社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－134／2008－EMB－8 (13)
2008 / 3 / 27

Two-cycleレベルセット法による輪郭抽出処理のハードウェア化

米澤　遼†　本田　郁二†

† 慶應義塾大学大学院 理工学研究科
〒 223-8522 神奈川県横浜市港北区日吉 3-14-1
E-mail: †yonezawa@rt.k2.keio.ac.jp, ††honda@elec.keio.ac.jp

あらまし　画像の輪郭抽出は，画像処理問題において重要な要素である．輪郭抽出を必要とする分野として，モバイル製品などの組込み機器が挙げられる．組込み機器では，PCのようにプログラムによる実装をするほどの計算機の資源がなく，ハードウェアでの実装が好ましい．輪郭抽出の代表的なアルゴリズムに動的輪郭モデルが挙げられ，さらにそれはスネーク法とレベルセット法に分けることができる．レベルセット法は，スネークでは対応できないトポロジー変化に対する自動的な処理が可能な反面，計算コストが高いという特徴がある．本論文では，レベルセット法をさらに高速にしたTwo-Cycleレベルセット法をverilog記述によりシミュレーションすることにより，ハードウェア化した際の回路規模やコスト，またハードウェア化する妥当性を検証する．
キーワード　画像処理, 輪郭抽出, 動的輪郭モデル, ハードウェア

Hardware implementation of contour definition by two-cycle levelset algorithm

Ryo YONEZAWA† and Ikuji HONDA†

† Graduate School of Science and Technology, Keio University
3-14-1 Hiyoshi, Kouhoku-ku, Yokohama, Kanagawa 223-8522 Japan
E-mail: †yonezawa@rt.k2.keio.ac.jp, ††honda@elec.keio.ac.jp

Abstract　The contour definition of an image is an important element in an image-processing problem. The contour definition is needed by embedded systems, such as mobile products. Embedded systems are no resources for a computer to implemented by a program like PC, but implemented by hardware is suitable. A dynamic contour model is listed to the typical algorithm of a contour definition, and it can be further divided into the Snake method and the Level-set method. While the automatic processing to topology change which cannot respond in Snake is possible for the Level-set method, it has the feature that calculation cost is high. In this paper, the simulation of the Two-Cycle Level-set method which made the Level-set method still more nearly high-speed is carried out by verilog description. The circuit scale at the time of hardware-izing, and cost and the hardware-ized validity are verified.
Key words　image-processing, contour definition, active contour model, hardware

1. はじめに

1.1 背景と目的

画像から対象とする物体の輪郭を抽出することは，画像認識において重要な問題である．代表的な領域抽出法の一つに動的輪郭モデル (Active Contour Model) が挙げられ，これは画像内に配置した曲線 (輪郭と呼ぶ) を物体形状を復元するために変形させる方法である．動的輪郭モデルは大まかに分けると，スネーク (Snakes) と呼ばれるモデルとレベルセット法によって実行されるモデルの二つの方法に分かれる．

スネークはパラメータを用いて表される閉曲線を定義した“力”によって領域境界に収束させていく方法で，Kassら [1] によって提案されてから幅広い応用分野で使われ，様々な改良が行われてきた．

一方，本論文で扱うレベルセット法による動的輪郭モデルはより後にCaselles ら [2] と Malladi ら [3] によって同時に提案され，輪郭発展の間に起こるトポロジー変化に対する解決策として受け入れられてきた．それはOsher と Sethian [4] が提案したレベルセット法と呼ばれる境界面追跡法を画像に応用することで実行され，輪郭を1次元高い補助関数のゼロ等高面に埋

め込み，その関数を画像特性を考慮した速度関数で発展させることでゼロ等高面を変化させて輪郭発展を行う方法である．

レベルセット法はは偏微分方程式を解くために，スネークよりも計算量が多くなってしまい，遅いという問題がある．

今回使う Two-cycle レベルセット法（以降 Two-cycle 法と呼ぶ）は，Yonggang ら [5] によって提案された．従来のレベルセット法よりも格段に計算範囲が狭まり，また，補助関数の発展に計算量を多くしていた偏微分方程式を使わないことで，従来よりも速い結果が出ている．しかし，ソフトウェア上でレベルセット法を用いて輪郭抽出を行うとフレームレートでの処理が行えず，リアルタイムに処理を行うことができない．

本論文では，Two-cycle 法を verilog で記述することでハードウェア化した際の回路規模やコスト，ハードウェア化する妥当性を検証する．また，妥当性の指標として，リアルタイムに動作することを目標とする．

1.2 本論文の構成

本論文は次のように構成される．第 2 章で動的輪郭モデルであるレベルセット法の説明をし，次の第 3 章ではレベルセット法の画像に適用した最初の形式について述べる．第 4 章では本論文でも用いる two-cycle 法の説明をし，従来のレベルセット法との比較を行う．第 5 章ではハードウェア化した際の構成を示した．第 6 章で実際に画像に適用した有用性および検討を行う．最後に第 7 章で結論と今後の展望を述べる．

2. レベルセット法

2.1 輪郭伝播理論

時刻 $t \geq 0$ での輪郭 $\mathbf{C}(t) = \{\mathbf{C}(s,t); 0 \leqq s \leqq 1\}$ に対し，点 $\boldsymbol{x}$ が初期の輪郭 $\mathbf{C}(t=0)$ の外側ならば正，内側ならば負となるような一次元高い符号付距離関数

$$\Phi(\boldsymbol{x},t) = \begin{cases} d(\boldsymbol{x}) & \text{点}\boldsymbol{x}\text{が輪郭 } \mathbf{C}(t) \text{ の外側} \\ 0 & \text{点}\boldsymbol{x}\text{が輪郭 } \mathbf{C}(t) \text{ の上} \\ -d(\boldsymbol{x}) & \text{点}\boldsymbol{x}\text{が輪郭 } \mathbf{C}(t) \text{ の内側} \end{cases} \tag{1}$$

をつくる．この $\Phi(\boldsymbol{x},t)$ をレベルセット関数と呼ぶ．(1) 式の $d(\boldsymbol{x})$ は点 $\boldsymbol{x}$ から輪郭 $\mathbf{C}(t=0)$ までの最短距離である．輪郭 $\mathbf{C}(t)$ とレベルセット関数 $\Phi(\boldsymbol{x},t)$ との関係は $\mathbf{C}(t)$ は $\Phi(\boldsymbol{x},t)=0$ の集合となっている．

$$\mathbf{C}(t) = \{\boldsymbol{x} \mid \Phi(\boldsymbol{x},t) = 0\} \tag{2}$$

上式の輪郭 $\mathbf{C}(t)$ に一致する条件 $\Phi(\boldsymbol{x}(t),t)=0$ を両辺について時間微分すると，

$$\frac{\partial \Phi(\boldsymbol{x}(t),t)}{\partial t} + \nabla\Phi(\boldsymbol{x}(t),t)\cdot\frac{\partial \boldsymbol{x}(t)}{\partial t} = 0 \tag{3}$$

となる．ここで，法線方向の移動速度（成長速度）F は単位法線ベクトル

$$\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{x}(t)) = \frac{\Phi(\boldsymbol{x}(t),t)}{\mid \Phi(\boldsymbol{x}(t),t) \mid} \tag{4}$$

を使って

$$F(\boldsymbol{x}(t)) = \boldsymbol{n}\cdot\frac{\partial \boldsymbol{x}(t)}{\partial t} \tag{5}$$

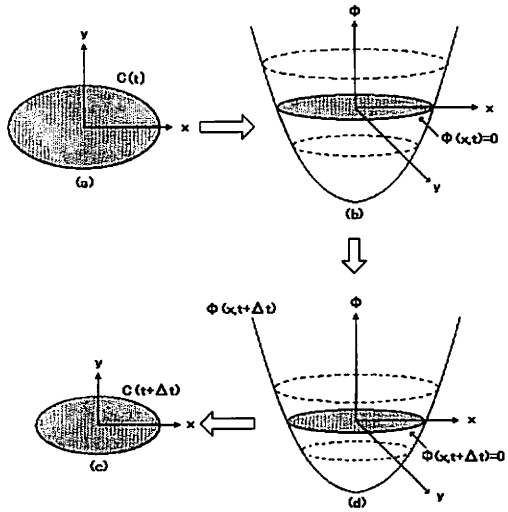

図 1 レベルセット関数の例

と表せ，これを (3) 式に代入すると，

$$\frac{\partial \Phi(\boldsymbol{x}(t),t)}{\partial t} = -F(\boldsymbol{x}(t))\cdot \mid \nabla\Phi(\boldsymbol{x}(t),t) \mid \tag{6}$$

となる．この (6) 式をレベルセット方程式という．初期輪郭 $\mathbf{C}(0)$ と成長速度を与えて (6) 式を解くことによって輪郭を抽出する．

2.2 離 散 化

格子点 ij で h 間隔の均一の網目の場合，時間微分の前進差分を用いて (6) 式は

$$\frac{\Phi_{ij}^{n+1} + \Phi_{ij}^{n}}{\Delta t} + (F)(\nabla_{ij}\Phi^{n} ij) = 0 \tag{7}$$

のように書ける．ここで Δt は time step であり，n は時間である．レベルセット法は，(7) 式を解くことによってレベルセット関数を求め，輪郭を抽出する．

2.3 再 初 期 化

レベルセット関数は (6) 式の更新方程式によって時間を進めると，符号付距離関数としての性質が失われる．そこで，定期的にレベルセット関数を符号付距離として定義し直す“再初期化”処理が必要となる．再初期化の最も一般的な方法は，現在のゼロレベルセットから各格子において距離計算を行うことである．しかし，これは各格子点の最も近いゼロレベルセット上の点を見つけなければならないので，計算コストが高い．そこで，Sussman ら [6] によって提案された次のような再初期化方程式がよく使われる．

再初期化方程式

時刻 $t=0$ でレベルセット関数 $\Phi(\boldsymbol{x},0)$ が定義されると，Φ には距離関数としての等高線のような性質

$$\mid \nabla\Phi(\boldsymbol{x}(t),t) \mid = 1$$

が備わっている．

更新によって失われたこの距離関数としての性質を回復させるために

$$\frac{\partial \Phi(\boldsymbol{x}(t),t)}{\partial t} = \mathrm{sign}(\Phi(\boldsymbol{x}(t),t))(1-|\nabla\Phi(\boldsymbol{x}(t),t)|) \qquad (8)$$

(sign は Φ の符号を与える関数) という式を使う．この式は，各点で勾配 $|\nabla\Phi|$ が 1 から外れた分をエラーとして加えることを意味している．これによって，ゼロレベルセットを明確に見つけずに再初期化することができる．

3. レベルセット法による動的輪郭モデル

3.1 基本的な考え方

前章で述べたレベルセット法を次のように画像に適用する．境界面を画像内に配置した輪郭 (曲線) と考え，それに対して全ての画像領域にレベルセット関数を作る．そして，その関数を輪郭の代わりに画像情報を考慮した成長速度によって更新してゼロレベルセットを変形させることで輪郭発展を行う．つまり，パラメータで定義された輪郭 $\mathbf{C}(s,t)$ をレベルセット関数 $\Phi(\boldsymbol{x},t)$ のゼロレベルセットに埋め込むことによって，輪郭発展を表す (3) 式

$$\frac{\partial \mathbf{C}(s,t)}{\partial t} = F(\mathbf{C}(s,t))\boldsymbol{n}(\mathbf{C}(s,t)$$

をレベルセット関数の動き

$$\frac{\partial \Phi(\boldsymbol{x})}{\partial t} = -F|\nabla\Phi(\boldsymbol{x})|$$

に置き換えることができる．

3.2 Malladi らの手法

レベルセット法を用いた動的輪郭モデルは近年数多く提案されているが，ここでは Malladi ら [3] が提案した最初の形式について説明する．そして，その形式を次章で改良する．

3.2.1 成長速度の設定

輪郭上の任意の点 $\boldsymbol{x}$ における成長速度を

$$F(\boldsymbol{x}) = g(\boldsymbol{x})(a+b\kappa(\boldsymbol{x})) \qquad (9)$$

(a,b は定数，κ は曲率) で定義する．ここで，a は一定な移流項 (a の符号が輪郭が膨張するか収縮するかを決定する)，b は曲率の強さを決定する係数を表し，$g(\cdot)$ は

$$g(\boldsymbol{x}) = \frac{1}{1+|\nabla G * I(\boldsymbol{x})|} \qquad (10)$$

(G はガウシアンフィルタ，$I(\boldsymbol{x})$ は輝度値) で画像から得られるエッジ停止関数である．この関数は輝度勾配が小さい個所では $g \to 1$ となり，輝度勾配が大きい領域の境界付近では $g \to 0$ となる．従って，領域の境界付近で $F \to 0$ となり，輪郭の動きが止まるようになる．

(9) 式の成長速度は輪郭においてのみ与えられているが，レベルセット方程式 (6) を計算するためには全空間で成長速度を必要とする．そこで，輪郭で与えられた速度から計算領域のあらゆる所で適切な速度を推定しなければならない．この考えを“拡張速度”(Extension Velocities) と呼ぶ．Malladi らが提案した拡張速度は単純で，輪郭上にない格子点での速度関数を輪郭上の最も近い点での速度関数と同じにするということである．

3.2.2 発展方程式

動的輪郭モデルの発展方程式は，設定した成長速度をレベルセット方程式に代入することで

$$\frac{\partial \Phi(\boldsymbol{x},t)}{\partial t} = -g(\boldsymbol{q})(a+b\kappa(\boldsymbol{q}))|\nabla\Phi(\boldsymbol{x},t)| \qquad (11)$$

点 $\boldsymbol{q}$ は点 $\boldsymbol{x}$ から最も近い輪郭上の点で $\boldsymbol{x}$ の関数という式で表すことができる．

3.2.3 数値計算

前章 (11) 式を差分方程式にして計算するのであるが，空間微分 $\nabla\Phi^n_{i,j}$ を近似する正確な計算は次のように移流項と曲率項を分けて行われる．

まず，(11) 式の移流項は風上差分によって計算される．x 軸と y 軸方向の前進差分，後退差分をそれぞれ

$$D^{+x}_{ij} = \frac{\Phi^n_{i+1,j}-\Phi^n_{i,j}}{h} \quad , \quad D^{-x}_{ij} = \frac{\Phi^n_{i,j}-\Phi^n_{i-1,j}}{h}$$

$$D^{+y}_{ij} = \frac{\Phi^n_{i,j+1}-\Phi^n_{i,j}}{h} \quad , \quad D^{-y}_{ij} = \frac{\Phi^n_{i,j}-\Phi^n_{i,j-1}}{h}$$

と表し，これらを用いて

$$a|\nabla\Phi| = \max(a,0)\nabla^+ + \min(a,0)\nabla^- \qquad (12)$$

$$\nabla^+ = \Big\{(\max(D^{-x}_{ij},0))^2 + (\min(D^{+x}_{ij},0))^2 + (\max(D^{-y}_{ij},0))^2 + (\min(D^{+y}_{ij},0))^2\Big\}^{\frac{1}{2}}$$

$$\nabla^- = \Big\{(\max(D^{+x}_{ij},0))^2 + (\min(D^{-x}_{ij},0))^2 + (\max(D^{+y}_{ij},0))^2 + (\min(D^{-y}_{ij},0))^2\Big\}^{\frac{1}{2}}$$

のように計算する．

次に，(11) 式の曲率項は中心差分を用いて

$$b\kappa|\nabla\Phi| = b\kappa\left\{\left(\frac{\Phi^n_{i+1,j}-\Phi^n_{i-1,j}}{2h}\right)^2 + \left(\frac{\Phi^n_{i,j+1}-\Phi^n_{i,j-1}}{2h}\right)^2\right\}^{\frac{1}{2}} \qquad (13)$$

のように計算される．

4. Two-cycle 法

4.1 はじめに

今までのレベルセット法は対象物の輪郭を抽出するのに偏微分方程式をとくことにより求めていたが，Two-cycle 法では偏微分方程式を解かないこと，および Narrow Band よりもさらに計算範囲を縮めた言うならば究極の Narrow Band により計算量と計算範囲を減らしている．これにより画像サイズが小さめのもの (64×64) でも十数倍，大きいもの (512×512) にいたっては 150 倍ほどの計算速度を記録している．

4.2 データ構造

従来のレベルセット法は輪郭を抽出するのにレベルセット方程式を離散化したものを解くことによって求めていたが，Two-cycle 法では，二つのリスト L_{in}, L_{out} 内の要素の速度が正か負かの判断のみで収縮を行っている．二つのリストは次のように定義される．

$$L_{in} = \{\boldsymbol{x} \mid \Phi(\boldsymbol{x}) < 0 \text{ and } \exists \boldsymbol{y} \in N(\boldsymbol{x}) \text{ such that } \Phi(\boldsymbol{y}) > 0\}$$

$$L_{out} = \{x \mid \Phi(x) > 0 \text{ and } \exists y \in N(x) \text{ such that } \Phi(y) < 0\}$$

ここで $N(x)$ は以下で定義された x の離散近傍である．

$$N(x) = \{y \in D \mid \sum_{k=1}^{K} \mid y_k - x_k \mid = 1\} \quad \forall x \in D \qquad (14)$$

L_{in} は境界 C の内側にある近傍格子点のリストであり，L_{out} は C の外側にある近傍格子点のリストである．つまり，従来のレベルセット法違い，境界 C は二つのリストの間にあることになる．

従来のレベルセット法では (6) のレベルセット方程式をとくため，偏微分方程式を解くことは計算的に厳しいので，リアルタイムに要求を満たすのは難しいというのがよく起こる．しかし，この手法では曲線を外側に動かすために，L_{out} から L_{in} に切り替えるだけ，同じように曲線を内側に動かすために，L_{in} から L_{out} に切り替えるだけでいい．この方法は高速な実装の基礎を形作っている．

この実装ではデータ構造はまったく簡単になり，レベルセット関数 Φ のための配列，速度 F のための配列，近傍格子点の二つのリスト：L_{in} と L_{out} のみで書ける．

近傍格子点の内側と外側に加えて，C の内側であり，L_{in} ではない格子点を内部点と呼び，C の外側であり L_{out} ではない点を外部点と呼ぶ．より早く計算するため，整数 $\{-3,-1,1,3\}$ の制限された集合から Φ の値をとる．

$$\Phi(x) = \begin{cases} 3 & \text{if } x\text{は外部点} \\ 1 & \text{if } x \in L_{out} \\ -1 & \text{if } x \in L_{in} \\ -3 & \text{if } x\text{は内部点} \end{cases} \qquad (15)$$

レベルセット法では速度を (11) 式のように決めたが，この方法では，発展速度 F はデータ依存外速度 F_{ext} と内速度 F_{int} から成っている．

$$F_{ext} = \begin{cases} 1, & if\ f(x) \in [I_1, I_2] \\ -1, & \text{otherwise} \end{cases} \qquad (16)$$

f は画像強度を示し，$[I_1, I_2]$ は分割される領域での強度の範囲だ．曲率に基づいた正則化において，発展速度は以下のようになる．

$$F = F_{ext} - \lambda\kappa \qquad (17)$$

この速度を用いてアルゴリズムを動かすが，このアルゴリズムでは発展速度は符号だけが使われるので，従って F は整数行列であり，1,0,-1 と等しい．

4.3 Two-cycle 法の輪郭発展経過

全ての反復で，まず L_{out} と L_{in} 内の全ての点で速度を計算し，配列 F にその符号を記憶する．その後，リスト L_{out} の全てを走査し，もし $F > 0$ ならば *switch_in*( ) の手続きを行う．この走査の後，L_{in} 内のいくつかの点は新しく加えられた近傍格子点により内部点になり，それらは削除される．リスト L_{in} を走査し，$F < 0$ の点で *switch_out*( ) の手続きを行う．同じように外部点はこの走査の後 L_{out} から削除される．両方のリストの走査の後，停止条件はチェックされる．もし停止条件を満たしているならば，発展を止める．さもなければ，この反復過程を続ける．

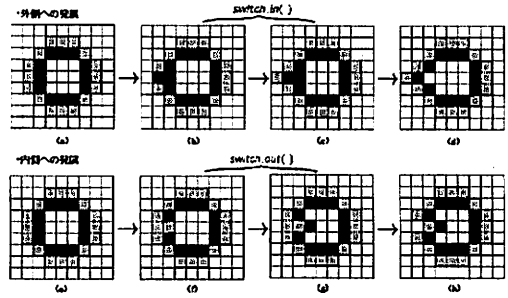

図 2　Two-cycle 法の輪郭発展経過

4.3.1 switch_in()

- ステップ 1：L_{out} からxを削除し，それを L_{in} に加える．$\Phi(x) = -1$ を設定する．(図 2(b))
- ステップ 2：$\Phi(y) = 3$ を満たす $\forall y \in N(x)$ で，yを L_{out} に加え，$\Phi(y) = 1$ を設定する．(図 2(c))

switch_in( ) の手続きで，最初のステップは L_{out} から L_{in} に xを切り替える．$x \in L_{in}$ で，以前に外側の点だった点の全ての近傍は格子点に隣接していて，ステップ 2 で L_{out} に加えられる．L_{out} 内のあらゆる点に *switch_in*( ) を行うことによって，境界はその場所から外側に格子点一つ分だけ動く．

4.3.2 switch_out()

- ステップ 1：L_{in} からxを削除し，それを L_{out} に加える．$\Phi(x) = 1$ を設定する．(図 2(f))
- ステップ 2：$\Phi(y) = -3$ を満たす $\forall y \in N(x)$ で，yを L_{in} に加え，$\Phi(y) = -1$ を設定する．(図 2(g))

近傍格子点の内側で *switch_out*( ) の手続きを踏むことによって，境界を内側に格子点一つ分動かす．

4.4 停止条件

もし次の状態を両方とも満たしているならば，曲線発展アルゴリズムは止まる．

(a)　全ての近傍格子点で，速度は下式を満たす．

$$F(x) \leqq 0 \qquad \forall x \in L_{out}$$
$$F(x) \geqq 0 \qquad \forall x \in L_{in}$$

(b)　前もって指定された反復の最大値に達している．

4.5 平滑化

Two-cycle 法では輪郭の平滑化のためにレベルセット関数にガウシアンフィルターを適用する．平滑過程は F_{ext} で動く発展と分離されているので，もしノイズレベルが低ければ，あまり高い頻度では起こらない．

ゼロレベルセットの滑らかさにだけ興味があるので，L_{out} と L_{in} 内の格子点で，フィルタ G に対する Φ の反応を計算する．

- L_{out} の全ての格子点xで，$G \bigotimes \Phi(x)$ を計算する．

もし $G \bigotimes \Phi(x) < 0$ ならば，*switch_in*(x) を使う．

- L_{in} の全ての格子点xで，$G \bigotimes \Phi(x)$ を計算する．

もし $G \bigotimes \Phi(x) > 0$ ならば，$switch\_out(x)$ を使う．

5. ハードウェア構成

ハードウェア構成を図に示す．前章で示したアルゴリズムをステートマシンとして実装した．

プロセッサ部は演算用の処理回路といくつかのレジスタからなる．レジスタは現在の状態を示す state レジスタ，ラスタスキャンを行うための pc レジスタ，S フラグの有無を判断する s_e レジスタ，F がセットされた状態かを判断する f_e レジスタからなる．基本的な動作は pc をインクリメントしていき，既定値に達したら次の状態に移動する．状態遷移図を図に示す．各状態での動作をそれぞれ説明する．

- 初期化

– 初期状態では，全てのレジスタには不定値が入っているのでこれらを全て初期化する．リセット信号が有効になったときに全てのレジスタはリセットされる．state レジスタには step1 の値が入り，ここから処理が始まる．

- STEP1

– 初期データを読み込む．

– 処理が完了したら STEP2 へ移る．

- STEP2 この STEP は Two-cycle におけるサイクル 1 である．

– s_e が無効の場合 (STEP2 の動作が初回の場合)，輝度 I から速度 F を生成する．速度 F のしきい値以内であれば 1，それ以外の場合は-1 を入れる．

– 速度 F，ϕ から s_e フラグを立てる．

– 速度 F と ϕ の組み合わせが 1,-1-か 1,1 のときに s_e にフラグが立つ．

– S フラグの立っているレジスタについて，switch_in, switch_out の処理を行い．s_e フラグを元に戻しておく

– 隣接点の削除処理を行う．

– ステップが既定値に達して STEP3 へ進む．

- STEP3

– s_e が有効ならば STEP2 へジャンプ．その際 s_e は無効に戻しておく．

– s_e が無効ならば STEP4 へジャンプ．

- STEP4 この STEP は Two-cycle におけるサイクル 2 である．

– 全画素に対し，ϕ の値が 1,-1 の場合はガウシアンフィルタをかける．

– ガウシアンフィルタの値によって s_e フラグを立てる．

– S フラグの立っているレジスタについて，switch_in, switch_out の処理を行う．

– 隣接点の削除処理を行う．

– ステップが既定値に達して STEP5 へ進む．

- STEP5

– s_e が有効ならば STEP4 へジャンプ．その際 s_e は無効に戻しておく．

– s_e が無効ならば停止へジャンプ．

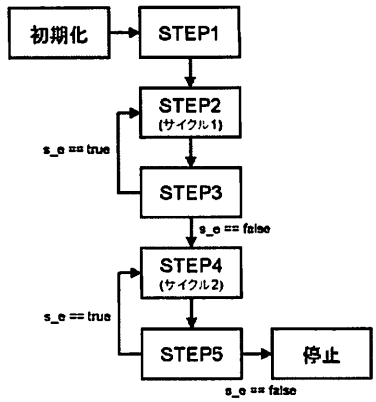

図 3　ステートマシンの状態遷移図

- 停止

– 全画素に対し，ϕ が-1 のときは輝度 I に 255 を，ϕ が 1 のときには輝度 I に 0 を入れる．

6. 評　　価

6.1 シミュレーション環境

シミュレーション環境は Xilinx 社の ISE webpack 8.2i を使用した．PC は CPU が Pentium4 2.8CGHz，メモリが 2GB，OS は WindowsXPproSP2 を使用した．

6.2 シミュレーション

verilog にて回路記述を行い，シミュレーションを行った．シミュレーションには 64 × 64pixel の画像を使用した．今回の開発ターゲットは携帯アプリケーションであるため，本来であれば画像は QVGA(320 × 240) 程度が望ましいが，開発の都合上やや不適切ではあるがこの解像度でのシミュレーションを行った．

以下にシミュレーションに用いた画像を図 4 に示す．左は人工画像，右はデジカメにて撮影した画像である．

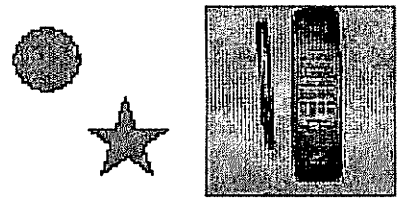

図 4　シミュレーション用画像 (左:人工画像 右:カメラ画像)

6.2.1 結　　果

結果の画像を以下に示す．処理にかかった時間はそれぞれ，人工画像が 907 マイクロ sec，カメラ画像が 513 マイクロ sec であった．

6.2.2 考　　察

シミュレーションによって出た数値はおよそ 1000 マイクロ sec 以下であった．処理時間は画素数に比例するので，QVGA では 18.8msec 程かかることが予想される．一方，java による

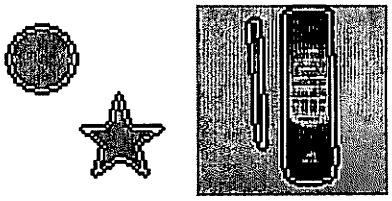

図 5 シミュレーション結果画像 (左:人工画像 右:カメラ画像)

シミュレーションでは，同等の処理に 2000msec 程かかるので，およそ 100 倍高速であることがわかる．また，リアルタイム処理を考えた場合，30fps で動作させるためには一回の処理を 30msec 以内に終わらせる必要がある．今回のシミュレーションで，Two-cycle 法はハードウェア化することでリアルタイム処理することが可能であることがわかった．これは，ハードウェア化する妥当性が示せたと言える．

6.3 回路規模

verilog による回路記述を ISE webpack 8.2i によって論理合成した結果，プロセッサのゲート数は合計で 411K ゲートとなり，これはチップ面積に変換すると $0.4mm^2$ 程度となる．なお，1FF あたり 4 ゲートとして計算した．

7. 結　　論

高速に輪郭抽出が可能な Two-cycle 法のハードウェアを設計し，シミュレーションによって動作を確認した．ソフトウェアによる実装よりも 100 倍程度高速になり，かつ，リアルタイム要求である 30msec を下回った．これによりハードウェア化した際の妥当性を示した．なお，チップ面積は $0.4mm^2$ であった．

今後は，FPGA などのデバイスに実装して評価を取る必要がある．

謝辞 本論文を作成するにあたり，研究の指導をしてくださった本田専任講師に感謝します．

文　　献

[1] M. Kass, A. Witkin, and D.Terzopoulos: "Snakes : Active contour modesl.", Int'l J.Computer Vision, Vol. 1, pp. 321-333 (1988).

[2] V. Caselles and F. Catte and T. Coll and F. Dibos: "A geometricmodel for active contours", Numerische Mathematik, Vol. 66, pp. 1-31 (1993).

[3] R. Malladi, J.A. Sathian, and B.C. Vemuri: "Shape modeling with front propagation a level set approach.", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 17, pp. 158-175 (1995).

[4] S. Osher and J.A. Sethian: "Fronts propagation with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations.", J.COmputational Physics, Vol. 79, pp. 12-49 (1988).

[5] William Clem Karl. Yonggang Shi: "A fast implementation of the level set method without solving partial defferntial equations.", Boston University Technical Report (2005).

[6] M. Sussman, P. Smereka, and S. Osher. "A level set approach for computing solutions to incompressible two-phase flow.", J.Computational Physics, pp. 146-159 (1994).