Vol.2011-CVIM-177 No.22
2011/5/19

局所グラフカットに基づく高速かつ省メモリな画像セグメンテーション

柴　涼介[†1]　泉　泰介[†1]　和田幸一[†1]

グラフカットを利用する画像セグメンテーションにおいては，対象画像の画素数に比例したサイズのグラフをオンメモリで構成し，その上でグラフの最小カットを計算する必要がある．高解像度画像，あるいは三次元画像を対象とした場合，この処理はメモリ使用量および計算量の点で非常に負荷が大きい．本研究は，画像全体に対するグラフカット (大域グラフカット) の計算を行わない，メモリ効率および実行速度の点で優れた画像セグメンテーション法を提案する．具体的には，画像をいくつかの小領域に分割し，各小領域毎で独立にグラフカット (局所グラフカット) を計算し，その結果を結合することで全体のセグメンテーションを得る．ただし，単に領域を分割し，独立にグラフカットを計算した場合，前景，背景を指定する種情報を一切持たない小領域が生じ，そのような領域でのセグメンテーション精度はしばしば大幅に低下する．本研究では，原画像を縮小した画像でプレセグメンテーションを行い，その過程で得られる大域的な種情報および色分布情報を各小領域毎のグラフカット計算に反映することで，この問題の解決を図る．評価実験により，提案手法は従来手法のグラフカットアルゴズムと比べ大幅な高速化・省メモリ化に成功している一方，精度には大きな差がないことを確認した．

Fast and Memory-Efficient Image Segmentation Using Local Graph cuts

Ryosuke Shiba,[†1] Taisuke Izumi[†1] and Koichi Wada[†1]

For high-resolution or 3-D images, the image segmentation based on graph cuts[1),2)] consumes a large amount of memory and computation time. In this paper, we proposes an approach to save those resource consumption. It uniformly divides the graph that corresponds to the original image into a number of disjoint sub-graphs(regions) and processes them independently. A problem arising in this approach is the generation of divided regions with no seeds. In such a region, segmentation quality becomes extremely low. To circumvent that problem, our approach uses the scheme that (1) we run pre-segmentation in downsampling image and (2) calculate GMM so that the pre-segmentation result is reflected as the cost of t-link of the graph. We demonstrate that our approach works faster and uses less memory compared to the conventional approach while the quality of segmentation results are competitive.

1. はじめに

グラフカット[1)] による画像セグメンテーションは，マルコフ確率場のエネルギーを大域的最小化する手法として近年盛んに研究が進んでいる．Boykov らは，グラフカットで扱うグラフに対して高速に最大フローを見つける BK アルゴリズム[3)] を開発した．BK アルゴリズムは，二次元画像セグメンテーションのデファクトスタンダードとなっているが，アルゴリズムの並列化は容易ではなく，高解像度画像や医療用三次元画像など巨大なデータに対してはその有効性は示されていない．また，画像に対応したグラフを構築するため，巨大なデータを扱う場合には膨大な量のメモリを必要とする．この問題に対して，Lombaert らは低解像度でのセグメンテーション結果を徐々に高解像度の画像へと反映させていく Multilevel Banded Graph cuts[4)] を提案しているが，並列化による高速化はなされていない．

グラフカットの並列化による高速化は，Liu ら[5)] により行われている．Liu らは，画像を均等な領域に分割し，それぞれの領域で並列にグラフカットを行い，その結果を徐々に結合していくことで従来法と同じ解を出す並列化を提案している．Liu らの並列化は，CPU のコア数に比例する高速化が達成されているが，メモリ使用量に関しては従来法と同じである．

本研究では，元画像をダウンサンプリングした画像でプレセグメンテーションを行い，その結果を前景・背景の色分布，グラフの辺の重みに反映させ，元画像を素に分割した各領域でグラフカットを行うことで，高速かつ省メモリなグラフカット画像セグメンテーション法を提案する．

2. グラフカット

本節では，従来手法である Interative Graph cuts[1)] について説明する．

2.1 ラベリング問題とエネルギー関数

画素 p の集合からなる画像を P としたとき，ラベリング問題とは，各画素 $p \in P$ にラベ

†1 名古屋工業大学
Nagoya Institute of Technology

ル L_p を割り当てる，すなわち，画素 p の集合 P からラベル集合 L への写像 H を決める問題である．セグメンテーション問題では，画素 p に割り当てるラベル L_p は，物体 (”obj”) か背景 (”bkg”) の二値を取りうる．また，すべての隣接画素の組の集合を N とする．グラフカットでは，エネルギー関数を式 (1) のように定義する．

$$E(H) = \sum_{p \in P} D_p(L_p) + \sum_{(p,q) \in N} V_{(p,q)}(L_p, L_q) \tag{1}$$

ここで，$D_p(L_p)$ は観測データに対するペナルティ関数であり，各画素 p が物体もしくは背景のモデルにどれだけ適合するかを示す．$V_{(p,q)}(L_p, L_q)$ は，近傍画素との関係を示す関数であり，p と q の輝度値が似ているほど大きい値を返す．

グラフカットでは式 (1) で定義したエネルギー $E(H)$ を最小にする写像 H を対応するグラフ G の最小カットを計算することにより求めることでセグメンテーションを行う．

2.2 グラフカットによる画像セグメンテーション

セグメンテーションを実行したい画像に対して，ユーザはマウスで線を引くことにより，確実に物体である画素と確実に背景である画素を seed として指定する．$\mathcal{O}$ と $\mathcal{B}$ をそれぞれユーザが選択した”物体 seed”と”背景 seed”である画素の集合とする．ユーザが選択した seed の情報をもとに，図 1 のようなグリッドグラフ G を構築する．

画像グラフ $G' = (V', E')$ を，重みつき有向グラフとする．G' の各頂点 $v \in V'$ は，画像 P の各画素 p に対応し，各辺 $(p, q) \in E'$ は，P の隣接画素の組 $\{p, q\}$ に対応する．G' の各辺 (p, q) を n-link と呼び，n-link のもつ重みは隣接画素の組 $\{p, q\}$ の輝度差により決まる．

グラフカットで扱うグラフ $G = (V, E)$ は，画像グラフ G' に二つのターミナル頂点 s, t を追加し，s, t と各画素頂点 $v \in V'$ を辺で接続したグラフである (図 1)．ターミナル頂点 s, t と画素頂点 $v \in V'$ を結ぶ辺 $(s, v), (v, t)$ を t-link と呼ぶ．したがって，グラフ G の頂点集合 V と，辺集合 E は次のようになる．

$$V = V' \cup \{s, t\}$$
$$E = E' \cup \{(s, v), (v, t) | v \in V'\}$$

G の各辺に割り当てる重みは表 1 により設定する．表 1 中の各式は，

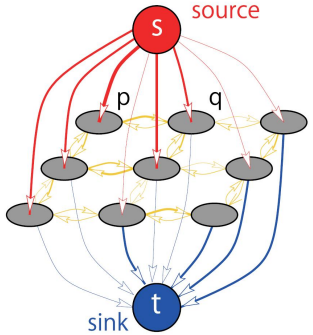


図 1 グラフカットで扱うグラフ G
Fig. 1 Example of Graph G

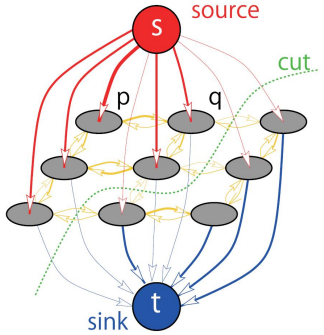


図 2 カットの例
Fig. 2 Example of Cut

表 1 辺に割り当てる重み
Table 1 weight for edge

edge	weight	for
(p, q)	$B_{(p,q)}$	$(p, q) \in N$
(s, p)	$\lambda \cdot R_p(”bkg”)$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
(p, t)	$\lambda \cdot R_p(”obj”)$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$

$$K = 1 + \max_{p \in P} \sum_{q:(p,q) \in N} B_{(p,q)} \tag{2}$$

$$R_p(”obj”) = -\ln Pr(I_p|\mathcal{O}) \tag{3}$$

$$R_p(”bkg”) = -\ln Pr(I_p|\mathcal{B}) \tag{4}$$

$$B_{(p,q)} \propto exp\left(-\frac{(I_p - I_q)^2}{2\rho^2}\right) \cdot \frac{1}{dist(p, q)} \tag{5}$$

である．

seed である頂点の t-link には，カットに含まれないように，接続する n-link の容量の和よりも大きい値 K を設定する．I_p は画素 p の輝度値，式 (5) 中の $dist(p,q)$ は画素 p,q それぞれの座標のユークリッド距離を表す．seed でない頂点の t-link には式 (3, 4) により，seed 情報を用いて画素 p の”前景らしさ”，”背景らしさ”を対数尤度により設定する．前景と背景の seed から，前景・背景それぞれのヒストグラム $\theta(c,\mathcal{O}),\theta(c,\mathcal{B})$ を得る．ここで $\theta(c,\mathcal{O})$ は，前景ヒストグラムでの明度 c の相対頻度 ($\sum_c \theta(c,\mathcal{O})=1$) を表す．背景ヒストグラムについても同様である．式 (3, 4) 中の尤度 $Pr(I_p|\mathcal{O}),Pr(I_p|\mathcal{B})$ は，このヒストグラムを用いて次のように割り当てる．

$$Pr(I_p|\mathcal{O}) = \theta(I_p,\mathcal{O}) \quad (6)$$

$$Pr(I_p|\mathcal{B}) = \theta(I_p,\mathcal{B}) \quad (7)$$

n-link には，式 (5) により隣接画素が似ていると大きい値を，似ていないと小さい値を設定する．λ は，t-link と n-link の強さを調整するパラメータであり，ユーザが経験的に決定する．

このようにして構成したグラフ G の最小カット C を，最大フローを見つけることで求め (図 2)，ソース s と接続する頂点のラベルを”obj”，シンク t と接続する頂点のラベルを”bkg”とすることで，式 (1) のエネルギーを最小化でき[1]，画像のセグメンテーションが実現できる．

2.3 従来手法の問題点

グラフカットによる画像セグメンテーションでは，グラフの頂点数が画像の画素数に対応するため多大なメモリを消費する．例えば，Boykov と Kolmogorov による実装では，$24|V|+14|E|$ バイトのメモリを必要とする．医用 CT 画像でよく用いられる 512^3 の 3 次元ボリュームでは，必要なメモリは $8GB$ を超える．これは現在の計算機では実用的な値ではない．さらに，最大フローを求める BK アルゴリズムの時間計算量の上界は $O(|E||V|^2|C|)$ であり[3]，このような大きさの画像を扱うには実用的ではない．

3. 提 案 手 法

本研究では，縮小した画像に対してプレセグメンテーションをおこない，その結果をグラフの辺の重みに反映させ，分割した画像でグラフカットを行うことで，高速かつ省メモリなグラフカット画像セグメンテーション法を提案する．

3.1 提案手法の流れ

図 3 に，提案手法の流れを示す．提案手法は次の三つのフェイズからなる．

フェイズ 1 pre-segmentation
フェイズ 2 uniform partitioning
フェイズ 3 adaptive halfway merging

はじめに，ユーザが入力画像に対して物体と背景の seed を入力する．次に，入力画像を一定の大きさにまで縮小し，watershed アルゴリズム[6] により seed を用いたプレセグメンテーションを行う (図 3(a))．このプレセグメンテーションの結果から，物体領域と背景領域の色分布を計算する．続いて，元画像を縦横均等に分割して (図 3(b))，それぞれの領域でグラフカットを実行する (図 3(c))．seed を含まない領域や，物体 (背景) が少しだけ含まれているような領域ではセグメンテーションの精度が悪いため，セグメンテーション精度が低い領域を結合し，新たにグラフカットを行う (図 3(d))．

3.2 フェイズ 1： pre-segmentation

画像の長辺が一定値 l 以下になるまで，画像のサイズを 1/4 にするダウンサンプリングを繰り返す．縮小した画像に対し，ユーザーにより与えられた seed を用いて watershed アルゴリズム[6] によりプレセグメンテーションを行う (図 3(a))．watershed によるプレセグメンテーションは，縮小した画像で行うので極めて高速に実行できる．このプレセグメンテーションの結果を用いて，前景・背景の色分布を GMM(Gaussian Mixture Model) に当てはめ計算する．本手法では縮小した画像で色分布の計算を行うので，元画像のサイズで色分布を計算する従来法に対して高速であり，ユーザにより選択された seed のみで色分布を計算する従来法と違い，プレセグメンテーションでの前景・背景の結果をもとに色分布を計算するので，より正確な色分布が計算できる．

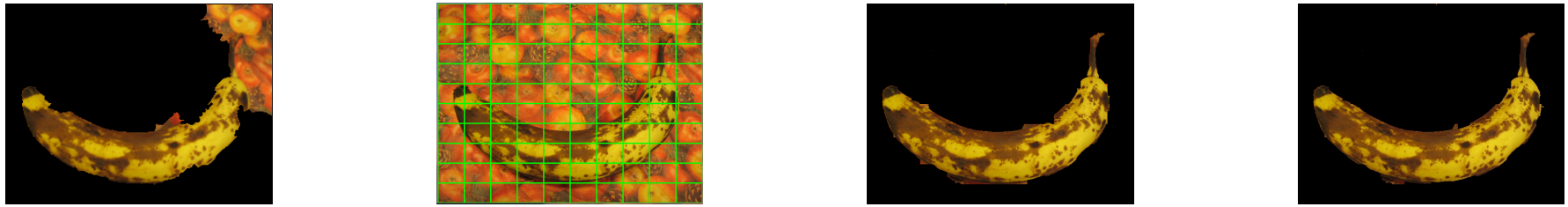
3.3 フェイズ 2： uniform partitioning

図 3(b) のように元画像を縦横 n 等分した素な領域に分割する．続いて，CPU が利用できるコアの数と同じだけスレッドを生成し，各領域でグラフを構築してグラフカットによりセグメンテーションを実行する．この分割は素な分割なため，それぞれの領域を並列に処理することができる．

3.3.1 グラフの構築

各スレッドが領域内のグラフを作成するときに，プレセグメンテーションの結果を t-link の重みに反映させる．具体的には，プレセグメンテーションで前景だった画素には，s 側の t-link を k 倍し，背景だった画素では，t 側の t-link を k 倍する．しかし，図 3(a) のように，プレセグメンテーションで間違ったセグメンテーションをすることもあるため，プレセグメンテーションで前景になった画素でも，色分布的には背景となる画素については，s 側

Vol.2011-CVIM-177 No.22
2011/5/19



(a) (b) (c) (d)

図 3 提案手法の流れ．(a)watershed によるプレセグメンテーション．(b) 画像を均等に分割．(c) それぞれの領域でグラフカット．(d) 精度の低い領域を結合して再度グラフカット
Fig. 3 flow of proposed method. (a)watershed pre-segmentation. (b)uniform partitioning. (c)run Graph cuts in each region. (d)merge the region that result is low and run Graph cuts

表 2 辺に割り当てる重み
Table 2 weights of edges

edge	weight	for
(p,q)	$B_{(p,q)}$	$(p,q) \in N$
(s,p)	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
	$\lambda \cdot R_p(\text{"}bkg\text{"}) \cdot 1/k$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}, p \in \mathcal{O}_{pre}, R_p(obj) > R_p(bkg)$
	$\lambda \cdot R_p(\text{"}bkg\text{"}) \cdot k$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}, p \in \mathcal{O}_{pre}, R_p(obj) \leq R_p(bkg)$
	$\lambda \cdot R_p(\text{"}bkg\text{"})$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}, p \in \mathcal{B}_{pre},$
(p,t)	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$
	$\lambda \cdot R_p(\text{"}obj\text{"}) \cdot 1/k$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}, p \in \mathcal{B}_{pre}, R_p(bkg) > R_p(obj)$
	$\lambda \cdot R_p(\text{"}obj\text{"}) \cdot k$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}, p \in \mathcal{B}_{pre}, R_p(bkg) \leq R_p(obj)$
	$\lambda \cdot R_p(\text{"}obj\text{"})$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}, p \in \mathcal{O}_{pre}$

の t-link を $1/k$ 倍にする．プレセグメンテーションで背景になった画素についても同様である．以下の表 2 に，グラフの各辺に割り当てる辺の重みを示す．

表 2 中の $\mathcal{O}_{pre}, \mathcal{B}_{pre}$ はそれぞれプレセグメンテーションで前景，背景になった画素の集合で，k は，プレセグメンテーションの結果をどれだけ強く t-link に反映させるかを表すパラメータで，大きいほどグラフカットによるセグメンテーション結果とプレセグメンテーションによる結果が近くなる．

図 4 は，同じ画像に同じ seed を与え，k の値を変化させたときの結果を示している．k が大きいときは (図 4(b))，グラフカットの結果とプレセグメンテーションの結果が同じになる．適切な k の値を設定することで (図 4(c))，より精度の高いセグメンテーションが得られる．

(a) (b) (c)

図 4 k の値による変化．(a)watershed によるプレセグメンテーション.(b)k=100 での結果.(c)k=2 での結果
Fig. 4 change of value k. (a)watershed pre-segmentation.(b)result of k =100. (c)result of k=2

適切な k の値を決めるために，Microsoft Reserch[7] で公開されている画像から 5 枚の画像を選び，k の値を変化させて精度がどのように変化するかを確認した．セグメンテーションの精度の計算は式 (9) を用いる．図 5 に，k の値を 1 から 20 まで変化させたときのセグメンテーション精度の変化を示す．

適切な k の値は画像にもよるが，実験により 2〜5 程度の値でより良いセグメンテーションの結果が得られることがわかった．

3.4 フェイズ 3：adaptive halfway merging

フェイズ 2 の uniform partitioning では，元画像を均等な領域に分割しそれぞれの領域

Vol.2011-CVIM-177 No.22
2011/5/19

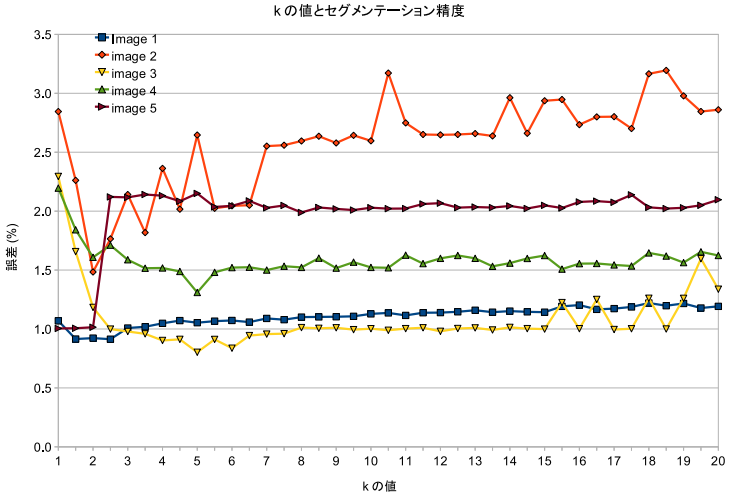
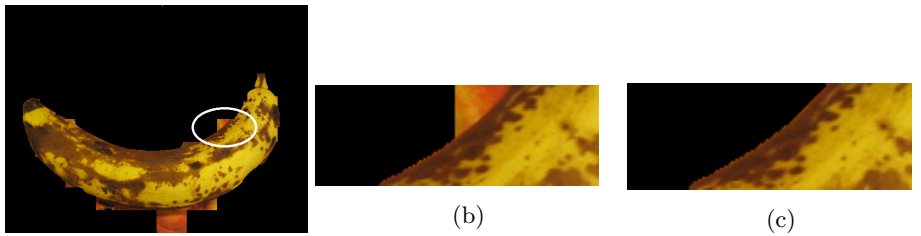


図 5 k の値とセグメンテーションの精度
Fig. 5 value of k and segmentation result

でグラフカットを行うため大域解は得られず，わずかに”物体 (背景)”である画素を含むような領域では，領域単位でセグメンテーションを誤ることがある．そのような領域の境目では直線的な誤識別が生じる (図 6(a))．このような領域では，頂点と，境目の反対側の頂点が異なるターミナルに属し，境界の辺を復元することで多くの s-t パスが見つかり，領域を結合して新たにグラフカットを行うことで大域解に近づけることができる．そこで，境界の反対側の頂点と異なるターミナルに属する境界の頂点を多く含む二つの領域を結合して新たにグラフカットを行う．adaptive halfway merging により，直線的なセグメンテーション誤りは少なくなる．図 6 に，adaptive halfway merging の効果を示す．

4. 実験比較

提案手法と従来手法の実行時間，メモリ使用量，精度を実験により比較する．実験は，CPU : core i3 530 2.93GHz，メモリ 4GB，32-bit Windows の環境下で行った．提案手法のパラメータは，$k = 2$, 分割数 $n = 8$(領域数=64)，スレッド数を 4，従来法と共通のパラメータ $\lambda = 2$ として実験を行った．



(a) (b) (c)

図 6 adaptive halfway merging の効果．(a) フェイズ 2 の直後.(b) 領域の境目で精度が悪い.(c) 結合して再度グラフカット
Fig. 6 effect of adaptive halfway merging. (a)just after phase 2.(b)segmentation result is bad in the vicinity of boundary. (c)merge regions and re-graph cuts

4.1 実行時間

1600x1200 の画像に対して，従来法と提案手法を実行したときの実行時間の内訳を図 7 に示す．図 7 の Pre-segmentation の項目は，フェイズ 1 のプレセグメンテーションに要した時間である．従来法ではプレセグメンテーションを行わないので 0(ms) となっている．提案手法の場合でも，縮小した画像でプレセグメンテーションを行うため実行時間は極めて短く，この実験ではおよそ 32(ms) であった．GMM の項目は，GMM の計算に要した時間で，縮小した画像で GMM を計算する提案手法の方が，元画像の大きさで計算する従来法よりも高速になっている．graph cut の項目は，グラフを構築してグラフの最小カットを計算するのに要した時間である．グラフの構築にかかる時間はどちらの手法も変わらないのだが，提案手法では分割した素な領域を並列に処理できるため従来法より高速になっている．

4.2 メモリ使用量

実験に用いた Boykov と Kolmogorov による BK アルゴリズムの実装では，グラフの構築に $24|V| + 14|E|$ バイトのメモリを消費する．1600x1200 の画像に対して，従来法ではグラフの構築におよそ $430MB$ のメモリを必要とした．提案手法では，画像を図 3(b) のように縦横 n 等分に分割し，それぞれの領域でグラフを作るため，フェイズ 2 でグラフ構築に必要なメモリは最大で

$$\frac{\text{従来法で必要なメモリ}}{n^2} \cdot \text{スレッド数} \tag{8}$$

となり，フェイズ 3 では一度に結合した二つ分の領域のグラフしか作らないので，必要なメモリは式 (8) のたかだか 2 倍となる．

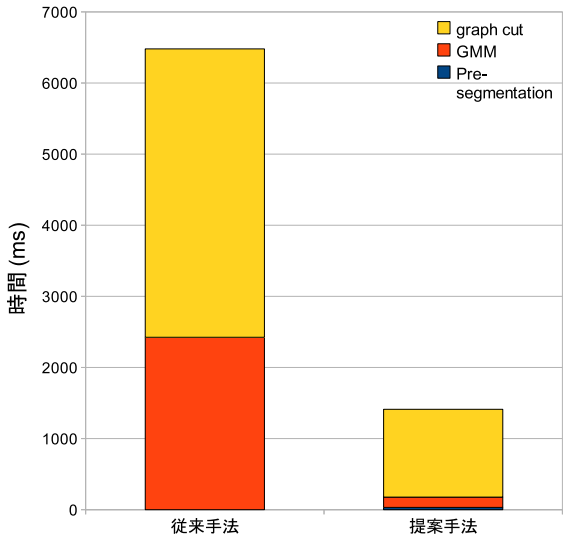


図 7 グラフカットの実行時間比較
Fig. 7 comparison of Graph cuts time

4.3 精 度 比 較

提案手法により，グラフカットの高速，省メモリ化は達成されたが，分割した領域でのグラフカットしか行わないため式 (1) の大域解は求まらない．セグメンテーション精度を保持するため，プレセグメンテーションにより大域的な色分布を学習しているが，その効果は明らかになっていない．提案手法のセグメンテーション精度を確かめるため，精度比較の実験を行った．

精度比較の実験には Microsoft Reserch[7] で提供されている 50 枚の画像を用いた．それぞれの画像に同じ seed を与えて提案手法と従来法[1),2)] との精度比較を行う．評価の方法として，正解データとの誤差を以下の式で求めたものを用いる．

$$error = \frac{\text{セグメンテーションを誤った画素数}}{\text{全画素数}} \tag{9}$$

表 3 に，50 枚の画像に対して式 (9) により評価した誤差の平均と，図 8 に，いくつかの

表 3 従来法と提案手法の精度比較
Table 3 Comparing our method with conventional approach in segmentation quality

	50 枚の画像に対する平均誤差 (%)
従来法	2.27
提案手法	2.16

画像のセグメンテーション結果を示す．表 3 より，提案手法はプレセグメンテーションにより従来法と大差ないセグメンテーション精度を達成していることがわかる．

5. まとめと今後の課題

5.1 ま と め

本研究では，縮小した画像に対してプレセグメンテーションを行い，その結果を元に前景・背景の色分布を計算して，元画像を素な領域に分割して並列にグラフカットを実行することで，高速かつ省メモリなグラフカット画像セグメンテーション法を提案した．また，セグメンテーション精度の低い領域を結合して再度グラフカットをおこなうことで，従来法と大差ないセグメンテーション精度を達成した．

5.2 今後の課題

今後の課題として，グラフカットに用いる各パラメータの自動決定が挙げられる．プレセグメンテーションの結果をどれだけ t-link に反映させるかを表すパラメータ k，画像をどれだけ分割するかを表すパラメータ n は，各画像によって最適な値がことなる．例えば，プレセグメンテーションがあまり正確でない場合に k の値を大きくし過ぎると，結果としてグラフカットセグメンテーション自体の精度も低下する．各画像の edge や corner，テクスチャ特徴量等により，これらのパラメータを Adaboost[8)] により決定する方法が考えられる．

謝辞 本稿を書くにあたり、有益なコメントと適切なアドバイスを与えて頂いた福嶋慶繁先生に深く感謝する

参 考 文 献

1) Boykov, Y.Y. and Jolly, M.-P.: Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images, *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001).
2) Rother, C., Kolmogorov, V. and Blake, A.: GrabCut ": Interactive Foreground Extraction using Iterated Graph Cuts, *ACM Transactions on Graphics* (2004).
3) Boykov, Y. and Kolmogorov, V.: An Experimental Comparison of Min-Cut/Max-

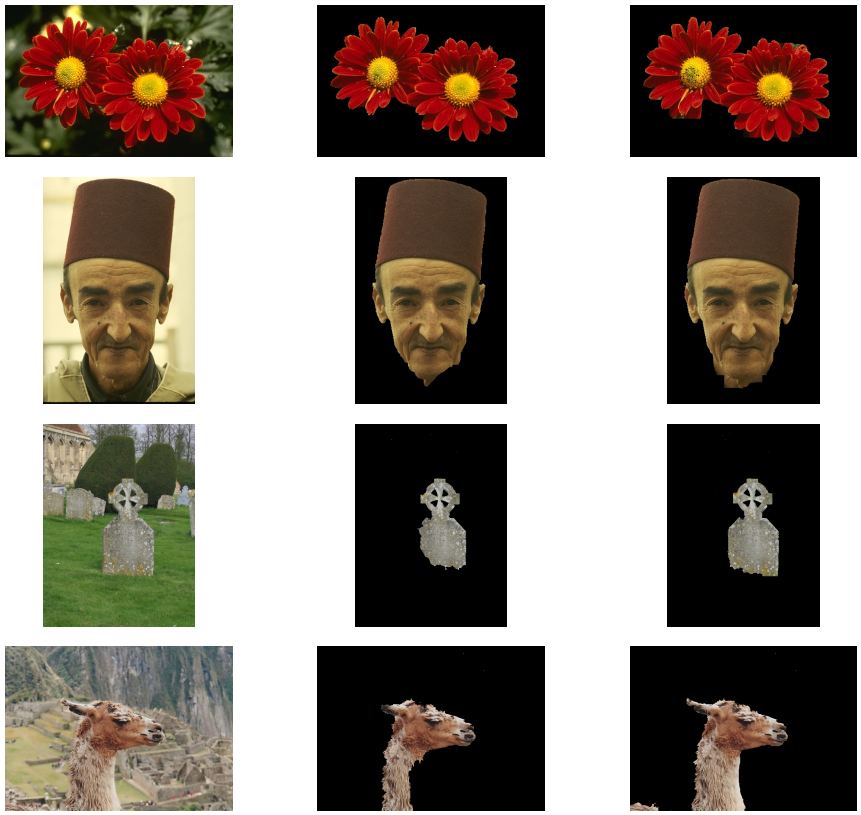


図 8 セグメンテーションの例
Fig. 8 ex. segmentation result

Flow Algorithms for Energy Minimization in Vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2001).
4) Lombaert, H., Sun, Y., Grady, L. and Xu, C.: A multilevel banded graph cuts method for fast image segmentation, In ICCV '05 (2005).
5) Liu, J. and Sun, J.: Parallel graph-cuts by adaptive bottom-up merging, *Computer Vision and Pattern Recognition* (2001).
6) Meyer, F.: Color image segmentation, *Image Processing and its Applications* (1992).
7) : Microsoft Reserach Cambridge, Image and Video Editing. `http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm`.
8) Freund, Y. and Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting, *European Conference on Computational Learning Theory* (1995).