J-082

輪郭の自己相似性を用いた被写体抽出の高速化アルゴリズム
A Fast Algorithm for Object Extractor Using Self-similarity of Contours

竹島 秀則 † 井田 孝 †
Hidenori Takeshima Takashi Ida

1. はじめに

近年，デジタルカメラや携帯電話など，デジタル画像を扱う電子機器が普及し，これらの機器で様々な画像加工が行われるようになってきた．所望の被写体（オブジェクト）を画像から抽出して，その背景を別のものに置き換える画像合成も実用化されているが，パソコンなどと比べ，前述した電子機器は計算速度が遅く，抽出処理の高速化が求められている．

被写体の抽出に際して，まず輪郭の概略形状を入力し，それを元にして正確に抽出する手法 [1, 2, 3, 4] が広く検討されているが，輪郭のコーナーでも抽出精度を落とさずに高速に抽出するのは容易ではない．Snakes[1] を高速化した Williams らの手法 [2] を使えば高速に輪郭を抽出できるが，Snakes の輪郭線を滑らかに保つ要素により，コーナーでは丸まるなどの誤差が生じやすい [4]．画像セグメンテーション手法の Watershed [5] に基づく平野らの手法 [3] を使えばコーナーでの誤差は生じにくいが，高い抽出精度を得るには時間のかかる前処理が必要である．

コーナーでも誤差が生じにくい別の手法として，フラクタル輪郭抽出法 [4] が提案されている．フラクタル輪郭抽出法は，正方形のブロックのサイズを段階的に小さな値に (例えば 32, 16, 8, 4 の順で) 切り替えながら，前段の結果を後段の入力として輪郭抽出処理を数回 (先の例では 4 回) 繰り返すことで輪郭を抽出する．例えばブロックサイズを 32 とした輪郭抽出処理の結果を，ブロックサイズを 16 とした輪郭抽出処理の入力とする．この手法の処理時間の大部分は，輪郭抽出処理の中で行われるブロックマッチングに費やされる．

このブロックマッチングにおいて，例えば探索範囲内で 1 画素ずつずらしたブロックすべてを候補とする full search を使えば高い抽出精度が得られるが，時間がかかる．ブロックマッチングは動画の動きや類似パターンの検索などでも使われ，例えば n-step search [6] で高速化できる．また，フラクタル輪郭抽出法専用の高速化手法 [7] も提案されている．しかしこれらは，輪郭の抽出精度が劣化する可能性がある．それに対し，本研究は精度劣化を伴わない高速化手法に関するものである．

フラクタル輪郭抽出法では，前段の輪郭抽出処理の出力が真の輪郭から多少ずれていても，後段の輪郭抽出処理で真の輪郭に近づけられる．したがって，ブロックマッチングを 1 画素単位で行う必要はない．そこで本稿では，抽出精度を保つために十分な途中段階での輪郭抽出処理のずれの許容範囲を求め，抽出精度への影響なく計算量を減らす手法を提案する．

2. でフラクタル輪郭抽出法について説明し，3. で提案する高速化手法について説明する．4. で実験結果を示し，5. でまとめる．

† （株）東芝 研究開発センター, Toshiba Corporation

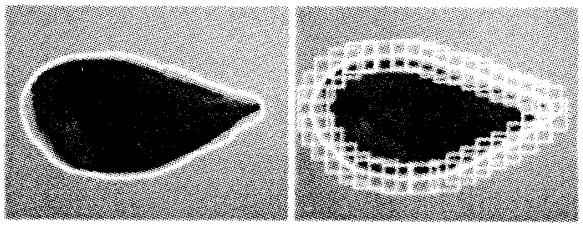

図 1: (左) 与えた画像と輪郭．(右) 配置された C_i.

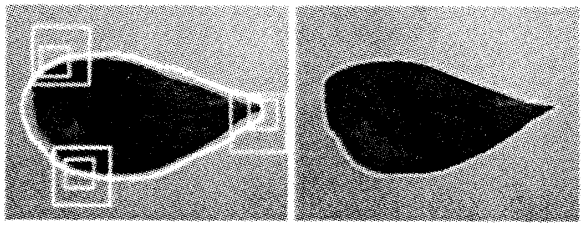

図 2: (左) 探索で得られた P_i．(右) 写像の収束結果.

2. フラクタル輪郭抽出法

フラクタル輪郭抽出法 [4] は，アルファマスクとして与えられた被写体の概略形状の輪郭を，抽出対象の輪郭に近づける手法である．アルファマスクとは，被写体領域に 1，背景領域に 0 を配置した，画像と同じ大きさの 2 値のビットマップである．植物の葉の画像に，手で与えたアルファマスクの輪郭を白線として重ねた例を図 1(左) に示す．

アルファマスクの輪郭を A，画像中の被写体の輪郭を B で表すとして，A を B に近づけるのがフラクタル輪郭抽出法である．以下にそのアルゴリズムの概略を示す．

step 1 1 辺の長さ (ブロックサイズ) が e 画素の正方形のブロック C_i (チャイルドブロック) を，図 1(右) に示すように，A に沿って数珠つなぎに，正方形の中心が A にのるように配置する．

step 2 各 C_i に対し，図 2(左) に示すように，サイズが $2e$ 画素で，画像における図柄が相似な正方形のブロックを探索する．この探索では，C_i を中心とした所定の範囲を探索範囲として，サイズが $2e$ 画素のブロックを逐次的に設定し，そこから取り出した画像データを縦横 1/2 に縮小したものと C_i の画素値の誤差を計算し，誤差が最小となるものを求め P_i(ペアレントブロック) とする．このブロックマッチングにおいては，絶対値距離 (SAD) を誤差として用いる．

step 3 アルファマスクの被写体領域に対して，自己相似写像を反復すると，A は B に近づき，図 2(右) のよ

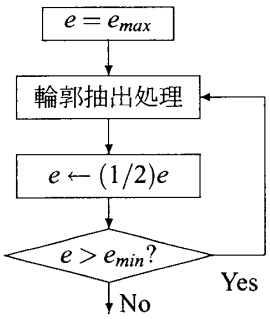

図 3: フラクタル輪郭抽出法の流れ.

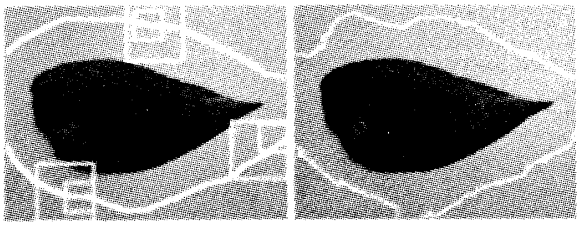

図 4: (左) 初期輪郭が離れすぎている場合の C_i と P_i. (右) 収束輪郭.

うに収束する．自己相似写像とは，P_i のアルファマスクを縦横 1/2 に縮小したもので C_i のアルファマスクを上書きする縮小写像 $P_i \rightarrow C_i$ を，step 1 で配置したすべての C_i について行う写像である．

step 4 図 3 に示すように，e の値が e_{min} 以下になるまで e を 1/2 倍に切り替えながら，その度に輪郭抽出処理 (step 1 ～ step 3 の処理) を繰り返す.

輪郭抽出処理において，入力されたアルファマスクの輪郭と真の輪郭とのずれが C_i のサイズの 1/2 におさまっていると，多くの場合，真の輪郭により近づく．このずれが C_i のサイズの 1/2 を超えてしまうと，真の輪郭に近づくことは期待できず，例えば図 4 に示す結果となる．

高速化には，step 2 のブロックマッチングの計算時間を減らすことが効果的である．ブロックマッチングで，図 5(左) のように 1 画素ずつずらしたブロックすべてを探索の対象とする方法を full search と呼ぶ．この方法では誤差が最小のブロックが得られるが，多くの計算時間を必要とする．図 5(中央) のように β 画素間隔で粗い探索を行えば速くなるが，β が大きすぎると得られるブロックの位置が大きくはずれ，抽出精度が劣化する．また，図 5(右) のように，まず粗く探索し，得られた位置の周囲でより細かく探索し，その細かさが 1 画素になるまで探索を細かくして繰り返すことで，計算量を少なくする n-step search [6] や，A の形により探索範囲を制限するフラクタル輪郭抽出法専用の高速化手法 [7] を用いても抽出精度が劣化する可能性はある．

3. 高速化アルゴリズムの提案

e を切り替えながら行う各段の輪郭抽出処理において，後段の許容誤差におさまる範囲で step 2 での計算を粗く行うことを考える．

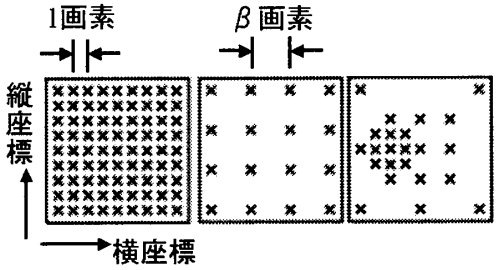

図 5: P_i の探索空間．(左)full search．(中央)β 画素間隔の探索．(右)n-step search．×印は探索空間内の候補位置.

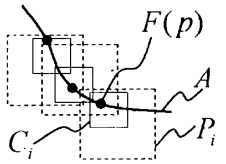

図 6: $F(\boldsymbol{p})$ と C_i，P_i，収束後 (step3 終了時) の A の関係.

以下，図 5(中央) のように，探索位置を β 画素ずつずらしながら探索するブロックマッチングを β-pel search と呼び，β をステップサイズと呼ぶ．このとき，

> β-pel search において，輪郭抽出で full search と同じ最終結果を得るには，各 e において，
>
> $$\beta < (1/2)e \qquad (1)$$
>
> であれば十分である

ことを以下に示す．このことから，β-pel search において，β として式 (1) を満たす最大の整数を用いることを提案する．ただし，最後の輪郭抽出処理 ($e = e_{min}$) では $\beta = 1$ とする．提案手法により，full search から抽出精度を劣化させることなく高速化できる．なお，提案手法では，探索候補数は e の大きさによらず一定となり，ブロックごとの計算量は $O(e^2)$ ですむ．n-step search では計算量はほぼ $O(e^2(\log e)^2)$ であるから，提案手法のほうが計算量が少ない．

式 (1) で十分であることを以下で説明する．はじめに記号を定める．

$\boldsymbol{c}$: C_i の左上の画素の位置ベクトル．$\boldsymbol{c} = (c_x, c_y)$.

$\boldsymbol{p}$: P_i の候補となるブロックの左上の位置ベクトル．$\boldsymbol{p} = (p_x, p_y)$．ブロックマッチングでは $\boldsymbol{c}$ を固定して $\boldsymbol{p}$ を切り替えながら探索する.

$\tilde{\boldsymbol{p}}$: full search を用いたブロックマッチングで選択される P_i の左上の位置ベクトル．$\tilde{\boldsymbol{p}} = (\tilde{p}_x, \tilde{p}_y)$.

$\Delta\boldsymbol{p}$: β-pel search において β を変化させると P_i の位置も変化するが，それと full search での位置との差分ベクトル．$\Delta\boldsymbol{p} = \boldsymbol{p} - \tilde{\boldsymbol{p}}$.

$F(\boldsymbol{p})$: 縮小写像 $P_i \rightarrow C_i$ の不動点 (写像によって位置が変化しない点)．$\boldsymbol{p}$ のパラメータとして $F(\boldsymbol{p}) = (F(p_x), F(p_y))$ と表す.

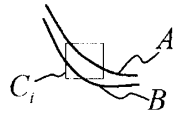

図 7: C_i は，与えた輪郭 A と真の輪郭 B の両方を含む必要がある．

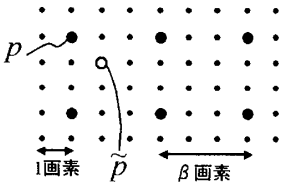

図 8: β-pel search では，$\tilde{\boldsymbol{p}}$ に最も近い画素が選ばれると仮定する．

フラクタル輪郭抽出法では，得られる輪郭線は多くの場合，不動点 $F(\boldsymbol{p})$ を通る (図 6)．また，C_i に真の輪郭 B が含まれていれば正しく抽出できる (図 7)．ここで full search で得られる輪郭は真の輪郭 B に十分近いと仮定する．このとき，C_i は中心が A の線上になるように配置されていることから，A と B とのずれが C_i のサイズの 1/2 程度までであれば，A を B に近づけることができる．そこで，以下を仮定する．

仮定 1 $\boldsymbol{a} = (a_x, a_y), \boldsymbol{b} = (b_x, b_y)$ に対し，距離 d を $d(\boldsymbol{a},\boldsymbol{b}) = \max(|a_x - b_x|, |a_y - b_y|)$ とする．full search で得られる不動点 $F(\tilde{\boldsymbol{p}})$ と，β-pel search で得られる不動点 $F(\tilde{\boldsymbol{p}} + \Delta\boldsymbol{p})$ が次の輪郭抽出処理での C_i のサイズ $(1/2)e$ の 1/2 未満，つまり

$$d(F(\tilde{\boldsymbol{p}}), F(\tilde{\boldsymbol{p}} + \Delta\boldsymbol{p})) < (1/4)e \tag{2}$$

であれば，輪郭抽出処理により同じ結果が得られる．

また，ブロックマッチングで $\beta > 1$ として探索を粗く行った場合に得られる $\boldsymbol{p}$ は，$\tilde{\boldsymbol{p}}$ の最寄りのサンプリング点となる (図 8)，つまり $\boldsymbol{p}$ と $\tilde{\boldsymbol{p}}$ のずれはステップサイズの 1/2 以下であるとして，以下を仮定する．

仮定 2 full search で得られる P_i と β-pel search で得られる P_i の差分ベクトル $\Delta\boldsymbol{p} = (\Delta p_x, \Delta p_y)$ は

$$\max(|\Delta p_x|, |\Delta p_y|) \leq (1/2)\beta \tag{3}$$

である．

写像 $\alpha_i : P_i \to C_i$ を $\boldsymbol{c}$ と $\boldsymbol{p}$ で表すと，$\alpha_i(\boldsymbol{x}) = \boldsymbol{c} + (1/2)(\boldsymbol{x} - \boldsymbol{p})$ となる．α_i の不動点 $F(\boldsymbol{p})$ は，方程式 $F(\boldsymbol{p}) = \alpha_i(F(\boldsymbol{p}))$ を解いて，$F(\boldsymbol{p}) = 2\boldsymbol{c} - \boldsymbol{p}$ である．したがって，$F(\tilde{\boldsymbol{p}}) - F(\tilde{\boldsymbol{p}} + \Delta\boldsymbol{p}) = \Delta\boldsymbol{p}$ であるから，$F(\tilde{\boldsymbol{p}})$ と $F(\tilde{\boldsymbol{p}} + \Delta\boldsymbol{p})$ との距離は

$$d(F(\tilde{\boldsymbol{p}}), F(\tilde{\boldsymbol{p}} + \Delta\boldsymbol{p})) = \max(|\Delta p_x|, |\Delta p_y|) \tag{4}$$

である．ここで $\beta < (1/2)e$ であれば，仮定 2 より

$$\max(|\Delta p_x|, |\Delta p_y|) < (1/4)e \tag{5}$$

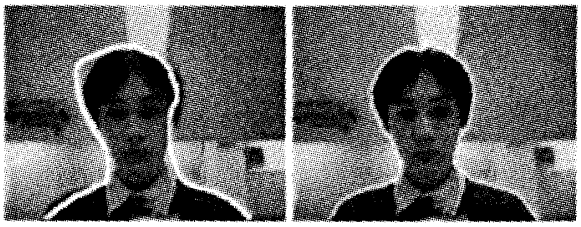

図 9: (左) 与えた画像と輪郭．(右)$\beta_{32} = 15$ の結果．

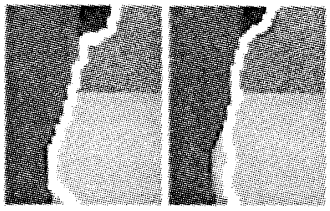

図 10: person の首の部分の拡大図．(左)$\beta_{32} = 15$．(右)$\beta_{32} = 16$．

である．式 (5) は式 (4) より，

$$d(F(\tilde{\boldsymbol{p}}), F(\tilde{\boldsymbol{p}} + \Delta\boldsymbol{p})) < (1/4)e \tag{6}$$

である．したがって，$\beta < (1/2)e$ であれば式 (2) を満たし，仮定 1 より，$\beta < (1/2)e$ であれば輪郭抽出処理で full search と同じ結果が得られる．

4. 実験

320x240 画素，RGB 各 8-bit のカラー画像に対してその概略形状を与え，ブロックマッチングのステップサイズを切り替えながら実験を行った．

概略形状は，テスト画像ごとにマウスを用いた手入力により与えた．ブロックマッチングでは，ブロックサイズ e を 32,16,8,4 の順に変えながら，輪郭抽出処理を 4 回行った．$e = 32$ でのステップサイズ β_{32} として，1 から 32 の間の整数値を与え，$e = 16, 8$ でのステップサイズ β_{16} (β_8) は，β_{32}(β_{16}) を端数切り捨てで 1/2 倍して決めた．その結果 $\beta < 1$ となるときは $\beta = 1$ とした．$e = 4$ でのステップサイズ β_4 は 1 とした．ステップサイズの例を表 1 に示す．

図 9(左) に示す person に対し $\beta_{32} = 15$ で輪郭抽出を行った結果を図 9(右) に，$\beta_{32} = 15, 16$ での首の部分の拡大図を図 10 に示す．$\beta_{32} = 15$ では full search (図は省略) と見かけ上は変わらない輪郭が得られたが，$\beta_{32} = 16$ では首の輪郭が外側にずれた．full search と $\beta_{32} = 15$ について，それぞれの途中段階 ($e = 32$ 終了時) の様子を

表 1: 実験で用いたステップサイズの例．

β_{32}	β_{16}	β_8	式 (1) を満たすか
8	4	2	満たす
15	7	3	満たす
16	8	4	満たさない
24	12	6	満たさない

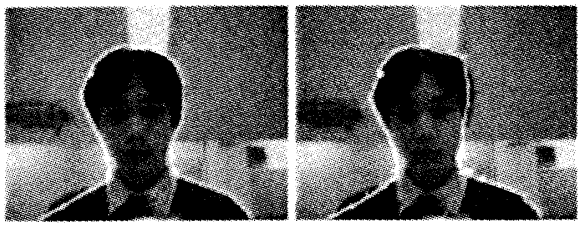

図 11: 途中段階 ($e = 32$ 終了時) の様子．(左)full search. (右)$\beta_{32} = 15$.

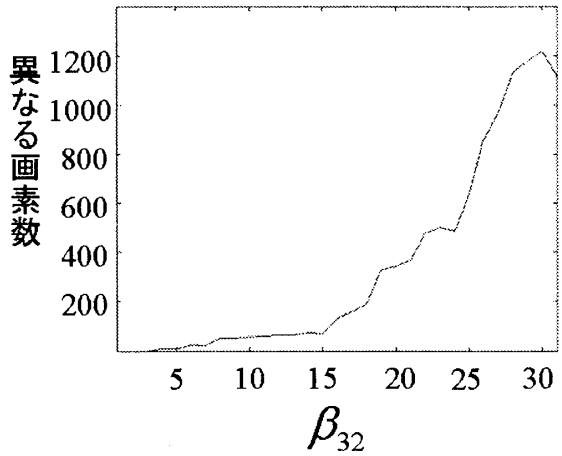

図 12: full search のときと異なる画素数 (画像 person).

図 11 に示す．P_i の配置が変化したため，この途中段階の結果も大きく異なった．しかし，先に述べたように，$\beta_{32} = 15$ で最終的に得られた輪郭は，full search でのそれに十分に近かった．これは，図 11(右) のずれが後段の輪郭抽出処理の許容誤差の範囲におさまったためである．

full search で得られたアルファマスクと，β-pel search のそれを比較し，画素値が異なる画素数を数えた結果を図 12 に示す．式 (1) の通り，$\beta < (1/2)e$，つまり $\beta_{32} < 16$ であれば抽出結果への影響はほとんどなく，$\beta_{32} \geq 16$ になると急激に異なる画素数が増加した．なお $\beta_{32} < 16$ でも誤差が 0 でないのは，仮定 1 と 2 が，完全には成立しないためと考えられる．

図 1(左) に示す leaf での抽出結果を図 13 に示す．leaf でも同様に，$\beta_{32} = 15$ では full search に近い輪郭が得られたが，$\beta_{32} = 16$ では葉の先端部分に小領域が誤検出された．異なる画素数も person と同様な結果であった．

person，leaf に対し，full-search，n-step search を用いた手法およびそれらを式 (1) で高速化した提案手法での，前処理や後処理も含めた輪郭抽出の総処理時間，および person の各ステップの処理時間を表 2 に示す．処理に要したクロック数を Pentium-4 3.06GHz の MPU で 5 回測定し，その平均値を処理時間とした．MMX などの SIMD 命令は用いなかった．

step 2 の処理時間削減により，提案手法 ($\beta_{32} = 15$) の総処理時間は，full search の 1/20 程度となり，n-step search より高速になった．

なお，full search からの抽出精度の劣化は伴うものの，提案手法を n-step search と組み合わせることができる．

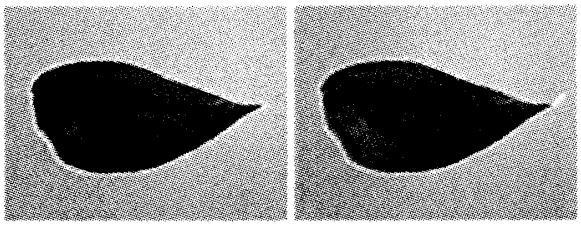

図 13: (左)$\beta_{32} = 15$ の結果．(右)$\beta_{32} = 16$ の結果．

表 2: 総処理時間と各ステップの処理時間 (単位:ミリ秒).

画像	full search	n-step	提案	提案 + n-step
person	446.5	38.2	25.8	19.6
leaf	587.0	41.1	28.0	20.1
step1	2.1	2.1	2.0	2.0
step2	434.6	26.2	14.4	8.3
step3	2.8	2.8	2.6	2.7

つまり，探索を細かくしていく途中で，式 (1) を満足した粗さで探索を打ち切る．これにより，n-step search の処理時間は 1/2 になった．

5. おわりに

フラクタル輪郭抽出法において，ブロックマッチングを粗く行いながらも抽出精度を劣化させない高速化手法を提案した．実験により，full search の 1/20 程度で輪郭抽出が行えることを示した．

今後の課題は，今回用いた仮定が満たされない場合の輪郭抽出結果への影響の評価である．

参考文献

[1] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models," *Int. J. of Computer Vision*, vol.1, No.4, pp. 321-331, 1988.

[2] D. J. Williams and M. Shah, "A Fast Algorithm For Active Contours," *Proc. of 3rd ICCV*, pp. 592-595, 1990.

[3] 平野 健嗣，吉田 俊之，酒井 善則，"電気回路シミュレーションに基づく画像エッジの補間・強調処理とその画像セグメンテーションへの応用，" 信学論 A, vol.J85-A, No.10, pp.1079-1090, 2002.

[4] T. Ida and Y. Sambonsugi, "Self-affine mapping system and its application to object contour extraction," *IEEE Trans. on Image Processing*, vol.9, No.11, pp.1926-1936, Nov. 2000.

[5] S. Beucher, "Watersheds of functions and picture segmentation," *Proc. of ICASSP 82*, pp.1928-1931, 1982.

[6] R. Plompen, "Displacement compensated prediction," *Motion Video Coding for Visual Telephony*: PTT Research Neher Laboratories, 1989, pp.325-332.

[7] 井田孝，三本杉陽子，"自己相似写像を用いたリアルタイムオブジェクト抽出，" 第 6 回画像センシングシンポジウム講演論文集，pp.161-166，2000.