大阪学院大学　人文自然論叢 ＜論　説＞
第61-62号　2011年３月

画像の分割・統合による粗輪郭からの対象物抽出の分割順次処理

淡　　誠一郎

Blockwise Processing of Object Extraction from Rough Trace of Contour with Block Division and Integration of Image

Seiichiro Dan

概要

本論文では、画像合成のための対話形式の対象物抽出アルゴリズムを取り上げ、問題分割による処理の効率化について検討する。対象とする手法は、粗輪郭とよばれる太く大まかに描いた輪郭線を一定の順序で細線化することにより、対象物の輪郭に沿った閉曲線を得る手法である。この手法は、アルゴリズムが単純な割に良好な抽出結果が得られることで知られているが、全体的な処理であり、繰り返し計算を伴うために計算量が大きいのが欠点である。このアルゴリズムは、画像全体の画素を順序付けして逐次的に処理するため、理屈の上では問題分割の考え方とは相いれない。しかし、輪郭線のトレースは直観的には局所的な処理であり、問題分割や並列処理と親和性が高いはずである。本稿では、画像を分割して当該アルゴリズムを適用した場合の問題点を分析し、それを克服できるような画像分割・統合法を提案する。

1. はじめに

画像理解やコンピュータビジョンの研究において、画像を意味のあるいくつかの領域に分割したり、画像から物体や人物の存在する候補領域を抽出したりする処理は、情景理解にいたる前段階処理と位置づけられており、認識や理解の前提として達成すべき重要な処理である。しかし、何をもって意味のある領域とみなすかは、情景のみによって決まることがらではなく、個人の背景知識や、問題、目的によっても変化する。それらを計算機シ

ステムが推し測ることは困難であり、対象を限定することなしに、対象物抽出を完全自動化することは、事実上不可能といってよい。

本研究は画像からの対象物抽出を扱うものであるが、その目的は画像の合成、すなわち、ある画像中の人物や物体を抜き出し、別の背景画像と合成することを想定している（図1）。対象の識別が目的の場合は、重要な特徴さえ損なわれていなければ、抽出される領域はそれほど厳密でなくても構わない。しかし、ここでは画像合成を目的としているので、なるべく物体の輪郭を忠実にとらえることが肝要となる。

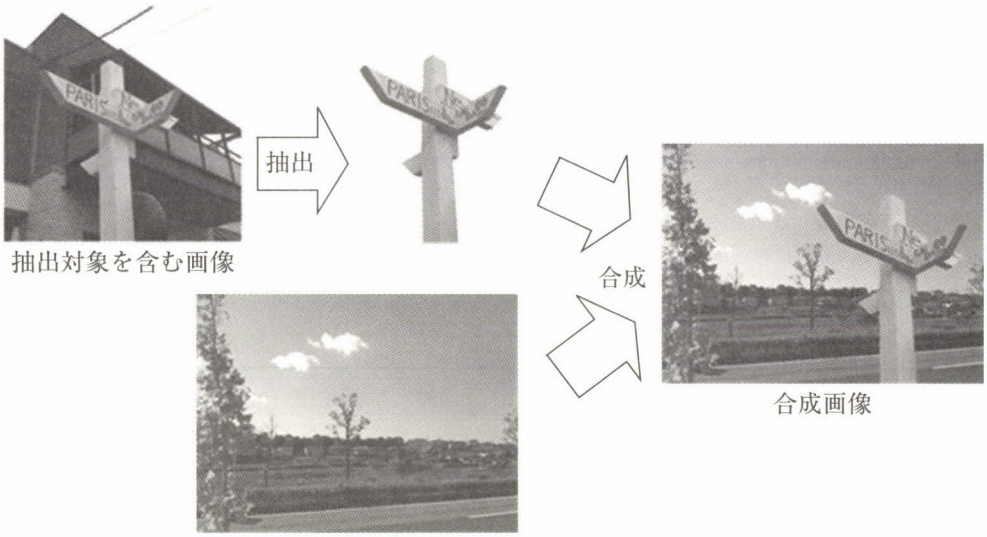


図1　画像からの対象物抽出と画像合成

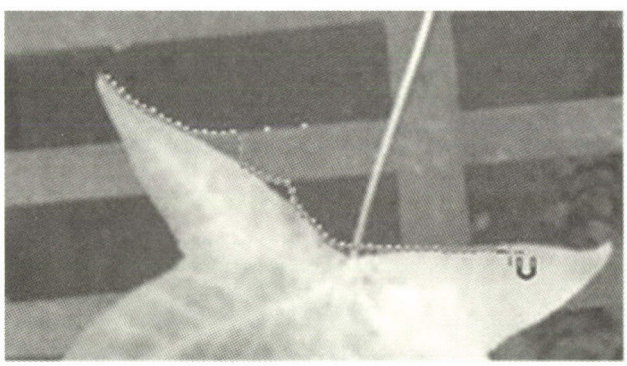


図2　輪郭線トレースツールの使用例（Paintgraphic2, Ver.1.0.0）

テレビ番組の制作現場などでは、クロマキー合成（色の着いたスクリーンを背景に用い、同系色を透明化して背景映像と合成する方法）などが用いられるが、既存の写真の一部を切り出して使いたいような場合にはクロマキー合成は使えない。上に述べたように、一般の画像から人間の意図する任意の対象物を切り出すことは困難であるゆえ、人間がデジタ

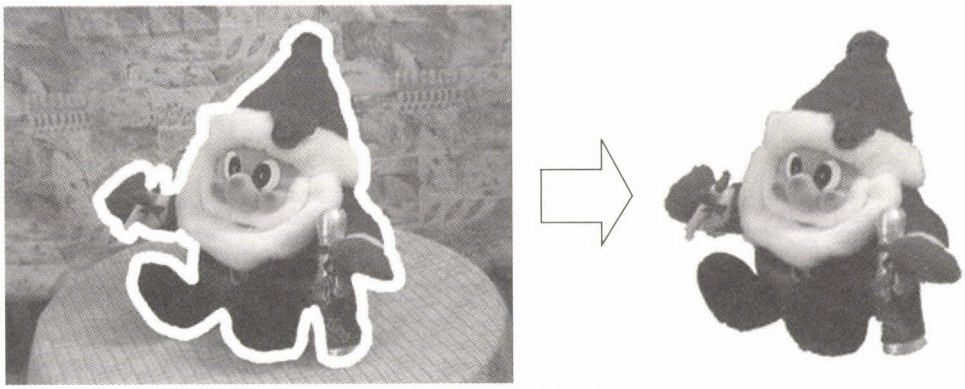


図３　粗輪郭からの対象物抽出

イザペンやマウス等で輪郭線をなぞって対象を指定せざるをえない。しかし、デジタイザペンやマウスで輪郭を正確にトレースするのはなかなか骨の折れる作業である。最近のフォトレタッチソフトには、大雑把にトレースするだけで、対象のエッジに沿うようにトレース線をフィットさせてくれるツールをもつものもあるが、背景が複雑だと意図せぬ結果となり、むしろ煩わしくさえある（図２）。

これに対し、井上ら［１，２］は、まず手動で輪郭の存在領域を幅のある閉曲線（以下、粗輪郭とよぶ）で大まかに与え、エッジ強度（明度の空間微分値）に基づく走査順序でこの領域を細線化することで対象物の正確な輪郭線を得ようとする手法を提案した（図３）。この手法はアルゴリズムの単純さの割には良好な抽出結果が得られるのが特徴であるが、通常の細線化に比べて計算量が非常に大きくなるのが難点である。淡ら［３］は井上らのアルゴリズムの見直しを行い、データ構造とアルゴリズムの実装方法の工夫により、周囲長1000画素程度の対象物の抽出で数十倍の改善率が得られることを報告している。

画像処理は、広い範囲の画素を参照して行われる大局的処理と、各画素の近傍や小領域内の画素だけを参照して行われる局所的処理に大別できる。井上らの方法は、粗輪郭中の画素全体をエッジ強度の弱い順に処理していくため、画像全体を参照する大局的処理（以下では全体的処理とよぶ）である。この事情は改良手法でも変わらない。

一方、人が物体の輪郭線をペンでトレースする場合を考えてみると、トレースは今まさにペン先がおかれた点の比較的狭い近傍だけを見て行うのが普通であり、輪郭位置の判断が遠く離れた部位の影響を受けるとは考え難い。例えば、人物のトレースなら、頭頂部のトレース時に遠く離れた腕や脚に目を向ける必要はなかろう。おそらく顎や頬さえも頭頂部の輪郭の判断には影響を与えない。このことは、輪郭線抽出のアルゴリズムが画像全体ではなく、小領域の情報だけを基にした計算となるように改良可能であることを示唆している。淡［４］はその可能性を探るべく、画像を単純にいくつかの区画に分割して処理し、結果を統合した場合についての基礎実験を試み、分割処理や並列処理の可能性と問題点について考察した。本稿では、そこで指摘された問題点への対応について検討し、粗輪郭か

らの対象物抽出の分割処理手法を提案する。

2. 粗輪郭からの対象物抽出アルゴリズム

まず、対象となるアルゴリズムの概要は以下のとおりである。

入力：[抽出対象画像]（濃淡画像）
[粗輪郭画像]（２値画像）
本アルゴリズムでは、対象物の輪郭を太い線で粗くトレースした閉曲線を“粗輪郭”とよび、粗輪郭上の画素を１、それ以外を０とした２値画像を粗輪郭画像とよぶ。

手順：粗輪郭を構成する画素のうち、削除しても粗輪郭領域の連結性が保存されるような画素の中で、抽出対象画像における同位置のエッジ強度が最も弱い点を見つけ、順次取り除いていく。削除可能な画素が無くなった時点で幅１画素の閉じた輪郭線が得られる。得られた輪郭線の内部を対象物として出力する。

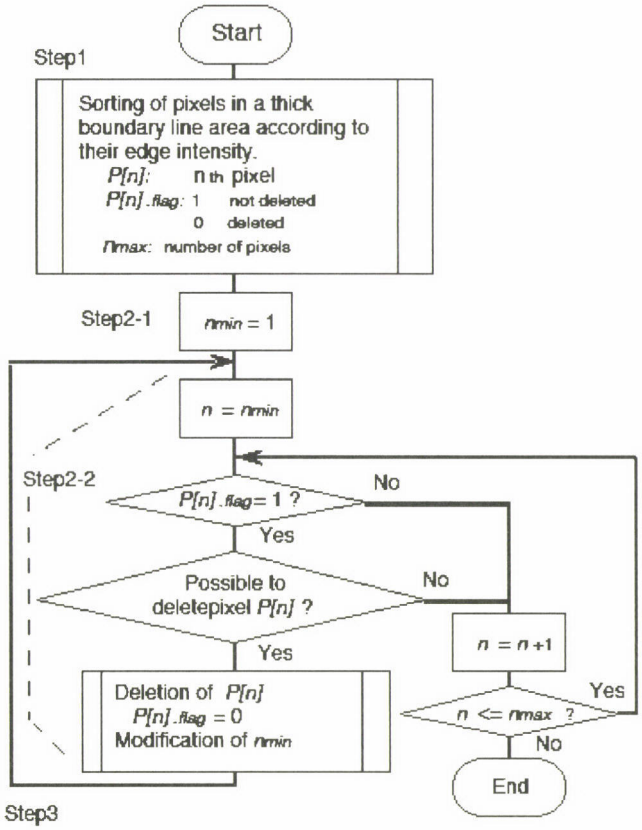
井上らのオリジナルのアルゴリズム（図４(a)）ではエッジ強度の弱い順に画素を走査し、最初に見つかった削除可能な画素を輪郭線画像から削除した後に再びエッジ強度の最も弱い画素から走査をやり直す。これに対し、淡らの改良アルゴリズム（図４(b)）ではエッジ強度をキーとし、削除可能な画素をデータとする平衡木を用いることにより、削除可能なエッジ強度最弱の画素を即座に取り出せる工夫がなされている。

なお、削除しても連結性が保存されるかどうかは、連結数（領域の境界線がその画素を通過する回数）により、機械的に判定可能である［３］。

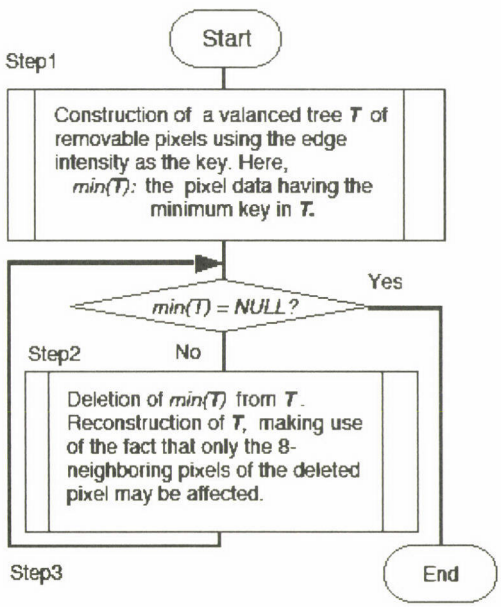
3. 問題分割と並列化に関する基本的考察

両アルゴリズムの計算量については、淡ら［３］に詳しい考察がある。井上らのオリジナルのアルゴリズムの計算時間は（粗輪郭の画素数）×（削除可能画素の検索に要する平均時間）に比例し、これはo(粗輪郭の画素数^2）と見つもることができる。これに対し、淡らによる改良アルゴリズムの計算時間は（粗輪郭の画素数）×（平衡木への粗輪郭の境界画素データの挿入／削除処理にかかる平均時間）に比例するので、o(粗輪郭の画素数×log(輪郭長)）と見つもることができる。

計算量が非線形かつ線形以上のオーダで増加するので、分割によって問題のサイズが縮小できるのであれば逐次処理であっても高速化が実現できる。それを並列処理すれば飛躍的に時間短縮できる可能性もある。しかし、残念ながら両アルゴリズムは走査順序が領域



(a) 井上らのオリジナルアルゴリズム



(b) 淡らの改良アルゴリズム

図４　粗輪郭からの対象物抽出アルゴリズム（文献［3］より）

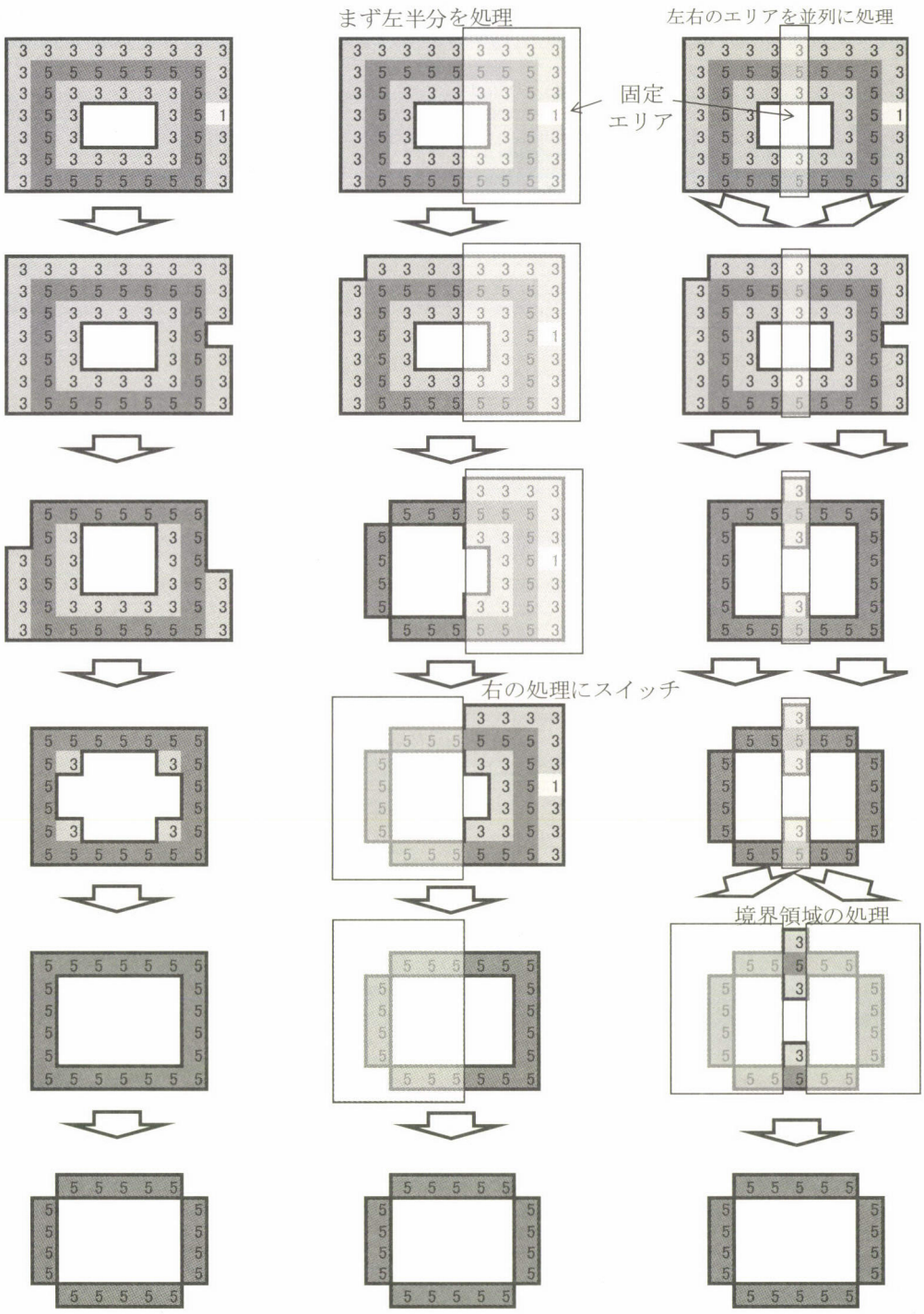
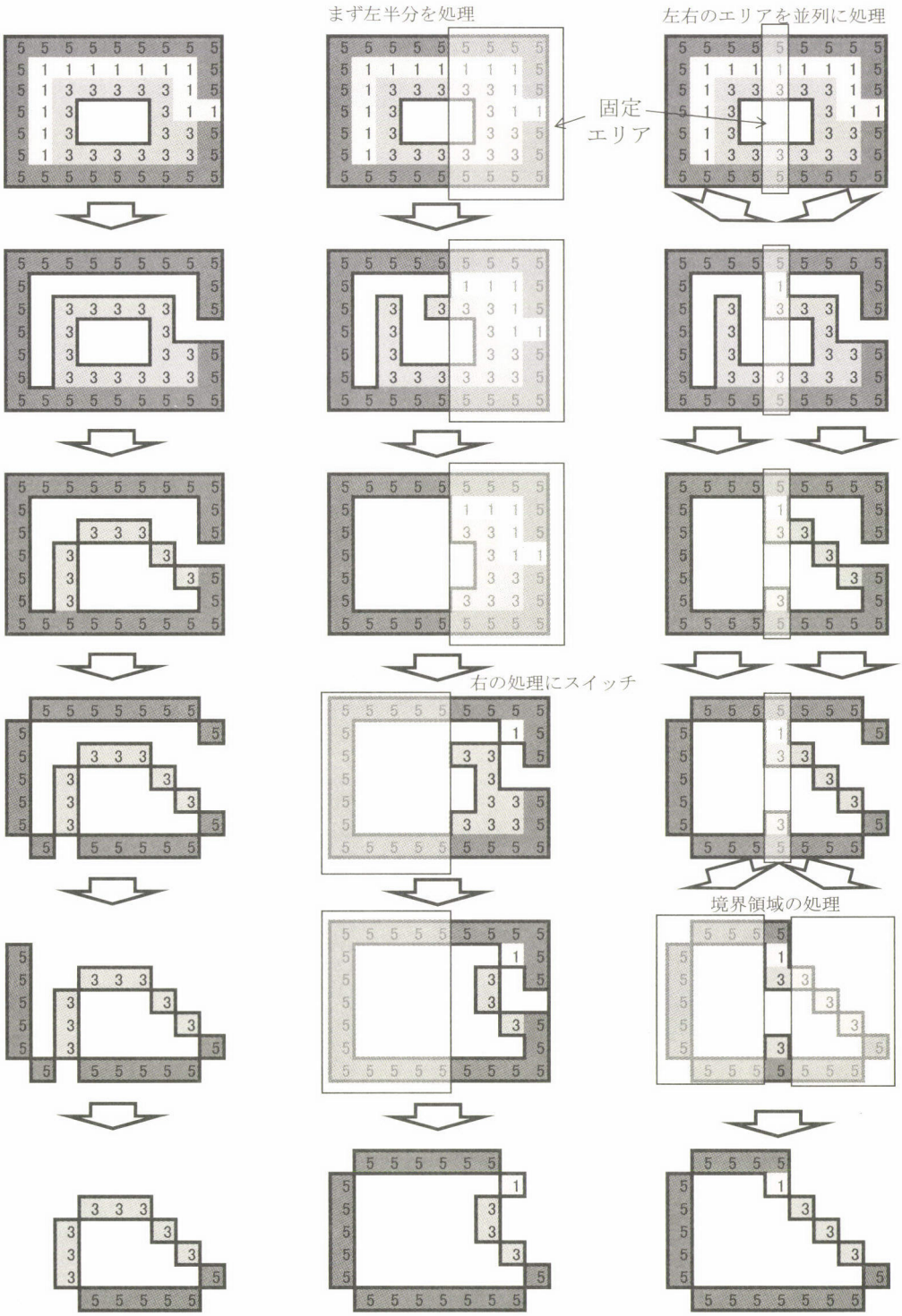


図5　分割順次処理と並列処理のシミュレーション（ケース1）



(a)通常の逐次処理　　(b)分割順次処理　　(c)並列処理

図6　分割順次処理と並列処理のシミュレーション（ケース2）

全体におけるエッジ強度順と決められており、処理結果はその順序に依存するため、同一の結果を得ねばならないという制約の下では基本的に分割順次処理や並列処理はできない。

試みに、簡単なモデルケースで分割順次処理と並列処理をシミュレートしてみた。図５，６にその様子を示す。ケース１（図５）は対象物が両アルゴリズムの想定している理想的な輪郭を持つ場合、すなわち対象物の輪郭（粗輪郭ではなく、対象の厳密な輪郭線）が、

輪郭画素のエッジ強度　＞　その近傍の輪郭以外の画素のエッジ強度

という条件を満たしているケースの一例である。また、ケース２（図６）はこの条件が成立しない例である。

両図とも一番上段の図が初期状態であり、太線内が粗輪郭領域を表す。着色部分が粗輪郭として指定した領域で、その濃さと数値が各画素のエッジ強度を表すものとする。粗輪郭外の画素は説明に無関係なので省略し白で示してある。(a) が基本となる逐次処理の結果、(b) が分割順次処理、(c) が並列処理した場合の処理過程である。分割処理 (b, c) の場合は、左右２分割と仮定し、分割順次処理 (b) では左の領域を先に処理した後に右を処理、並列処理 (c) ではまず領域の境界（中央のライン）に位置する画素を固定して境界上を除く左右の領域をそれぞれ並列に処理し、各々の結果を合成した後に境界上の画素を処理するものと仮定して処理をシミュレートした。

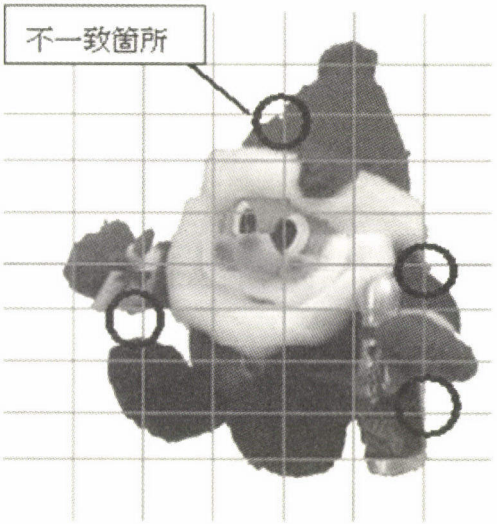
逐次処理 (a) の場合、初期状態での粗輪郭内の画素中、エッジ強度最小で連結性を損なわずに削除可能な画素は、両ケースとも右端中央の画素だけである。まず、最初にその画素が削除され、エッジ強度最小で削除可能な画素が順次削除されていく。同じ強度の画素が同時に削除可能である場合の削除順序は任意である。オリジナルのアルゴリズム［1］では、画像データのスキャン順、すなわち下よりは上、上下の位置も同じなら右よりは左の画素が優先的に削除されると考えてよい。また、改良アルゴリズム［3］の場合は、平衡木に挿入された順、つまり、その画素が削除可能となった時期の早い順である。ここでは改良アルゴリズムを前提としてシミュレートした。

まず、ケース１（図５）では、逐次処理、分割順次処理、並列処理の最終的な結果は一致した。このモデルケースは一例に過ぎないので、すべてをこの結果から推し量ることは危険であるが、明確な強い輪郭を持つ対象物では同様の結果になるであろうことは推測できる。次にケース２（図６）では、３つの処理方式の最終結果はすべて異なる。この簡単なシミュレーション結果から分かることが二点ある。まず一点は、分割順次あるいは並列処理した場合、オリジナルの逐次処理の結果と異なる輪郭線が得られてしまう場合もあるということ、もう一点は、逐次処理の結果が必ずしも最も理想的とは限らないということである。粗輪郭内のエッジ強度の強い部分をつないだ閉曲線を得るという本来の意図からすると、図６(a) よりもむしろ図６(b) の方が結果として望ましく、さらに言えば、もし

人がトレースするなら最外殻の画素をつないだ閉曲線を輪郭として選ぶであろう。このことからも、全体的処理の結果を絶対視する必然性はないことがわかる。

(a) 対象画像（粗輪郭を重畳表示したもの。サイズは640×480）

(b) 分割しない場合の抽出結果



(c) 10×10分割の場合の抽出結果

図7　単純分割による分割順次処理の基礎実験結果

表１　単純な分割順次処理結果とオリジナルの結果との一致度

画面分割数	領域数	不一致画素数	一致度（%）
2 × 2	4	1	99.9%
3 × 3	8	47	97.0%
4 × 4	12	18	98.9%
5 × 5	14	35	97.8%
6 × 6	18	50	96.8%
7 × 7	26	80	94.9%
8 × 8	31	53	96.6%
9 × 9	34	126	92.0%
10×10	41	123	92.2%

（＊オリジナルの細線化輪郭画素数：1569）

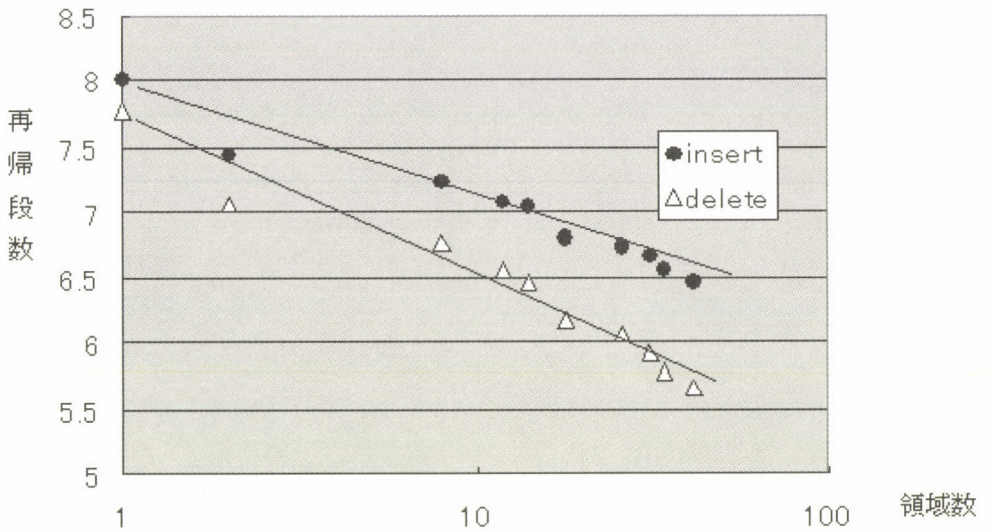


図８　領域数と平均再帰段数の関係

4. 分割処理の基礎実験

画像を単純にＮ × Ｎの区画に分割して、左から右、上から下という順で、左上の区画から順に区画ごとに淡らの改良版アルゴリズムを適用し、対象物抽出を試みた。抽出結果の例を図７に示す。(a) が実験対象画像であり、粗い輪郭線を重畳表示してある。この画像を単純にＮ × Ｎに分割して区画ごとに輪郭線を決定する。(b) が分割しない場合の抽出結果、(c) は10×10に分割した場合の抽出結果（対象物の含まれる区画のみ表示）である。輪郭位置が分割しない場合とどの程度一致するかを表１に示す。“領域数”とはＮ × Ｎの区画のうち粗輪郭領域を含む区画の数である。“不一致画素数”とは分割しない場合の結果と一致しない画素の数である、多くの場合、ずれは隣接画素への移動であり目視で

は識別できない。

大きく位置ずれを起こしている箇所を図７(c) では円で囲って示してある。これらの部分は画像の分割線の格子点が粗輪郭内部に位置したり、区画の境界線が粗輪郭をスライスするように横切ったりする場所であることがわかる。ほかの画像についても実験したが、同様であった。このような状況を避けるように分割すれば大きなずれは発生しないものと予想される。

先にも述べたように淡らの改良アルゴリズム［３］ではエッジ強度をキーとする平衡木で削除可能な画素を管理しており、計算時間のほとんどは平衡木へのデータの挿入と削除で消費される。平衡木へのデータの挿入と削除は再帰的なプログラムとして実装されており、再帰呼出しの段数に比例した処理時間を消費する。そこで、領域数とデータ挿入・削除操作の平均再帰呼出段数との関係を調べてみた。結果を図８に示す。ちなみに、データの挿入・削除の総回数は分割数による大きな変動はなく、ほぼ一定と見なせる。したがって、分割順次処理でも計算時間は短縮できることがわかる。

5. 分割処理の問題点と対処法

5.1 問題点の分析

区画の格子点が粗輪郭内に位置したり、区画の境界線で粗輪郭がスライスされたりするときに、分割順次処理で得られる輪郭位置が変動する原因について考察する。前提として、分割せずに処理した場合に、対象物の輪郭線は理想的な形で得られるものとする。すなわち、結果として得られる輪郭線上のすべての画素に対し、その近傍において次の条件が成立しているものとする。

輪郭画素のエッジ強度　＞　輪郭以外の画素のエッジ強度

粗輪郭は閉曲線状の領域であるので、２つの領域境界線を持つ。１つは対象物の外側領域との境界線、もう１つは内側領域との境界線である。画像の分割単位が矩形の区画であるものとすれば、粗輪郭は区画を構成する４つの辺によって分断される。区画の辺によって粗輪郭から切り取られた領域が、図９(a) のように２つの断面を持ち、その断面が外側の境界と内側の境界の橋渡しとなっているような場合には、エッジ強度の弱い順に画素を取り除いていくと、理想的な輪郭位置の画素が残ることは自明であり、分割しない場合と同じ結果が得られるはずである。

図９(b) はある区画の右下隅の頂点が右肩上がりの粗輪郭内部に位置した場合の処理の様子を示している。区画が左から右、上から下の順に順次処理されていくものとすると、図の左上の区画を処理する段階ではそれ以外の区画は固定されているので、結果的に左上

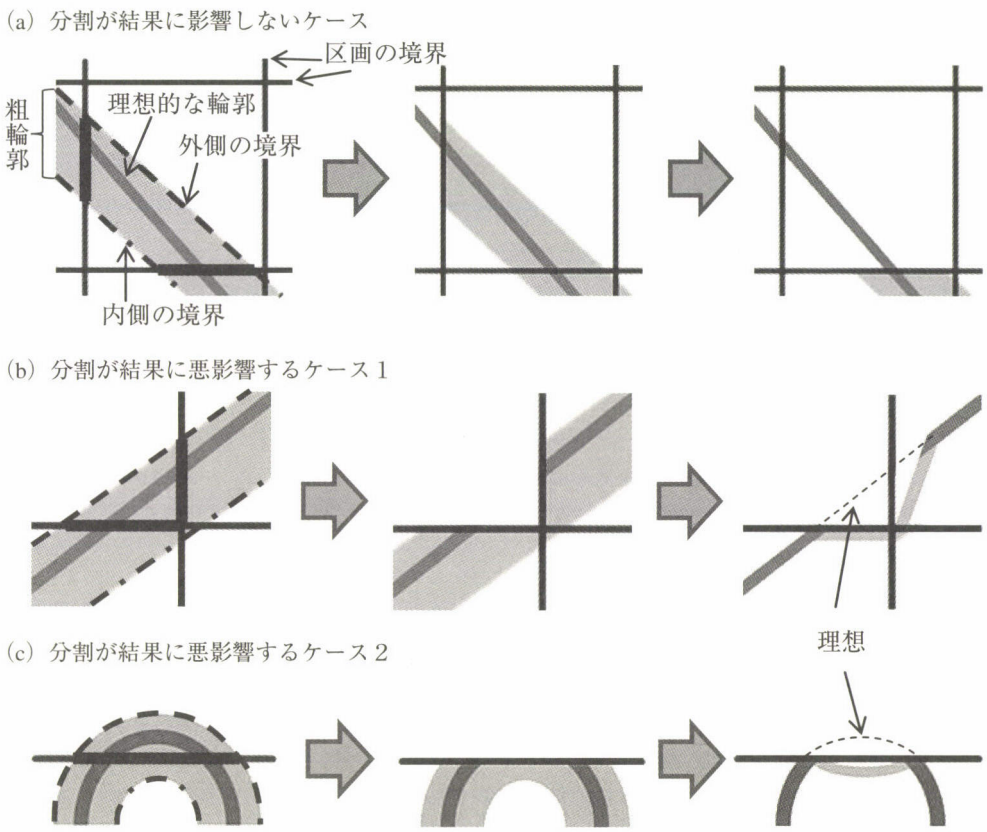


図9　分割位置が抽出結果に及ぼす影響

区画内の粗輪郭画素はすべて除去されてしまい、理想的な結果とずれが生じる。右下以外の区画の隅が粗輪郭内部に位置する場合も同様である。図9(c)は、区画の境界線が粗輪郭をスライスするように通る場合である。この場合も、(b)と同様、上の区画内の粗輪郭画素はすべて除去されてしまう。

5.2　区画の統合による問題点への対処法

画像を単純に等間隔に区画化した場合、運悪く上で述べたような不具合の起こるケースが発生するのは避けられない。そこで、まず画像を等間隔の区画に分割した上で、不具合が生じる可能性のある区画が見つかった場合には、その区画と隣接する区画を統合して処理単位を拡大することで、不具合を回避することにする。区画の統合アルゴリズムを図10に示す。

図11のモデルケースを例に統合アルゴリズムを説明する。

まず、区画A1では、区画の辺が粗輪郭領域を輪切りにする形で外部から内部へと横切って通っている。このようなケースでは図10のアルゴリズムの☆の終了条件が直ちに成立

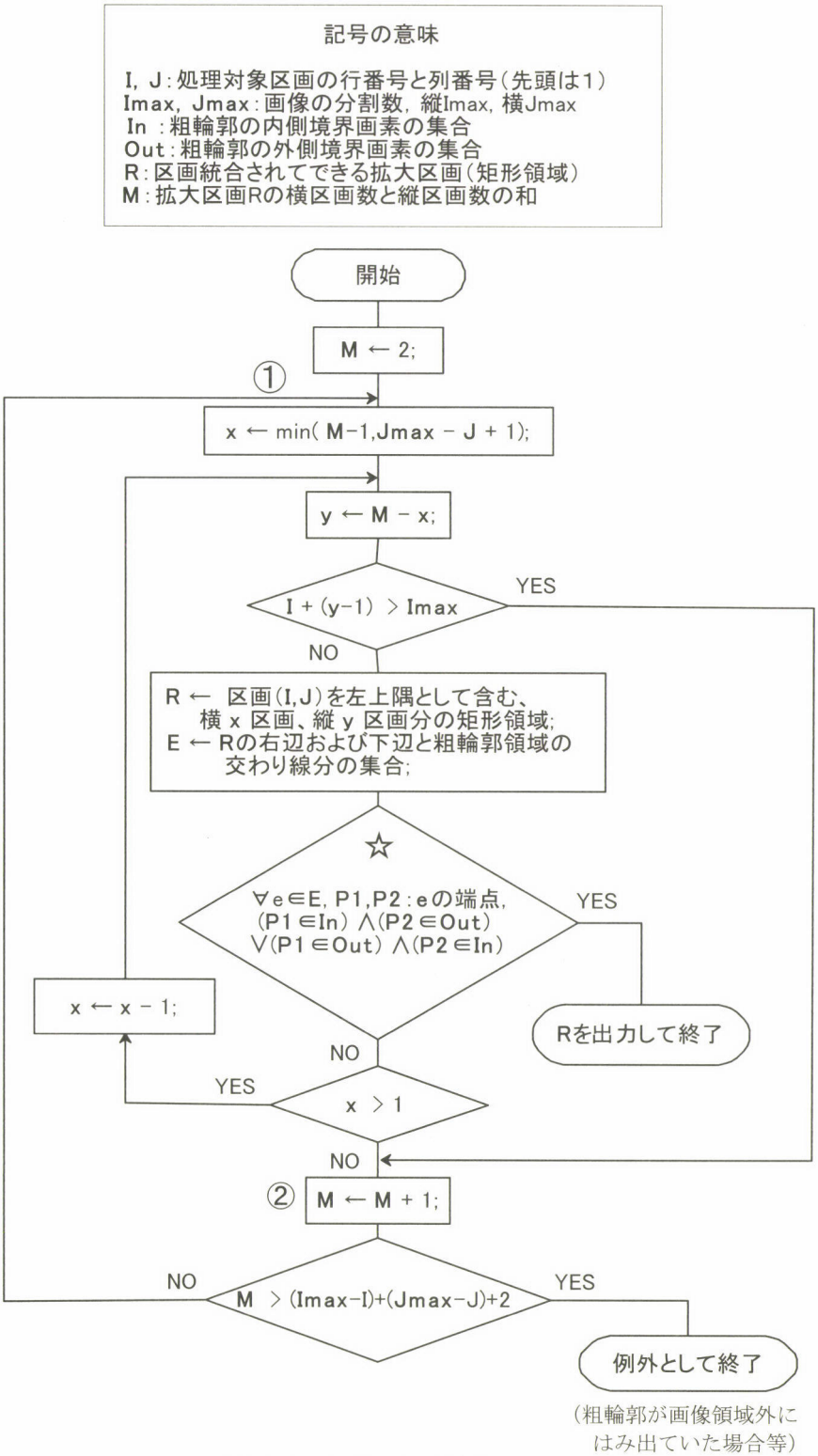


図10　区画統合アルゴリズム

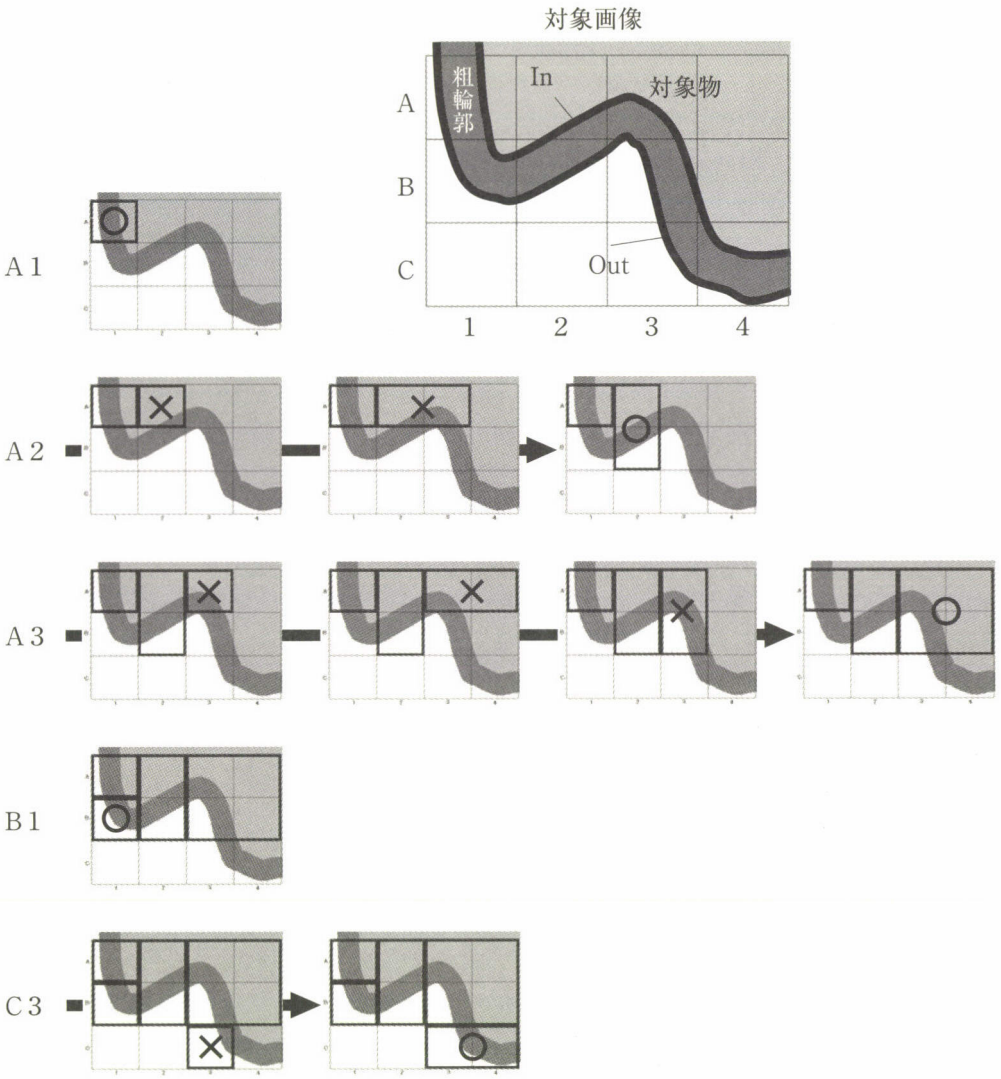


図11　区画統合の様子

するので、区画統合は発生しない。区画 B1も同様である。区画 A2は区画の隅が粗輪郭内部に位置するケースである。このケースでは☆の条件が満たされず、②のステップで M がインクリメントされて、M = 3として①に戻る。変数 M は統合される区画の縦と横の区画数の和を表し、最初は 2（統合なし）にセットされている。統合が必要な場合は、M の値は 1 ずつインクリメントされていく。M = 3 を満たす区画の統合の仕方は、縦 1 横 2 か縦 2 横 1 かの二通りであり、この順に統合が試みられる。結果として、区画 A2は下の区画 B2と統合されてアルゴリズムは終了する。区画 A3は 2 区画の統合では☆の条件は満たされないので、M = 4 となり、横 3 縦 1、横 2 縦 2、横 1 縦 3 の区画統合が順に試みられる。その結果、A3は A4, B3, B4と統合される。

表2　単純分割と区画統合を行う場合との不一致画素数の比較

画面分割数	単純分割の場合		区画統合を行った場合	
	領域数	不一致画素数	領域数	不一致画素数
2×2	4	1	4	2
3×3	8	47	6	1
4×4	12	18	10	1
5×5	14	35	11	1
6×6	18	50	15	4
7×7	26	80	17	10
8×8	31	53	20	29
9×9	34	126	23	48
10×10	41	123	26	0

（＊オリジナルの細線化輪郭画素数：1569）

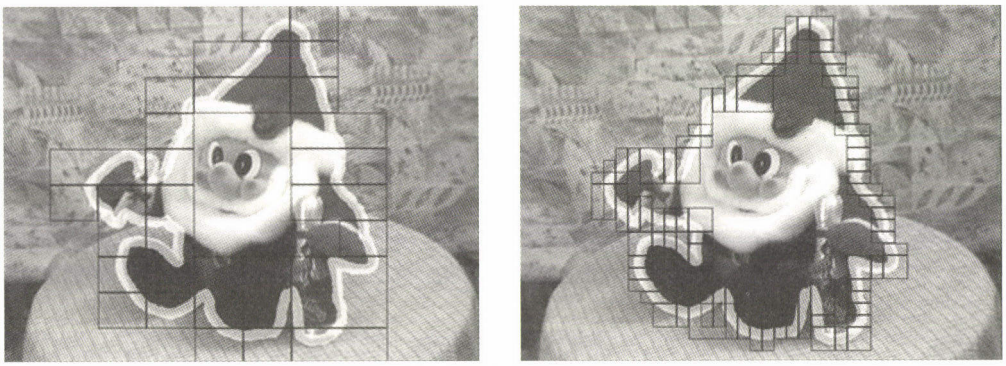


図12　区画統合の結果

5.3　実験結果

区画の分割・統合アルゴリズムをインプリメントし、領域抽出実験を試みた。表1の基礎実験と比較するために、図7(a) を対象画像、初期分割を2×2～10×10とし、単純分割の場合と区画統合を行った場合について実験した。各ケースの領域数と不一致画素数を表2に示す。“不一致画素数”は、分割しないで処理した場合の抽出結果と一致しない輪郭画素の数である。区画統合を行った場合の方が、単純分割と比べて不一致画素数が大幅に減少していることがわかる。

初期分割を10×10, 40×30として区画統合した結果を図12に示す。左が10×10、右が40×30の結果である。たまたまではあるが、10×10分割の場合、分割しないで処理した結果との不一致はなく、まったく同一の抽出結果が得られた。40×30分割の場合、不一致画素数は、統合なしでは417であったが、統合ありでは86と大きく改善された。

区画統合アルゴリズムでは必要に応じて区画のサイズが決まるので、初期分割数はいくら大きくてもよく、初期分割の区画幅が粗輪郭の幅より少し大きくなるぐらいに設定しておけばよい。副次的ではあるがこれは区画統合の長所である。単純分割の場合、分割数を

決める目安がない。分割数が大きくなればなるほど、区画の粗輪郭上に格子点が存在する確率は高くなり、全体的処理の結果と多くの不一致箇所が生じるし、運が悪ければ小さい

図13　区画統合後の粗輪郭の分割結果と対象物抽出結果

分割数であっても生じる。

なお、図12で区画の格子点が粗輪郭内部に位置しているように見える箇所があるが、実際の区画統合は区画ごとに細線化を施しながら順次実行していくので、格子点が初期状態の粗輪郭内に位置していても、その区画が処理される時点では内部点ではなくなっているためである。

ほかの画像についても比較実験を試み、良好な結果が得られることを確認した。結果の一部を図13に示す。左端の列は抽出対象画像に粗輪郭と分割線を重畳表示した画像、中央の列が分割順次処理の抽出結果、右の列は比較対象となる分割せずに処理した場合の抽出結果である。図中に示された数字は、「不一致画素数／抽出対象物の輪郭画素数」である。上３つの画像は512×512ピクセルで、初期分割が16×16、一番下の画像は1024×1024ピクセルで初期分割は32×32である。分割順次処理の結果と比較対象の結果は、注意深く観察しないとほとんど違いが分からない。判別可能な不一致箇所を丸で示した。

以上の結果により、粗輪郭からの対象物輪郭線抽出処理は、分割方法を工夫すれば分割順次処理可能であることが検証できた。

6. むすび

粗輪郭からの対象物抽出手法の空間的な問題分割の可能性について検討した。今回議論の対象とした手法では、抽出結果は画素の処理順序に依存して変化するはずであり、その順序が画像全体の画素に対して定義されているため、理屈の上では、分割処理すると得られる結果は違ってくることがある。実際、基礎実験として行った単純な分割では、部分領域の境界付近で、得られる輪郭に大きなひずみが認められた。本稿ではその不具合を極力抑えることのできる、画像の分割・統合法を提案し、実験的にその有効性が検証できた。今後の課題としては、まず、並列処理の可能性の検討が挙げられる。今回行った実験は分割順次処理であり、それだけでも計算時間の短縮につながるはずであるが、並列処理ができればさらに時間短縮が期待できる。また、粗輪郭を用いる他のアイデアとして、スネークによる輪郭線抽出手法（エッジ強度に比例した引力場を仮定して環状に配置した点列を輪郭部分に吸着させる手法）［５］を粗輪郭内に限定して動作させるという方法を考えており、その実装と今回の結果との比較も行いたい。

参考文献

［１］井上誠喜：“画像合成のための対象物の抽出法”，信学論（D-II），Vol. J74-DII, No.10, pp. 1411-1418 (1991).

［２］井上誠喜，小山広毅：“動画像合成のための対象物の抽出とはめ込み法”，テレビジョ

ン学会誌 , Vol.47, No.7, pp.999-1005 (1993) .

[3] 淡誠一郎 , 鄭小江 , 北橋忠宏 : “対話型対象物抽出アルゴリズムの高速化”, 信学論 (D-II), Vol.J79-DII, No.11, pp.1984-1987 (1996).

[4] 淡誠一郎 : “粗輪郭からの対象物抽出アルゴリズムの並列化について”, 第56回情報処理学会全国大会 , 3P-04, 論文集分冊2, pp.133-134 (1998) .

[5] Kass, M., Witkin, A. , Terzopoulos, D.: “Snake: Active Contour Models”, International Journal of Computer Vision, pp. 321-331 (1988) .

(2011年2月7日受理)