解説：GrabCut の OpenCV 実装

2016/03/27

ビジョン&IT ラボ　皆川卓也

1. 概要

GrabCut は Rother らによって提案された、画像の前景と背景を Graph Cut を用いた繰り返し処理により分離する手法です。

Rother, C., Kolmogorov, V., & Blake, A. (2004). "GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts. *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*.

OpenCV では imgproc モジュールに cv::grabCut として実装されています。本書は、この OpenCV 内で実装されている GrabCut のアルゴリズムについての日本語解説資料です。なお、コードはバージョン 3.1 で確認しました。

GrabCut では、まず前景と背景の初期境界をユーザが大まかに指定し、次にそれぞれの色分布を基に Graph Cut で前景と背景の境界を求めます。この新しい境界から求めた前景と背景の色分布計算と Graph Cut による境界算出を繰り返し行うことで、前景と背景を求めます。

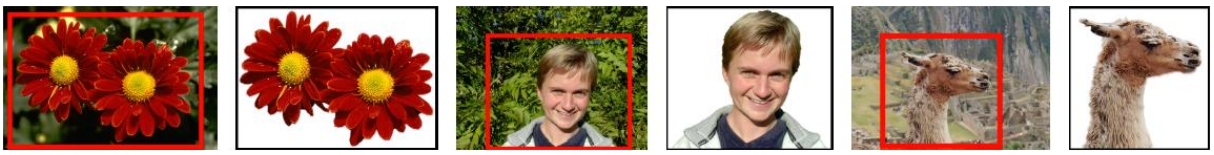


図1 GrabCut の例

1.1. 定式化

画像を構成する N 個の画素の色ベクトルを$\boldsymbol{z} = (z_1, \ldots, z_N)$とします。
色分布は混合ガウス分布でモデル化し、画素 n が前景、または背景に含まれる確率を eq.(1)で表します。

$$p(z_n) = \sum_{k=1}^{K} \pi_k \, N(z_n|\mu_k, \Sigma_k) \qquad \textbf{eq. (1)}$$

ここで$N(z_n|\mu_k, \Sigma_k)$は平均μ_k、分散Σ_kのガウス分布で、π_kは混合係数で各ガウス分布の大きさを表します。

ただし、ここでは簡単のため以下の制約を設けます。

- 前景用と背景用、２つの混合ガウス分布を用意します。
- 各画素に前景か背景かを表す値α_nを割り当て、$\alpha_n = 1$の画素は前景、$\alpha_n = 0$の画素は背景とし、それぞれ対応する混合ガウス分布でのみ評価されます。
- 画素z_nは混合ガウス分布を構成する K 個のガウス分布のうちどれか１つに割り当てられるとします。

それぞれの画素が、前景と背景どちらに属するかを表すベクトルを$\underline{\alpha} = (\alpha_1, \dots, \alpha_N)$とします。また、それぞれの画素が何番目のガウス分布に割り当てらえるかをベクトル$\mathbf{k} = \{k_1, \dots, k_N\}$（ただし、$k_n \in \{1, \dots, K\}$）で表します。

この時、平均μ、分散Σ、混合率πをα_nとk_nの関数とみなすことで、eq.(1)を次のように書き換えます。

$$p(\alpha_n, k_n, z_n) = \pi(\alpha_n, k_n) \, N\big(z_n|\mu(\alpha_n, k_n), \Sigma(\alpha_n, k_n)\big) \qquad \textbf{eq. (2)}$$

次に、前景/背景の領域分割の問題をエネルギーE の最小化問題と捉え、以下の式で表します。

$$E\big(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}\big) = U\big(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}\big) + V\big(\underline{\alpha}, \boldsymbol{z}\big) \qquad \textbf{eq. (3)}$$

ここで、U はデータ項と呼ばれ、前景/背景の色分布と画素の色を基に決定するエネルギーです。また Vは平滑化項と呼ばれ、隣り合う画素の持つラベルαが近いほど（同じ領域に属するほど）エネルギーが低くなる関数です。

$\underline{\theta}$は混合ガウス分布のパラメータを表します。

$$\underline{\theta} = \{\mu(\alpha, k), \Sigma(\alpha, k), \pi(\alpha, k), \alpha = 0,1, k = 1, \dots, K\} \qquad \textbf{eq. (4)}$$

データ項は eq.(2)の負の対数尤度を全画素で総和をとった値であり、以下の式で表せます。

$$\begin{aligned} U\big(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}\big) &= \sum_{n=1}^{N} D\big(\alpha_n, k_n, \underline{\theta}, z_n\big) \\ &= -\sum_{n=1}^{N} \log N\big(z_n|\mu(\alpha_n, k_n), \Sigma(\alpha_n, k_n)\big) - \sum_{n=1}^{N} \log \pi(\alpha_n, k_n) \end{aligned} \qquad \textbf{eq. (5)}$$

また、平滑化項は以下の式で表します。

$$V(\underline{\alpha}, \boldsymbol{z}) = \gamma \sum_{(m,n)\in \boldsymbol{C}} [\alpha_n \neq \alpha_m] \exp(-\beta \|z_m - z_n\|^2) \qquad \textbf{eq. (6)}$$

すなわち、前景と背景の境界(隣り合う画素(m,n)において$(\alpha_n \neq \alpha_m)$では、境界を挟んで隣り合う画素の色が異なるほどエネルギーが小さくなります。

2. 処理の流れ

GrabCut では入力パラメータとして、以下を与えます。

- 切り出したい対象物を囲んだ矩形 R
- 入力画像の一部に前景/背景のラベルを付けたマスク M
- 繰り返し回数 T

また以下のパラメータはコード内で固定されています。

- 混合ガウス分布の混合数$K = 5$
- eq.(6)の$\gamma = 50$
- 入力画像の隣り合う画素の二乗距離の平均をκとすると eq.(6)の$\beta = 1/2\kappa$

処理の流れは以下の通りです。

1. 初期化
    1. 矩形 R の外側を背景領域として$\alpha_n = 0$、内側は前景候補領域として$\alpha_n = 1$とします。または、マスク M で指定した背景領域を$\alpha_n = 0$、前景領域を$\alpha_n = 1$とします。
    2. $\alpha_n = 0$、 $\alpha_n = 1$それぞれの領域で混合ガウス分布（GMM）を推定します。
        1. 各領域内の色を k-means で K 個のクラスへクラスタリングします。
        2. 各クラス内の画素を基に eq.(7)と eq.(8)によって混合ガウス分布のパラメータ$\underline{\theta}$を割り当てます。
2. 繰り返し処理によるハードセグメンテーション
    1. 前景ピクセル n を GMM 中の K 個のガウス分布のうち、どれに割り当てるかを決めます。（最小のエネルギー＝最大尤度を持つガウス分布）

$$k_n := \underset{k_n}{\operatorname{argmin}} D(\alpha_n, k_n, \underline{\theta}, z_n) \qquad \textbf{eq. (7)}$$

    2. 割り当てられたピクセルを元に，そのエネルギーを最小化（尤度最大化）するパラメータ$\underline{\theta}$を求めます。

$$\underline{\theta} := \underset{\underline{\theta}}{\operatorname{argmin}} U(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}) \qquad \textbf{eq. (8)}$$

3. min cut を使って前景領域から背景領域を分離します。
4. 1-3 を T 回繰り返します。

尚、Kolovinskiy らの提案手法では、最後に Border Matting という処理を行うことで輪郭部分をスムーズにする処理を入れていますが、OpenCV にこの実装はありませんでした。

3. パラメータ最適化

各画素の前景/背景割り当て$\underline{\alpha}$ が与えられた状態で、対応するガウス分布への割り当て $\boldsymbol{k}$が eq.(7)により求まったとき、eq.(8)は以下の流れで計算されます。

1. 各ガウス分布の平均$\mu(\alpha, k)$を割り当てられた画素の色の平均から算出
2. 各ガウス分布の精度行列$\Sigma(\alpha, k)$を割り当てられた画素の色の共分散行列から算出(共分散行列の逆行列)
3. 混合率$\pi(\alpha, k)$を以下の式で算出

$$\pi(\alpha = \alpha', k = k') = \frac{(\alpha = \alpha', k = k')\textit{の画素数}}{(\alpha = \alpha')\textit{の画素数}} \qquad \textbf{eq. (9)}$$

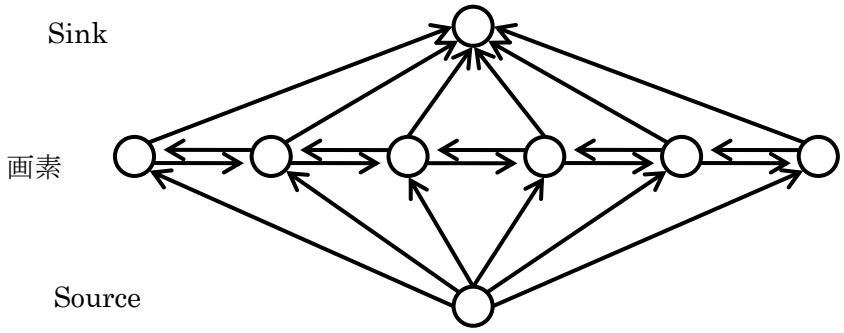
4. Min Cut

Min Cut は eq.(3)のような形で与えられるエネルギー関数を（劣モジュラの場合）大域的最小化となるようにラベルαを求めることが可能なアルゴリズムです。

処理の流れは以下の通りです。

1. グラフの生成

入力画像の各画素をノードとし、さらに Source/Sink の２つのノードを設定します。
各画像から隣接する画素に対しリンクを張ります。また Source から各画素へ、各画素から Sink へリンクを張ります。



2. ノードとリンクの重み設定

ノードの重みを eq.(5)、リンクの重みを eq.(6)に設定します。

3. Source/Sink からのリンク重み設定

Source から各ノードへの重みは、その画素の色を用いて背景 GMM から求めた負の対数尤度、各ノードから Sink への重みは、前景 GMM から求めた負の対数尤度です。

ただしユーザによって前景とラベル付けされた画素に関しては、Source からのリンク重みは9γ、Sink への重みは 0 です。逆に背景とラベル付けされた画素は、Source からのリンク重みは 0、Sink への重みは9γです。

4. 最小カットによる前景抽出

Source と Sink とが分離するように（Source から Sink へ向かう）リンクを切断してグラフを分割します。この時切断するリンクの重みの和が最小となるように最大フローアルゴリズムを用いて切断します。

その結果、Source 側にある点を前景、Sink 側にある点を背景とします。