Vol.2018-MPS-118 No.31
Vol.2018-BIO-54 No.31
2018/6/15

遺伝的プログラミングによる閾値変動を用いたエッジ検出の検証

佐竹 睦稀[1]　小野 景子[2]

概要：遺伝的プログラミング (Genetic Programming:GP) は進化計算手法の一つであり，局所最適解が複数ある問題や設計変数間に依存関係がある問題など，大域最適解を得ることが難しい問題に対する，高い解探索能力と汎用性を持つ．画像のエッジ検出は物体抽出や識別の基本要素となる重要な技術であるが，画像形状によって最適なエッジフィルタが異なることが知られている．本研究では GP を用いた学習法を提案することでエッジ検出の高性能化を図る．また，同時に GP の計算コストを削減するため GPGPU を適用する手法を提案し，その有効性を検証する．

Edge Detection based on Adaptive Threshold using Genetic Programming

Satake Mutsuki[1]　Ono Keiko[2]

1. はじめに

社会には膨大な画像データがあふれており、そこから有用な情報を抽出することは，情報利活用の点から重要な研究課題である．しかし，画像は様々な特徴量を有しているため，自動的で汎用的な情報抽出は困難である．その要因の一つとして物体検出の難しさが挙げられる．一方，遺伝的プログラミング (Genetic Programming:GP) [1] は進化計算手法の一つであり，局所最適解が複数ある問題や設計変数間に依存関係がある問題など，大域最適解を得ることが難しい問題に対する，高い解探索能力と汎用性を持つ．本研究では物体検出の基本要素であり抽出性能に大きく影響を及ぼすエッジ検出に対して，閾値変動型 GP を提案し，適用的なエッジ検出を実現する．また，画像処理において多く用いられる画像形状に基づいたエッジフィルタではなく，GP は学習データに基づきエッジフィルタの設計が可能である．輪郭のみを抽出するように学習をすることで，より物体検出に向いたエッジフィルタの作成ができると考える．GP を用いたエッジフィルタはこれまでに提案されているが [2]，エッジ検出のための閾値が自動化されていない．エッジ検出のための閾値を必要としており，それが性能に大きく影響を及ぼす．最適なパラメータ学習には多くの計算コストが必要であり，また，GP によるエッジ検出は Moving-Window を用いるため計算コストが非常に大きくなる．そこで本研究では，GP によるエッジ検出においての閾値変動手法を提案し，その性能を検証する．また，同時に GP の計算コストを削減するために GPGPU を適用する手法を提案し，その有効性を検証する．

[1] 龍谷大学 大学院 理工学研究科 電子情報学専攻
[2] 龍谷大学 大学院 理工学部

2. 提案手法

2.1 *shift* 関数

本研究では，[2] と同様に $shift$ 関数 $\{shift_(n,m)\}$ を導入した．$shift$ 関数の役割は，注目しているピクセルから n 行 m 列のピクセルの画素情報を取得する関数であり，この関数により，多様なフィルタ構造をシンプルに設計することができる．今回は取得する情報はグレースケール値とした．例えば，n が 1 の場合，注目ピクセルから 1 つ右のピクセルの輝度値を取得し，m が-1 の場合は注目ピクセルから 1 つ上のピクセルの値を取得する．注目しているピクセルからシフトした結果，画像領域外になってしまった場

Vol.2018-MPS-118 No.31
Vol.2018-BIO-54 No.31
2018/6/15

合は，画像の折り返しを行う．ここで，n 及び m は個体生成時に一様な乱数で決定される．既存のフィルタの 1 つであるロバートフィルタを $shift$ 関数用いて表すと図 1 のようになる．

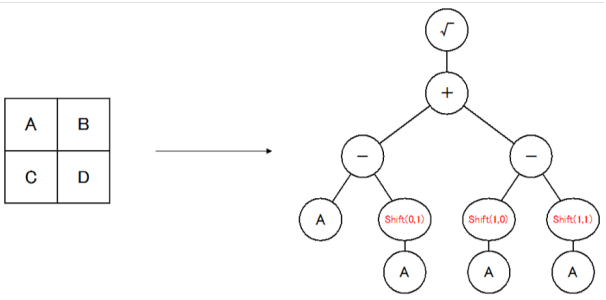

図 1　shift 関数によるロバートフィルタの表現

2.2　閾値変動法

本研究では，母集団 P を分割しサブ集団に異なる閾値を割り当て，相互に情報を交換することで最適な閾値を学習する．母集団 P を N 分割し，$P = \{p_n; n = 1, 2, \cdots, N\}$ とし，サブ集団 p_n の閾値を t_n とする[*1]．初期個体では $t_n \sim U(0, 255)$ とする．一定周期ごとに t_n を個体の評価値に基づき更新する．ここで，本研究では目的関数の最大化を図り，また，閾値に最適な値が存在することを仮定して，各サブ集団の上位個体の閾値と評価値から次状態の閾値を推定する．具体的には，上位個体群から推定した正規分布の平均値 μ を次の世代で継承するように，次世代の閾値 $t_n \sim \mathcal{N}(\mu, \sigma)$ をサブ集団数サンプリングを行う．これを繰り返すことでサブ集団における適切な閾値の学習を実現させる．

2.3　CUDA による並列化

評価においてピクセルごとに同じフィルタを適用する必要があるため，本研究では CUDA を用いた並列処理を用いる．本研究では GP の個体は深さ優先探索の木構造表現となっている．このままでは GPU で扱えないため，個体情報を通常の演算式に変換する．また，個体ごとに木構造が異なるために，それらを GPU で扱えるようにする必要がある．

そこで，本研究では，Python で CUDA を扱える Pycuda を用い，その SouceModule というモジュールを用いて異なる個体を GPU で処理できるようにした．具体的には，C 言語ベースでエッジ検出を行うソースコード（個体情報）を入力し，文字列として変数に代入する．ここでソースコードの全体を 1 つの変数に代入するのではなく，個体情報である演算式を入力する前後の部分でソースコードを分解して 2 つの変数に代入する．その前後のソースコードの間に個体情報である演算式を入れて文字列を連結させることにより，全ての個体に対して，適切なソースコードの生成を実現する．Pycuda は CUDA と同じように関数名，実引数，実行時の並列度 (スレッド数とブロック数) を指定して実行する．

[*1] 実験では $N = 10$ とした．

3.　評価実験

3.1　比較実験

本研究では，閾値変動を 10 世代周期で行う提案法とエッジ検出の閾値を {50,100,150,200} とし，始めから固定にする手法を比較法として実験を行なった．また CUDA による計算コストの削減を検証するために GPGPU を用いる手法と，CPU を用いる手法によって比較実験を行なった．また，画像のサイズを 10,20 倍と増やして実験を行なった．

3.2　実験設定

提案法において，shift 関数の n 及び m の範囲を $-3 \sum 3$ とし 3×3 のウィンドウサイズでエッジ検出を行なった．共通の設定として，最大世代数は 100 世代，全個体数は 400 個体，突然変異率を 1%，非終端ノードを $\{+, -, \times, \div, sqrt, square, abs, shift_(n, m)\}$, 終端ノードを $\{0.1, 0.2, \cdots, 0.9, x\}$ とした．ここで x は注目ピクセルの輝度値を示す．テスト画像は，The Barkeley Segmentation Dataset and Benchmark[3] から選択し，画像サイズは 200×133 の 26600 ピクセルの大きさで実験を行なった．図 2 に使用したテスト画像，図 3 に正解データの画像を示す．

3.3　実験結果

まず閾値変動手法と閾値を固定にした手法の評価値を次の図 4 に示し，閾値変動手法において，各サブ集団の閾値の変動を図 5 に示す．また，それぞれでのエッジ検出の結果を図 6，図 7 に示す．

図 2　テスト画像

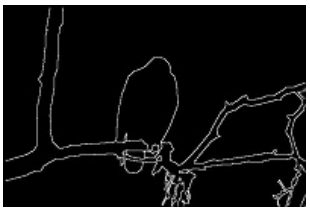

図 3　正解データ

図 4 から提案法は比較法に比べ，閾値が 100 の時以外で性能が低くなった．また図 5 から，閾値の学習により最終世代では約 140~250 に収束していることがわかる．これらの結果から，提案法は手動でチューニングした最適な閾値と一致していていないものの，ある程度の閾値の絞り込みが可能であることが分かった．また，変動型の提案法においてもある程度エッジが抽出できていることが図 6 より分かる．今回は第一歩として，サブ集団を 10 としたが，

Vol.2018-MPS-118 No.31
Vol.2018-BIO-54 No.31
2018/6/15

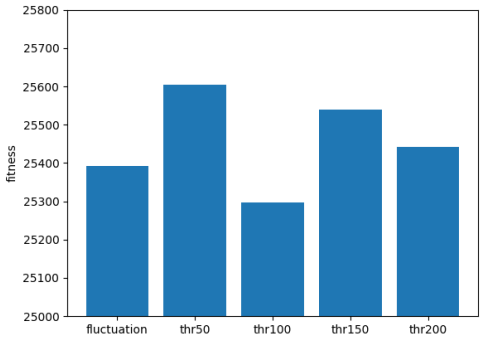

図 4　提案法と比較法のそれぞれの閾値での評価値

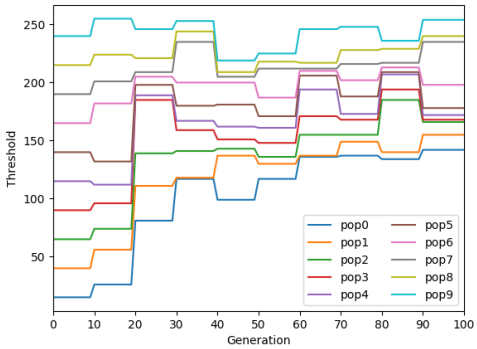

図 5　各サブ集団の閾値の変動

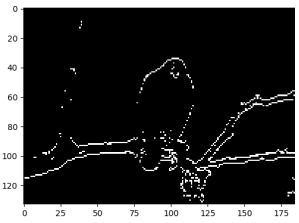

図 6　提案法での検出結果

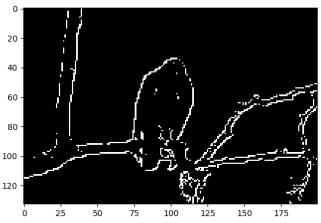

図 7　比較法での検出結果

個体の分割数を増やしより多いサブ集団で絞り込みを行う，または，閾値を変動させる時に，次世代の閾値を閾値の分散によって制御することでより良い閾値の絞り込みを行うことができると考える．次に，CUDA を組み込んだ GPGPU と，CPU で計算を行う手法の実行時間の結果を図 8 に示す．

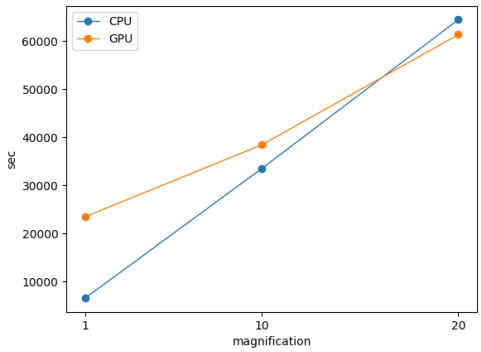

図 8　GPGPU と CPU での実行時間

図 8 から，本研究でのデータサイズでは CPU の方が実行時間が短かかった．原因として，Pycuda の SouceModule によるソースコードの取り込みに時間がかかってしまったため CPU より実行時間がかかってしまった．しかし，データのサイズ 20 倍にしたところ実行時間が逆転し，GPU での実行の方が実行時間が短くなった．20 倍以上のサイズでは，評価値の計算時間で差がつき GPU のほうが実行時間が短くなり，GPU が有効に働くことが分かった．

4. まとめ

本研究では，閾値変動手法を用いた GP によるエッジ検出を提案し，閾値を固定にする場合との比較実験を行い性能の検証を行った．また，GP の計算コストを削減するために GPGPU を適用する手法を提案し，CPU による実行時間との比較実験を行い性能の検証を行った．結果より，提案法ではある程度での閾値の絞込みが可能であることを確認し，ある程度のエッジが抽出が出来ていることが確認できた．また GPGPU では，データサイズが大きくなると有効に働くことが確認できた．

参考文献

[1] John.R.Koza. GENETIC PROGRAMMING.The MIT press 1992
[2] Mengjie Zahng , Genetic Programming For Edge Detection : A Global Approach , Evolutionary Computing (CEC) , pp.254-261 , 2011.
[3] The Berkeley Segmentation Dataset and Benchmark 入手先 ⟨https://www2.eecs.berkeley.edu/Research /Projects/CS/vision/bsds/⟩
[4] VA.Patil , S.RJagtap : GENETIC PROGRAMMING FOR OBJECT DETECTION , International Journal of Engineering Science and Technology , Vol.4 No.04 , pp.1526- 1513 , 2007
[5] W,B Langdon and R.Poil Foundations of genetic programming. Springer , 2002