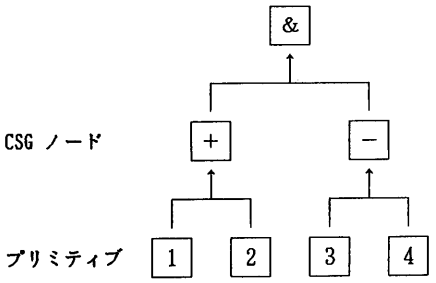


図2-3　状態木の構造

各ノードに保持されている状態（図2-4)は，そのノードをルートとする部分木の論理値，すなわち CSG評価を考慮した論理的な可視／不可視を表す。

Operator	PointerToParent	
StatusSelf	StatusLeft	StatusRight

図2-4 状態木ノード

これを使って，隠面処理を行う方式をアルゴリズム－4に示す。

Step-1: ある光線と交わる総てのプリミティブとの交点を検出し（交点検出計算），視点側からソートする。

Step-2: 視点に近い側から状態木のリーフに注目点を移し，その親のノードに遷移する。（注目点の遷移）

Step-3: 演算子ノードでは，左右の子の状態を使って演算を行う。その結果が，ノードに記録されている(StatusSelf)前の状態と異なるときはこれを更新し，更なる遷移を行う。

Step-4: この遷移をルートまで繰り返し，ルートに達した場合は，伝播された値をシーンの論理的可視値とする。

アルゴリズム－4　状態木を使った CSG評価法

この方式で重要な点は，状態木を使った CSG評価法は副作用を伴って遷移を行うため，あるノードの評価の結果が前のそれと同じ場合は更なる遷移を省けることである。このため，毎回ルートノードまでのトラバースが不要となる。

また，本方式はボトムアップに評価を進めるので，遷移の開始を任意のプリミティブから行なえる。一方，Roth［10］の方法では，トップダウンに評価する。この方式を使うと，全交点情報を求めてからでないと CSG評価を行えない。以上の２点が，状態木法が Roth の方式と比べて優れている点である。

次に，この状態木法を voxel要素単位で行う部分評価方式について述べる。この方式は，光線に沿って視点側から逐次的に voxel単位で交差計算と CSG評価を行い，ある voxel要素で可視となった場合は処理を終える。部分評価法では，空間分割で大局的にソートされた情報を有効に使って処理を大巾に高速化している。Rothの方法では，ある光線に交差する総てのプリミティブとの交点を求めておく必要がある。本方式では，光線が入った voxel空間毎に交差計算，交点ソート，CSG評価を行う。このため，一回の処理に必要な情報はその部分空間内の交点に関する情報で十分であり，可視点より奥にある物体に関する処理を省ける。これは，無駄な交差計算やソートの数を大巾に減少できることを意味する。

3.　ＣＡＰへの実現

3.1 CAP のアーキテクチャ

CAP は，64台のセルと呼ぶプロセッサを二次元格子上に接続したMIMD型の並列計算機である。（図3-1）ホストコンピュータとセルとは，コマンドバスという共通バスで結ばれ放送によってホストコンピュータからセルへ効率のよいデータ転送を行なえる。セルがこのバスを介して他のセルやホストコンピュータにデータを送ることもできる。

セルの構成を図3-2 に示す。CAP では，セルのビデオメモリの内容がリアルタイムでビデオをバスを通じて表示さ

れるのが特徴である。ビデオメモリの画像データをスクリーン上のどの位置にどういう形（パターン）で表示するかは，セルのウィンドコントローラが制御する。標準的なパターンとして，次の画素担当モードがある。

(1)ブロック：矩形領域を担当

(2)ライン：　飛び飛びの水平ラインを担当

(3)ドット：　飛び飛びの画素を担当

このような画像の分割モードは，表示のアルゴリズムや，負荷分散等を考慮して選択する。

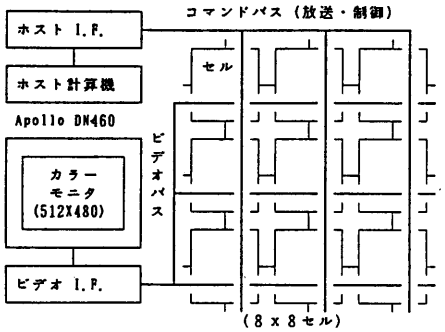


図3-1　CAP の構成

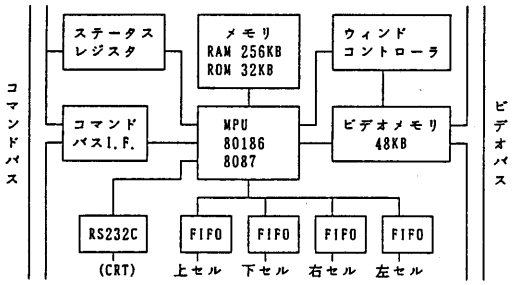


図3-2 セルの構成

3.2 静的負荷分散法

一般に，ある問題を並列処理する場合に，並列処理特有の通信オーバーヘッドや仕事待ちのアイドル等を考慮する必要がある。こ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見込めない。光線追跡法アルゴリズムのＣＡＰへの実現にあたって，各セルが担当する仕事の最小単位を画素単位とした。これは，各画素毎に独立に処理が行えるので通信オーバーヘッドが少なくて済むことを考慮したためである。

次に負荷分散について述べる。負荷の少ないセルは仕事を終えてアイドルの状態になるが，まだ仕事をしているセルもある。セル毎の処理時間のばらつきが少ないことが並列処理においては重要となる。しかし，モデルによって負荷分散が変り，また反射・透過等のため，事前に評価を行って均一に仕事を割り付けるのは困難である。LINKS-1［8］では，これを動的に決定しているが，通信等のオーバーヘッドが起る問題がある。

筆者等が提案する静的負荷分散法では，オーバーヘッドを少なくするために仕事の動的再割り付けをしない。この方法は，スクリーンを 8x8ドットの矩形に分割しておき各セルはこの領域のある位置の画素を担当する。図3-3 でセル0は☆で示された画素を飛び飛びに担当する。

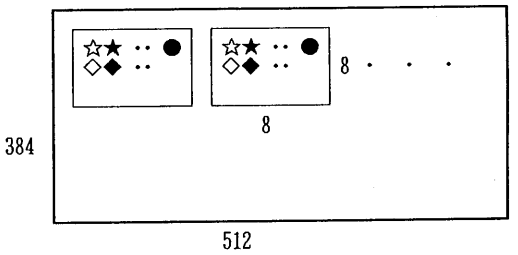


図3-3　　ドットモード

この方式を採用するにあたって，負荷が統計的に平均されると仮定した。第4章でこの方式を使った処理結果について詳細な評価を行うが，ドットモードでは前述した他の担当モードに比べて圧倒的に良い結果を得た。

3.3　インプリメンテーション

今回開発した並列光線追跡法システムの実現法について述べる。基本方針として，

(1)セル間の通信量を抑えるために，総てのセルに同じプログラムとデータを登録する。

(2)セルは自分のIDを知っており，これによってドットモードの矩形領域内の担当画素を決定する。

アンタイエイリアシングについては，サブピクセルの輝度を平均する方式［1］をとっている。まず，光線追跡処理を行い担当画素の輝度をビデオメモリに格納する。次に，

アンタイエイリアシング処理に入る。ここでは，ピクセル四隅の輝度値によってさらに深いレベルまでの動的な分割処理を行う。二つの処理の間には，隣接する画素情報の収集が必要になる。すなわち，あるセルから見ると，自分と下・右・右下のセルに必要な輝度情報がある。ドットモードでは，あるセルが担当する画素の隣接画素は物理的に隣接するセルが担当しているので隣接セル間通信をつかって情報を高速に得ることができる。

右と下のセルからは通信距離1で，右下のセルからは右（または左）のセルを経由して通信距離2で情報を得られる。総てのセルが自分のデータを左と上のセルに送り，さらに下から受け取ったデータを取り込んだのち左のセルに送る。

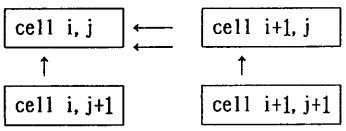


図3-4 アンタイエイリアシング処理時の通信

4．評価

4.1 空間分割による高速化

ここでは，3DDDA を用いた空間分割法のコスト評価，および実験結果の検討を行う。モデルには次の仮定を置く。

(1)プリミティブは空間に一様に分散する。

(2)プリミティブの大きさは voxelのそれよりも平均的に小さい。

つまり，光線は物体に衝突するまで，複数voxel を通過すると考える。この仮定のもとで総ての物体と交差計算を行う従来の方式と，本方式のコストCconvとCvox は，それぞれ

$$C_{conv} = R(RO + C_c \cdot P) + SO \qquad (1)$$

$$C_{vox} = R \cdot m \cdot N(RO + \frac{C_c \cdot P}{N^3} + C_t) + SO \qquad (2)$$

R：光線の総数，N：一次元の分割数，
P：primitive 数，Cc ：交差計算コスト，
Ct ：トラバースコスト（voxel 1個当り），
SO：システムオーバーヘッド，
RO：光線に関する他の処理（光線の傾き・分岐，Shading，CSG 評価等）
m ：平均 traverse 距離係数（m･N は通過 voxel数）

となる。

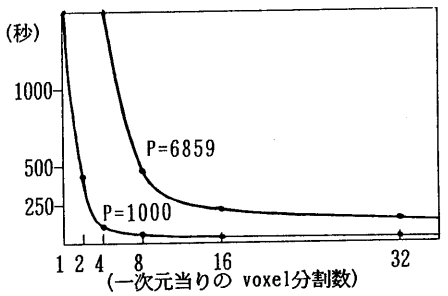
空間分割方式において効率の良い高速化を実現するためには，係数の間にCt ≪Cc の条件が要請される。実験ではその比（Cc ／Ct ）は30である。表4-1 に光線追跡処理の主要な演算コストを示す。

表4-1 CAP とM380の基本演算性能

演算	CAP CELL (usec)	M380 (usec)	M380 / CAP CELL
交差計算 (球)	2980	18.2	164
(円錐)	5783	31.8	182
(箱)	5580	42.2	132
(平均)	4718	27.9	169
3DDDA (SETUP)	4237	33.8	156
(TRAVERSE)	160	1.4	114
matrix * vector	597	3.6	164
matrix * matrix (4次)	3134	18.5	169

表4-2 分割数と処理時間（M380で測定）

物体数	分割数（1次元当り）						
	1	2	4	8	16	32	48
16	102	39	23	20	21	25	30
512	825	212	68	34	28	31	36
1000	1573	450	126	59	38	38	43
4096	13094	2856	678	211	104	96	101
6859	32621	7306	1557	482	219	162	163



グラフ4-1 処理時間（物体数1000と6859）

表4-2 とグラフ4-1 に，前述した仮定を満足するようなモデルを使って得た性能を示す。(2)式でも示されるように，分割数が少ないうちは急激に，多くなると緩やかに処理時間が減少する結果が得られた。また，最適分割数を超えると，トラバースのコストが増加して逆に遅くなる。物体数と高速化率（＝分割数1での時間／最適分割数での時間）の関係をグラフ4-2 に示す。高速化の効果は物体数が多くなると顕著になっており，複雑なモデルでの有効性を示し

ている。

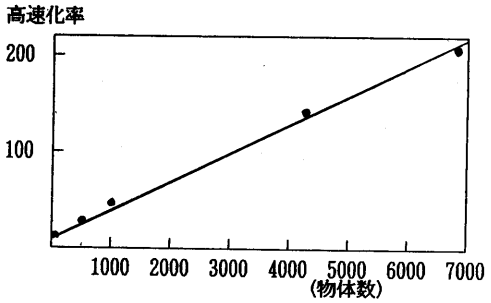


図4-2 物体数と処理時間

処理速度の高速化については，6859個のモデルでは，200倍程加速された。しかし，実験に使ったモデルでは少ない物体数と空間非均一性のため，数十倍程の高速化に留まっている。処理の内容を調べるために，モデル "PISTON" の分割数に対する各処理の割合とコストを表4-3 に示す。このモデルの部品数は 179個で，高速化率は12倍である。

表4-3 処理のコスト・割合　(時間単位は秒)

処理	2分割	16分割	32分割
	コスト(%)	コスト(%)	コスト(%)
交差計算	108.5 (84.4)	11.4 (34.2)	7.8 (23.0)
3DDDA (含setup)	3.4 (2.7)	5.1 (15.4)	6.9 (20.4)
CSG 評価	4.9 (3.8)	4.5 (13.4)	6.0 (17.6)
Voxel 生成	0.2 (0.1)	0.5 (1.6)	2.3 (6.6)
他の処理	5.5 (4.2)	5.8 (17.4)	5.1 (14.8)
処理時間	128.5	33.4	34.1

(2)式からは，トラバースコストが小さいと大きな分割数で最適となって処理時間が少なくなると予想される。しかし，上のデータから分るように，このモデルの最適分割数(16)では，交差計算の処理時間は十分小さくなり，更なる分割での大幅な減少は見込めない。一方，トラバースコストは線形に増加するが，これよりも voxel生成時間の増加率が大きい。また，光線の傾き，分岐等の光線操作・シェーデング・3DDDA の初期計算等の処理時間は一定なので，処理が高速になるにつれて，処理時間に対する割合は大きくなる。以上の議論から，単に 3DDDAの速度を速くしてもあまり効果が見込めないことが分る。

4.2 静的負荷分散の評価

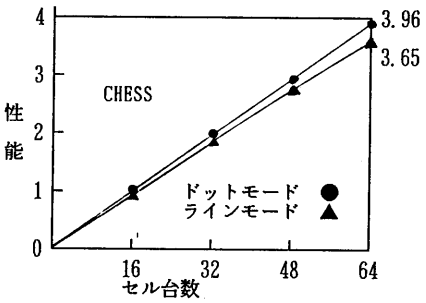
表4-4 に幾つかのモデルについての実験結果を示す。ドットモードでは，処理時間のばらつきは数パーセント以内に収まっている。

表4-4　表示処理時間（セル64台）

モデル (物体数)	ラインモード Max — Min (d)	ドットモード Max — Min (d)
BALLS (125)	93—84 (9.7)	88—86 (2.3)
CHESS (143)	324—259 (20.1)	294—281 (4.4)
CAD (185)	792—637 (19.6)	691—667 (3.5)

Max: 最も遅いセルの処理時間 (秒)
Min: 最も速いセルの処理時間 (秒)
d : 最大偏差
d=(Max-Min)/Max *100 (%)

グラフ4-3 は，モデル "CHESS"をドットモードとラインモードで表示した場合の台数効果を表す。16台での性能を1としている。ドットモードでは64台で3.96の性能が得られた。つまり，台数の増加に対して，ほぼ線形に性能が向上することが分る。



グラフ4-3　　台数効果

表4-5 にモデル"DISE"についてのアンタイエイリアシング処理における性能を示す。通常の静的負荷分散法では，物体のエッジ等でサブピクセル単位での光線追跡を行うので局所的に負荷の高い部分が生じ，結果として負荷分散が

悪くなる傾向がある。そこで処理単位を，１フレームとせずに動画への適用を考慮して複数フレームとする。この際，前述したドットモードのパターンをフレーム毎に変え，フレーム毎にセルは異なる画素領域を担当するようにする。

この方式によれば，時間軸に対しても負荷が平均されることが期待できる。このフレーム平均化された静的負荷分散方式の評価結果は，表4-5 から分るように単なる静的負荷分散方式の動画適用に比べて，そのばらつきは減少していることが分る。

表4-5 フレーム平均された静的負荷分散方式

処　　理	処理時間	最大偏差 (%)
非アンタイ	1336	6.5
アンタイ	4626	60.4
フレーム平均アンタイ	2595	8.8

(モデル :DISE, 64フレーム処理合計時間 (秒)

5. まとめ

レイトレーシングアルゴリズムによる映像生成の高速化を目的として，ＣＡＰによるレイトレーシングの並列処理を実現し，性能評価を行った。

(1)空間分割による光線追跡法の高速化（十～数百倍）を実現し，提案したパラメトリック3DDDA の有効性を確認した。

(2)効率の良い CSG評価処理を行うために，状態木法を部分空間毎に逐次的に適用する部分評価法を提案した。

(3)静的負荷分散法によりオーバーヘッドの少ない良好な分散効果を得た。さらに，動画生成を考慮したフレーム平均化静的負荷分散方式の評価を行った。

表5-1 各種モデルによるＣＡＰとM380の処理時間

モデル (物体数)	ＣＡＰ (秒)	Ｍ３８０ (秒)
CHESS (144)	１５５	１２８
BALLS (125)	７２	６０
PISTON (179)	４０	３３
CAD (178)	２０	２０

表5-1 から分るように，いくつかのモデルの比較では，ＣＡＰ64台システムと汎用計算機のM380では，同程度の性能が得られた。さらに，上の評価から分るようにＣＡＰ上の並列光線追跡システムはセル台数の増加に伴って直線的な性能向上が期待でき，コストパフォーマンスのよい映像生成システムが構築出来ると考えられる。また，専用計算機では，CPU 時間と処理時間が等しいので，速い応答が要求される会話型システムに向いていると言える。

今後の課題として，現在ホスト計算機で行っているvoxel 生成を，ＣＡＰで並列処理し，トータルな性能改善を図る必要があると考えている。

[謝辞] 本研究に当り議論等で御援助して下さいました石井部長代理に感謝します。

[参考文献]

[1] Whitted, T:"An Improved Illumination Model for Shaded Display", CACM Vol.23, No.6, 1980
[2] 村上:"オクトリーデータ構造による光線追跡法", 情報処理学会第27回全国大会後期 , 1983
[3] A.S.Glassner:"Space Subdivision for Fast Raytracing", IEEA, CG&A, Oct., 1984
[4] 藤本:"レイトレーシングの高速化技法について", Pixel, No.37, Oct., 1985
[5] Requicha, A., Voelcker, H.:"Solid Modeling:Current Status and Research Direction", IEEE, CG&A, Vol.3, No.7, 1983
[6] 村上:" CSG表現からの画像生成方式", 情報処理学会第27回全国大会後期 , 1983
[7] 村上:"Stacked Status Tree によるRay-tracing", 60年度電子通信学会情報・システム部門全国大会, 1-210
[8] 西村:"コンピュータグラフックスシステムLINKS-1に於ける画像合成の高速化手法", 情報処理学会論文誌, Vol25, No.6, Nov, 1984
[9] 佐藤:"セルラアレイプロセッサＣＡＰのアーキテクチャ", 信学技報, Vol25, No282, CAS84-200, 1985
[10] Roth, S.D.:"Ray Casting for Modeling Solids", CG&IP, No.18

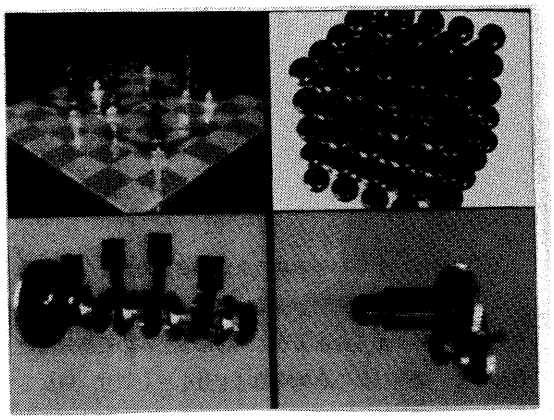


図5-1 CAPで生成された画像の例
(CHESS, BALLS,
PISTON, CADのモデル)