グラフィクスとCAD 77－2
（1995．11．9）

# ピクセルレベルにおける並列図形処理アルゴリズム

荒田 秀樹† 高井 昌彰†† 佐藤 義治†
†北海道大学工学部, ††北海道大学大型計算機センター

ピクセル空間に幾何図形を描画する場合、ピクセル空間を受動的なメモリ機構と見なし、描画をそれに対する手続き的処理ととらえることが多い。

これに対し、それぞれのピクセルを能動的な実体として考え、幾何学的制約のソルバーとピクセル空間とを統合化しようとする試みがある。

本論文では並列計算モデルであるセル構造オートマトンを用いて、幾何図形の生成・変換を実現する並列図形処理アルゴリズムを議論する。幾何学的制約は目的関数の最大化問題として表し、ピクセルレベルにおける制約緩和問題として扱う。

# A Parallel Algorithm for Generation and Transformation of Geometrical Images at Pixel Level

Hideki Arata†, Yoshiaki Takai††, and Yoshiharu Sato†
†Faculty of Engineering, Hokkaido University,
††Computing Center, Hokkaido University

There are many cases that we think the pixel space as passive one and painting pictures as step by step. In this paper we think each pixel as an active object, and try to genarate and transform geometrical images using geometrical constraint solver above with pixel space.

We discuss a parallel algorithm for generation and transformation of geometrical images using cellular automata. Geometrical constraint solving is realized by maximization of objective functions at pixel space level.

# 1 はじめに

ピクセル空間に幾何図形を描画する場合、端点座標や線分の長さなどの情報を一括して集中管理する何らかの上位機構が存在し、その制御の下に描画処理が進められていく場合が多い。この際、ピクセル空間は受動的な下位のメモリ機構と考えられ、図形描画はピクセル空間に対する一方的なデータ書き出しということができる。

この2次元機構は、グラフィクスエンジンのアーキテクチャの基本となるばかりでなく、ユーザのシステムビューを含む描画モデル全体の論理的な構造に影響を及ぼしており、手続き的な描画手法との親和性が強い。

一方これに対し、ピクセル空間そのものに能動的機能を与え、その中に本質的な描画機構を埋め込むアプローチが考えられる。すなわち、各ピクセルを自律的な処理要素としてとらえ、それらの間で局所的な情報のやりとりを通して、描画空間であるピクセル空間全体で直接幾何図形の生成や変換を実現する。いわば「能動的キャンバス」をシステムビューとして提供するものである。このような描画モデルにおいては、手続き的方法よりも、むしろ宣言的描画手法が自然である。すなわち、ピクセル空間は単なるメディアではなく、幾何学的制約のソルバーとして機能するわけである。

以上の観点に立ち、我々は幾何学的制約に基づく会話型図形生成システムにコネクショニズムの原理を適用する研究を行ってきた[1, 2, 3, 4]。Neuropadは、1つの幾何制約を透明なパッドとして表現し、基礎となる図形の上に必要なパッドを多数重ねることで必要とする図形を生成する[3]。各パッドには、高次エネルギー関数を持つ拡張ボルツマンマシンが対応付けられており、制約のソルバーとして機能する。しかしこのソルバーは、座標値を直接最適化するものであり、描画空間であるピクセル空間とは別に存在する。

そこで我々は、局所的並列計算モデルであるセル構造オートマトン[5]に着目し、この上で幾何学的制約に基づく図形生成を直接実現することを試みてきた[6, 7]。これは、ピクセル空間と制約のソルバーをセル構造オートマトンの枠組の中で融合しようとするものである。セル構造オートマトンを用いてテクスチャを発生させるような研究[8]は従来から行われているが、幾何図形の生成そのものに関する研究は少ない。

本論文では、セル構造オートマトンを用いて、幾何制約に基づく図形生成ならびに図形の変換・移動を実現する並列図形処理アルゴリズムについて述べる。次に本アルゴリズムの適用例をいくつか示し、最後に今後の方針及び課題について述べる。

# 2 ピクセル空間図形処理モデル

本モデルでは、ピクセル空間を多層の2次元セル構造オートマトンと見なし、その上で幾何学的制約に基づく図形生成を実現する。幾何学的制約は目的関数の最大化問題として表現され、ピクセルレベルで直接処理される。

一方、生成された図形の変換・移動はピクセル空間上の物理的な運動としてとらえ、その運動を制御するためのベクトル場を定める[9]。このベクトル場を用いてセル構造オートマトンを間接的に制御することにより変換・移動処理を実現する。

## 2.1 セル構造オートマトンとは

$d$次元無限格子からなる空間（セル空間）を考える。この格子点に同一の有限オートマトンを用意する。各有限オートマトンは現在の自分の状態と、予め定義されている近傍の状態から、局所写像により次の時刻における自分自身の状態を決定する。この遷移はセル空間全体にわたって同期して並列に行われる。ある時刻のセル空間全体の状態を様相と呼び、現在の様相から次の時刻の様相を定める写像を並列写像という。

セル構造オートマトンは形式的には4項組$(Z^d, Q, X, f)$で定義される。$Z$は整数の集合、

$Z^d$は $d$ 次元のセル空間である。$Q$ は基本となる有限オートマトンの状態集合であり、$X = \{z_1, z_2, \ldots, z_n \mid z_i \in Z^d\}$ は近傍形、$f$ は $Q^n$ から $Q$ への局所写像である。ある時刻 $t$ の様相 $c$ は $Z^d$ から $Q$ への写像であり、次の時刻の様相 $c'$は任意の $z \in Z^d$ に対して次のように定まる。

$$c'(z) = f(c(z+z_1), c(z+z_2), \ldots, c(z+z_n))$$

様相 $c$ 全体の集合を $C$ とすると、$C$ から $C$ への並列写像Fは、近傍形と局所写像から一意に定まる。

## 2.2 生成アルゴリズム

ピクセル空間を図1のような多層の2次元セル構造オートマトンと見なす。ユーザは端点数などの幾何学的制約条件を与え、セル構造オートマトンの局所写像と初期様相を定める [6, 7]。

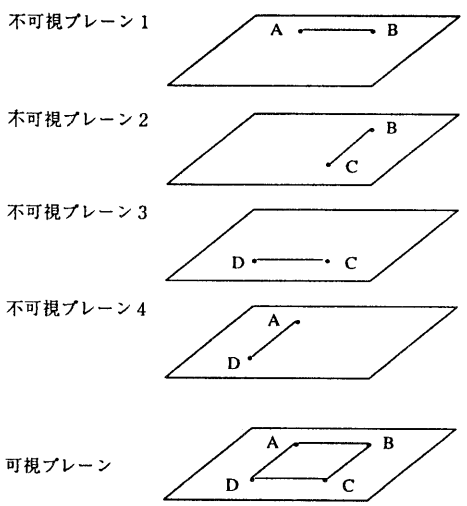


図 1: 多層の 2 次元セル構造オートマトンによる図形生成（正方形 ABCD の生成例）

生成アルゴリズムでは、幾何学的制約を目的関数の最大化問題として扱う。このことは、本モデルが、機械製図などのように厳密解を求めるものではなく、幾何学的曖昧さをある程度許すような制約緩和システムであることを意味する。したがって、不十分または冗長な制約に対しても、ある程度の対処が可能になる。

具体的には端点発生、線分発生の 2 つのフェーズにより幾何図形を生成する。以下に生成アルゴリズムの詳細について説明する。

### 2.2.1 セルの状態

セルのとり得る状態と属性は、基本的に参考文献 [7] に基づいている。それぞれのセルは、(1) 端点、(2) 端点候補、(3) 情報の伝搬中、(4) そのいずれでもない、の 4 つの状態をとる。また属性として、(1) 自分自身の座標、(2) セル固有の番号 (ID)、(3) 制約の充足度を持ち、他の端点候補の座標・ ID ・制約の充足度を保持するための領域を有する。

### 2.2.2 幾何学的制約条件

線分の長さや平行・直交などの制約条件を目的関数の最大化（最小化）問題として表し、最大化問題を解くことにより条件を満たすような端点を生成する。例えば、線分の長さ、2 線分の平行・直交、角度を表す目的関数はそれぞれ以下のようになる。

・長さ $l$ の線分 PQ

$$f_{len}(PQ, l) = -((x_P - x_Q)^2 + (y_P - y_Q)^2 - l^2)^2$$

・線分 PQ と線分 RS が平行

$$\begin{aligned} f_{para}(PQ, RS) &= -((x_P - x_Q)(y_R - y_S) \\ &\quad -(x_R - x_S)(y_P - y_Q))^2 \end{aligned}$$

・線分 PQ と線分 RS が直交

$$\begin{aligned} f_{orth}(PQ, RS) &= -((x_P - x_Q)(x_R - x_S) \\ &\quad +(y_P - y_Q)(y_R - y_S))^2 \end{aligned}$$

・線分 PQ と線分 RS のなす角 $\theta$

$$\begin{aligned} f_{angle}(PQ, RS, \theta) &= -(U - V cos(\theta))^2 \\ U &= \overrightarrow{PQ} \bullet \overrightarrow{RS} \\ V &= |\overrightarrow{PQ}||\overrightarrow{RS}| \end{aligned}$$

個々のセルは、セル構造オートマトンに与えられる幾何学的制約条件を参照して、自分自身の制約の充足度を、上のいずれか、もしくは複数の式によりそれぞれ独立に計算する。この各セルにおける制約の充足度は、次に述べる端点発生アルゴリズムで使われる。

### 2.2.3 端点発生

端点の位置座標は、ユーザから明示的に与えられない限りピクセル空間上で任意である。したがって、2 次元のピクセル空間から任意に 1 つのセルを選び出す機能が必要になる。これは、分散システムにおけるリーダ選出問題と同様である。

図 2 は、端点発生の概念図である。それぞれのセルは、個々で独立に計算された制約の充足度を持っている。この制約の充足度と、ID、自分自身の座標値を近傍に伝搬させていく。異なる端点候補の情報が衝突した場合は、充足度の大きいセルの情報を更に伝搬させる。充足度が同じ場合は ID を比較する。この手順を繰り返すことにより、制約を満たす端点を発生させることができる。

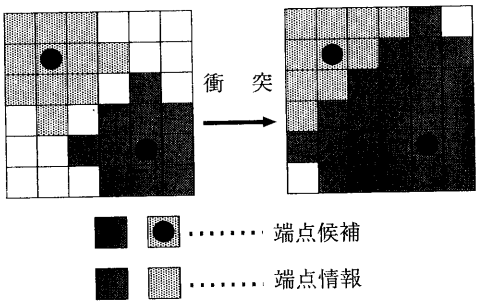


図 2: 情報伝搬による端点発生

### 2.2.4 線分発生

先の方法で端点を発生させた後には、すべてのセルに端点の座標値が伝搬している。この座標値を利用して線分を発生させる。

一般に、2 点 $(x_1, y_1)$、$(x_2, y_2)$ を通る直線の式は以下のように表される。

$$(y - y_1)(x_2 - x_1) = (x - x_1)(y_2 - y_1)$$

この事実を利用して、各セルで、自分自身がどれだけ直線の式を満たしているかを表す充足度を計算する。セルの座標を $(x, y)$ とすると、充足度 $f$ は以下のように与えられる。

$$f = ((y - y_1)(x_2 - x_1) - (x - x_1)(y_2 - y_1))^2$$

この充足度とあらかじめ定めた閾値により、任意の 2 点を結ぶ線分を発生させることができる。また、制約条件により結びたい端点を指定し、その線分のみを発生させることもできる。

## 2.3 変換・移動アルゴリズム

図形の変換・移動をピクセル空間における物理的な運動としてとらえる。この運動を規定するためにベクトル場を導入し、セル空間を間接的に制御する。図 3 のように、1 つのセルは、セル空間における自分の近傍の状態とその位置のベクトル場を参照して、次の時刻の自分自身の状態を決定する。

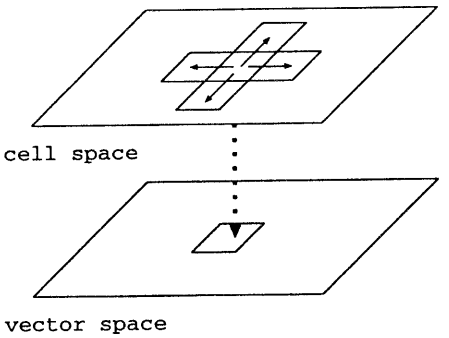


図 3: 変換・移動のためのベクトル場

### 2.3.1 ベクトル場の定義

図形変換の種類に応じて適切なベクトル場を用意する。このベクトル場は、セル空間とは別に存在する上位機構により発生させる。ピクセル空間の座標 (i,j) におけるこのベクトル

を $\mathbf{p}_{i,j} = (p_x, p_y)$ で表すことにする。また、セル構造オートマトンの性質上、ピクセルは1ステップで最大1セル分しか移動できないことを考慮して、$-1 \leq p_x, p_y \leq 1$ の制限を設ける。

### 平行移動

平行移動はその方向と距離を示す定数ベクトルから成る場を用いて表現される（図4参照）。一般に $x$ 軸方向に $m_x$ 、$y$ 軸方向に $m_y$ の平行移動に対して、ベクトル場 $\mathbf{p}_{i,j} = (p_x, p_y)$ を以下のように定義する。

$$p_x = Cm_x \quad , \quad p_y = Cm_y$$

ここで C は正定数であり、セル構造オートマトンの状態遷移回数と実際の移動量の比を定める。

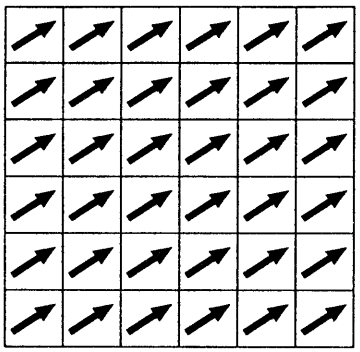

図 4: 平行移動のベクトル場

### 回転移動

回転移動では、回転の中心を定め、図5のような渦状のベクトルを用意する。各位置のベクトルの大きさは、中心 $(i_0, j_0)$ からの距離と回転の角度$\theta$ に比例する。C を正定数として、ベクトル場を以下のように定義する。

$$p_x = -C(j - j_0)\theta \quad , \quad p_y = C(i - i_0)\theta$$

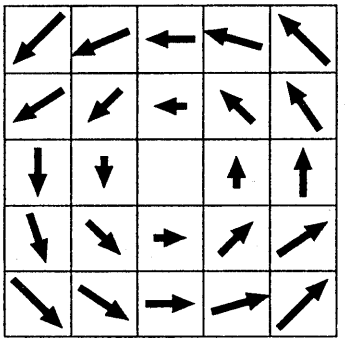

図 5: 回転移動のベクトル場

### 拡大・縮小

拡大・縮小では、相似の中心 $(i_0, j_0)$ を定めておく。このとき、ベクトル場は次のように定義できる。

$$p_x = \beta C(i - i_0) \quad , \quad p_y = \beta C(j - j_0)$$

C は正定数、$\beta$ は拡大率（縮小率）で、$\beta > 0$ のとき拡大、$\beta < 0$ のときは縮小を表す。ベクトル場の様子を図6に示す。

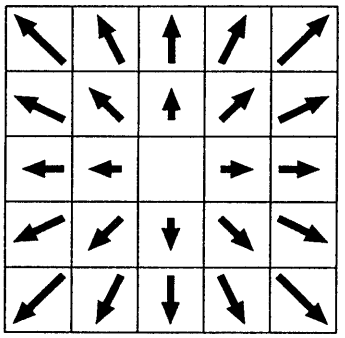

図 6: 拡大・縮小のベクトル場

### 2.3.2 セルの属性

変換・移動アルゴリズムでは、先に定義したベクトル場にしたがい図形を移動させるため、x 軸、y 軸方向への積算値 $\mathbf{S}_{i,j} = (s_x, s_y)$ を考え、セルの属性とする。セル構造オートマトンの性質上、積算値も、$(-1 \leq s_x, s_y \leq 1)$ とする。

### 2.3.3 ベクトル場による図形の変換・移動

ピクセル空間上の図形を質点（粒子）の集合と考え、これらの運動をベクトル場によって制御し、図形変換を実現する。

幾何図形を構成する各粒子はその位置のベクトル場を参照し、各状態遷移ごとに積算値 $\mathbf{S}'_{i,j}$ を以下の式にしたがって繰り返し計算する。

$$\mathbf{S}'_{i,j} = \mathbf{S}_{i,j} + \mathbf{p}_{i,j}$$

実際の粒子の移動は、どちらかの成分の積算値の絶対値が 1 を超えた場合に、その符合方向に移動する。移動後、積算値に 1 を足すか、もしくは積算値から 1 を引く。

## 3 実行例

先に説明したアルゴリズムにより得られた幾何図形の生成例と変換・移動例を以下に示す。セル空間の大きさはすべて 256 × 256 である。

### 3.1 生成例

本モデルにより生成した幾何図形の例をいくつか示す。図 7 はそれぞれ正三角形、正方形を生成したものだが、正三角形では $\{AB = BC = CA = 120\}$ 、正方形では $\{AB = BC = CD = DA = 120, AB \perp BC\}$ の幾何学的制約を与えた。更に正五角形と正六角

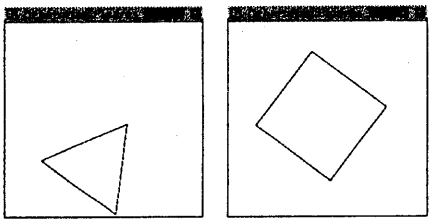

図 7: 正三角形と正方形

形を生成した例と設定した幾何学的制約をそれぞれ図 8 、図 9 に示す。

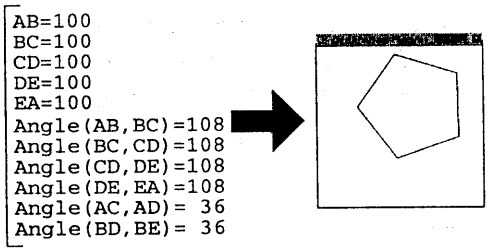


図 8: 正五角形

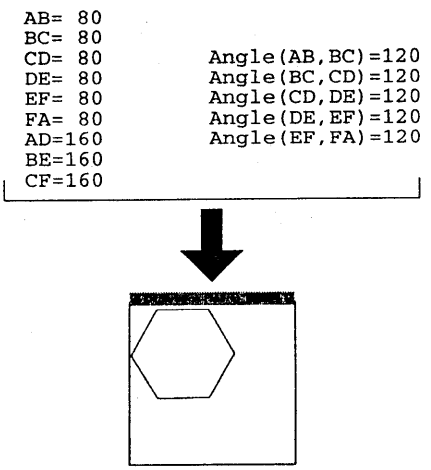


図 9: 正六角形

### 3.2 変換例

幾何図形にいくつかの変換を施した例を次に示す。なお、変換例に用いた四角形は、事前にあらかじめ用意したものである。

図 10 に、四角形を平行移動した様子を示す。この例は、$x$ 軸方向に 4、$y$ 軸方向に 3 の方向に平行移動したものである。移動量は、セル構造オートマトンの状態遷移回数（ステップ数）に比例する。

図 11 は、回転の中心をセル空間の中心として、四角形を半時計方向に回転させたものである。回転後の図形にゆがみが生じるのは、粒子運動の量子化による誤差が蓄積するためと考えられる。

相似の中心をセル空間の中心とし、四角形を拡大させたのが図 12 、縮小させたのが図 13

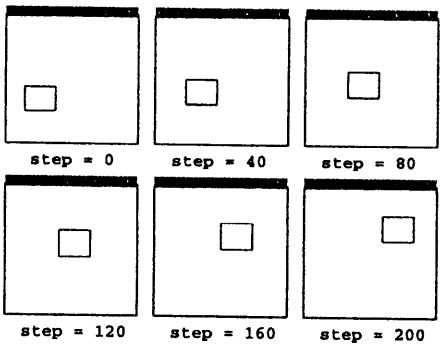


図 10: 平行移動

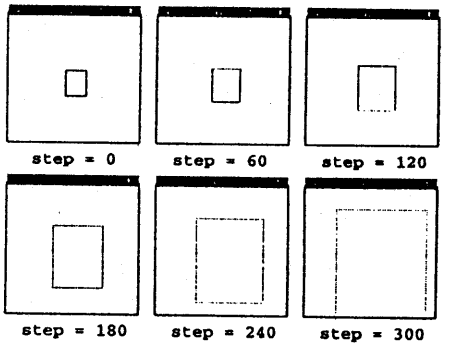


図 12: 拡大

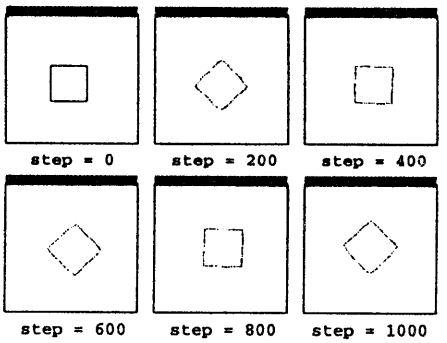


図 11: 回転移動

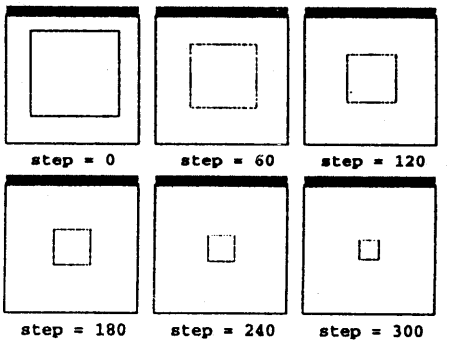


図 13: 縮小

である。

## 4 おわりに

ピクセル空間を 2 次元のセル構造オートマトンと見なし、制約条件を考慮に入れた幾何図形の生成や、平行移動などの変換処理を実現する並列図形処理モデルについて述べた。

今後の課題としては、本モデルに基づく幾何図形の生成、変換・移動の機能を統合化した対話型図形エディタのインプリメント、及び本モデルが扱うことのできる幾何学的制約の範囲の特定、すなわちモデルの般化性の検証などが挙げられる。

## 参考文献

[1] N.Kin, Y.Takai and T.L.Kunii: "PictureEditorII : A conversational graphical editing system considering the degree of constraint", *Proc. of Computer Graphics Internatinal '92*, pp. 711–730 (1992).

[2] N.Kin, Y.Takai and T.L.Kunii: "Geometrical constraint solving based on the extended boltzmann machine", *Computers in Industry*, Vol. 19, No. 2, pp. 239–250 (1992).

[3] Y.Takai and N.Kin: "Neuropad : A geometric constraint solver based on the connectionist architecture", *Proc. of the Fourth ISMM/IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 404–407 (1991).

[4] 高井昌彰, 金那美: “幾何学的制約緩和問題のためのコネクショニスト・アーキテクチャ”, 信学技報, Vol. 90, No. 173, pp. 123–130 (1990).

[5] T. Toffoli and N. Margolus: *Cellular Automata Machines*, MIT Press (1987).

[6] 荒田秀樹, 高井昌彰, 佐藤義治: “セル構造オートマトンによる図形生成”, 第 8 回札幌国際コンピュータグラフィックスシンポジウム論文集, pp. 33–38 (1994).

[7] 荒田秀樹, 高井昌彰, 佐藤義治: “幾何図形のピクセルレベルにおける並列生成”, 情報処理学会第 50 回全国大会講演論文集, Vol. 2, pp. 329–330 (1995).

[8] R. Fisch: “Cyclic Cellular Automata and Related Process”, *Physica*, Vol. 45D, pp. 19–25 (1990).

[9] Y. Takai, K. Ecchu and N. K.Takai: “A cellular automaton model of particle motions and its applications”, *The Visual Computer*, Vol. 11, No. 5, pp. 240–252 (1995).