Fast interactive segmentation using color and textural information

Olivier Duchenne[1] and Jean-Yves Audibert[1]

Research Report 06-26
September 2006

CERTIS, ParisTech - Ecole des Ponts,
77455 Marne la Vallee, France,

[1]CERTIS, ENPC, 77455 Marne la Vallee, France,http://www.enpc.fr/certis/

Fast interactive segmentation using color and textural information

Segmentation interactive basée sur la couleur et la texture

Olivier Duchenne[1] et Jean-Yves Audibert[1]

[1]CERTIS, ENPC, 77455 Marne la Vallee, France,http://www.enpc.fr/certis/

Abstract

This work presents an efficient method for image multi-zone segmentation under human supervision. As most of the segmentation methods, our procedure relies on an energy minimization and this energy contains a data-consistent term and a coherence term. In this work, the pixel-by-pixel data-consistent term is based on a Support Vector Machine designed in order to deal with color and textural information. For the coherence term, our contribution is to take into account the gradient information of the image through the use of a Canny edge detector. To minimize the energy, as in Blake et al. ([1]), we use a Graph-Cut algorithm. Finally, we obtain excellent segmentation results at low computational cost. We compare our results on an image data bank on-line[1].

[1] http://research.microsoft.com/vision/cambridge/i3l/segmentation/GrabCut.htm

Résumé

Ce travail présente une méthode efficace et rapide pour segmenter en plusieurs zones une image dans laquelle l'utilisateur a spécifié grossièrement un début de segmentation (voir p.3). Comme la plupart des méthodes de segmentation, notre procédure repose sur la minimisation d'une énergie composant un terme d'attache aux données et un terme de cohérence. Dans ce travail, l'attache aux données pixel par pixel est assurée par le score de Séparateurs à Vastes Marges ayant appris les couleurs et textures caractéristiques des différentes régions. Pour le terme de cohérence, notre contribution est de prendre en compte l'information de gradient de l'image à travers un détecteur de contours. Pour minimiser l'énergie, comme dans Blake et al. ([1]), nous utilisons un algorithme de coupe minimale. Finalement, nous obtenons d'excellents résultats de segmentation pour un temps de calcul faible. Nous comparons nos résultats sur une base de données accessible en ligne[2].

[2] `http://research.microsoft.com/vision/cambridge/i3l/segmentation/GrabCut.htm`

Contents

1 Introduction

To segment an image is to partition the image in regions having homogeneous meaning. For instance it may be used to separate an object from its background or to find the different areas (forest, field, montain, town) in satellite images. It has direct applications in painting software and may be used as a first step to detect, recognize, localize object, and more generally to understand the image.

However, state-of-the-art algorithms do not manage to segment images as perfectly as human brains. The reason is simple. To cut objects in the image, humans use meaning information. For instance, we know that animals have a head so we know that the red head of a turkey has to be segmented with the turkey even if their texture are radically different. Besides the brain combine multiple source of information to cut object such as form, texture, color, binocular vision. Meaning information are still rather hard to find, and is still an active research topic.

For a given image, there is no unique way of segmenting it. In this work, to circumvent this problem, we take the same approach as in [1]: we let a human user point out patches representative of the regions he wants to separate (see figure 1a).

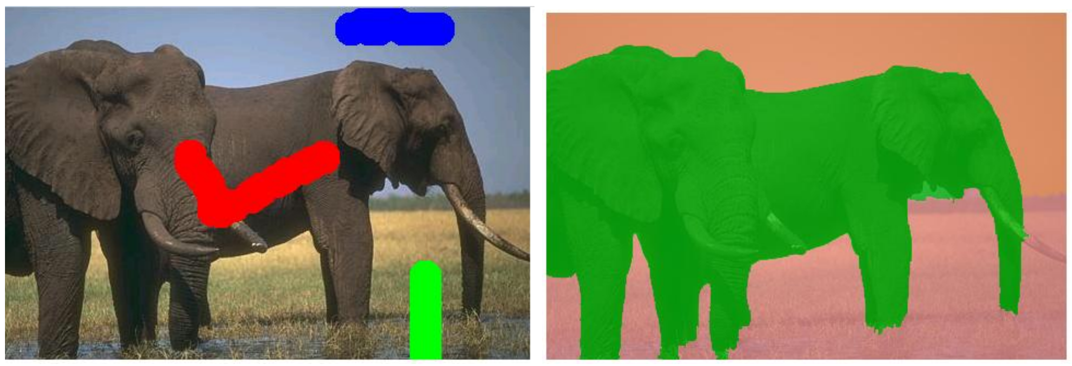


Figure 1: (a)To the left, patches put by the user to show the three parts he wants to segment. (b) To the Right the resulting segmentation

The segmentation algorithm will learn from the given examples in order to perform a meaning-consistent segmentation of the entire image coherent with the patches of the user (see figure 1b). However for one given set of patches there is often several possible segmentations. For instance if the user wants to segment a character on a photo he could have forgotten to show where he wanted the object this one has in hand to be cut. The algorithm will give one of the possible segmentation. If the user is not satisfied he can add new patches to be more explicit. The algorithm computes another segmentation. And it loops until the user obtains the results he wanted. However a good segmentation algorithm is supposed to be able to find it at the first iteration for most of the time.

The paper is organized as follows. Section 2 gives an overview of one iteration in which an energy is minimized. Section 3 and 4 specify the first and second order terms of the energy. Section 5 is dedicated to experimental results.

2 Overview of the segmentation procedure

Before giving the overview of our segmentation method, let us present formally the segmentation problem. Let us consider a pixel set $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ and a label set $\mathcal{L} = \{l_1, l_2, ..., l_n\}$. The segmentation problem consists in assigning a label $\lambda_p \in \mathcal{L}$ to each pixel $p \in \mathcal{P}$. For any pixel p, we assume that we have a subset $\mathcal{V}_p$ in $\mathcal{P} - \{p\}$ such that $q \in \mathcal{V}_p$ implies $p \in \mathcal{V}_q$. The typical content of the neighbourhood $\mathcal{V}_p$ is the set of points close to p up to some distance threshold. For any set $A \subset \mathcal{P}$, let λ_A denote the set $\{\lambda_a : a \in A\}$. To segment an image, we will minimize the energy

$$E(\lambda_{\mathcal{P}}) = \sum_{p\in\mathcal{P}} D_p(\lambda_p) + \sum_{p\in\mathcal{P}\,,\,q\in\mathcal{V}_p} V_{p,q}(\lambda_p, \lambda_q), \tag{1}$$

in which D_p and $V_{p,q}$ are functions allowed to depend on the image colors. In our procedure, D_p is a pixel-by-pixel data consistent term to the extent that it ensures that a pixel should be labelled coherently with image data. For instance, a blue pixel labelled as sky would give a lower energy than a green pixel labelled as sky. The second order term $V_{p,q}$ leads to a coherent labelling. For instance a pixel should not be labelled foreground if all his neighbours are labelled background. So we will start from this formalization of the problem, and try to give well-chosen D_p and $V_{p,q}$ to obtain efficient segmentations. To compute the minimum energy labelling, we resort to the graph cut method ([8]), which is known to be fast and to obtain the exact minimum at least in the two-labels setting. In the more than 2 regions segmentation task, we use the α-expansion technique (see [2] and also e.g. [7, Chapter 3]).

The main steps of the algorithm are described below and presented in figure 3.

- Extract square patches on a regular thin grid of the image
- Extract features (see Section 3.1) from all those patches
- For each user-labelled region, from a Support Vector Machine (SVM), learn a score function to distinguish the features of the region from the features of all other regions. We have as many SVM as there are parts in the segmentation
- For each SVM, for each extracted square patch, give its associated score
- For each SVM, give the associated score to all pixels by bilinear extrapolation of the previously obtained scores
- Use an edge detector to compute a binary image of frontiers

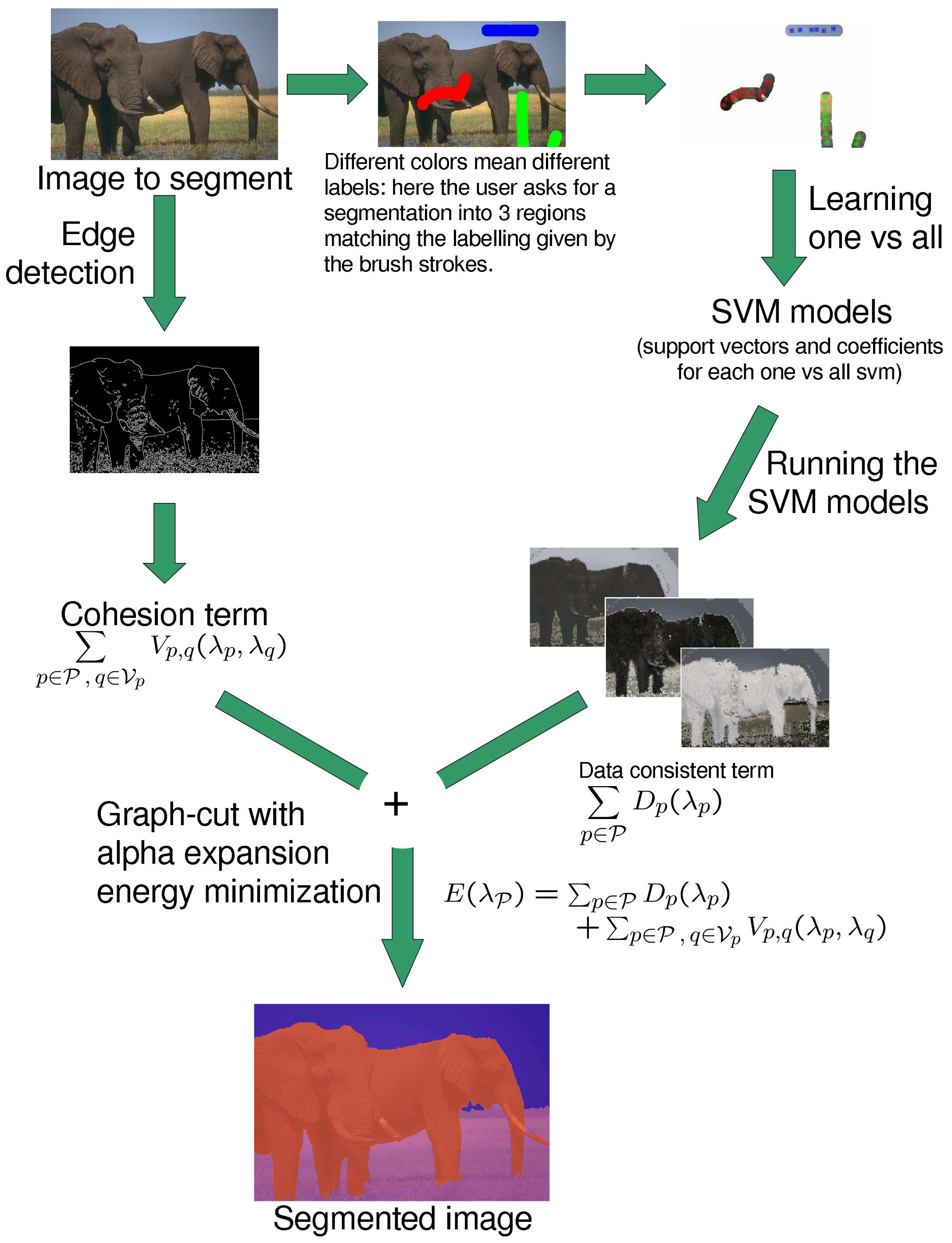


Image to segment
Different colors mean different labels: here the user asks for a segmentation into 3 regions matching the labelling given by the brush strokes.
Learning one vs all
SVM models
(support vectors and coefficients for each one vs all svm)
Edge detection
Running the SVM models
Cohesion term
$\sum_{p\in\mathcal{P}, q\in\mathcal{V}_p} V_{p,q}(\lambda_p, \lambda_q)$
Data consistent term
$\sum_{p\in\mathcal{P}} D_p(\lambda_p)$
+
Graph-cut with alpha expansion energy minimization
$E(\lambda_\mathcal{P}) = \sum_{p\in\mathcal{P}} D_p(\lambda_p) + \sum_{p\in\mathcal{P}, q\in\mathcal{V}_p} V_{p,q}(\lambda_p, \lambda_q)$
Segmented image

Figure 2: Overview of the segmentation procedure

- Combine score functions and edges to obtain the energy of a configuration
- Minimize this energy using a graph cut with α-expansion algorithm
- Segment the image associated to the minimum obtained

3 Color and texture learning

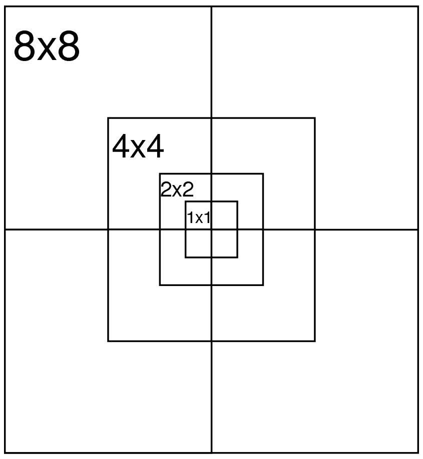
Most of state-of-the-art algorithms (e.g. [1, 6]) only use the color of the pixel p to calculate the pixel-by-pixel data dependent term D_p. Here we use the color and texture information (through the local representations proposed in Section 3.1), learn their characteristics in each region through an SVM (defined in Section 3.2) and convert the SVM scores into the pixel-by-pixel data dependent term given in Section 3.3.

3.1 Color and texture representation

There are several ways of describing the texture at a location in the image. Here we use square patches. These patches can be considered as a vector of colors with as many colors as there are pixels in the patches. This 'brute representation' can be replaced with a more compact one based on the color information at different scales.

Typically, a 16-by-16 square patch can be represented through

- the colors of the four central pixels,
- the mean colors of the four central 2-by-2 square around the point,
- the mean colors of the four central 4-by-4 square around the point,
- the mean colors of the four central 8-by-8 square around the point (see figure).



More generally, this 'compact representation' can be used for any square of size 2^j with $j \in \mathbb{N}^*$. In this representation, the learning algorithm will decide at which scale the color is pertinent. For instance, if you try to distinguish black and white stripes from unvarying grey, the smaller scale is the important one: at this scale, colors are black or white in one case and grey in the other. However if we try to separate black and white stripes from unvarying white, a larger scale is the important one: at this scale, colors are grey in one case and white in the other. Besides those features have the advantage to take far less coordinates: '4 scales x 2x2 squares'=16 colors for a 16x16 square instead of 256 colors for the 'brute representation'. Finally thanks to the integral image trick, the computations of these features takes only few computational resources. Both representations have been tested in the numerical experiments (see Section 5).

3.2 Gaussian kernel SVM with no constant term

Let $\mathcal{X}$ denote the set of possible representations of the neighbourhood of a pixel. Typically, $\mathcal{X} \subset \mathbb{R}^{256}$ for a brute representation of a 16x16 square. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric definite positive function, i.e. a real-valued function such that for any $m \in \mathbb{N}$, $\alpha_1, \dots, \alpha_m \in \mathcal{X}$ and $x_1, \dots, x_m \in \mathcal{X}$, we have $\sum_{1 \le j,k \le m} \alpha_j \alpha_k k(x_j, x_k) \ge 0$ and this sum is zero iff all the α_j's are zeroes. The function k is often referred to as a Mercer kernel. On a training set $(x_1, \dots, x_n, y_1, \dots, y_n) \in \mathcal{X}^n \times \{-1; +1\}^n$, the output of the standard SVM with Mercel kernel k and trade-off constant $C > 0$ is the prediction function sign$\big(\sum_{i=1}^n \alpha_i k(x_i, x) + b\big)$, where the α_i's are the solution of

$$\max_{\substack{\sum_{i=1}^n \alpha_i = 0 \\ 0 \le \alpha_i y_i \le C}} \sum_{i=1}^n \alpha_i y_i - 1/2 \sum_{1 \le i,j \le n} \alpha_i \alpha_j k(x_i, x_j), \tag{2}$$

and b is obtained by $b = y_j - \sum_{i=1}^n \alpha_i k(x_i, x_j)$ for (any) j such that $\alpha_j \neq 0$. When one wants to assess the confidence in the prediction of the output associated with the input x, one uses the margin $\big|\sum_{i=1}^n \alpha_i k(x_i, x) + b\big|$, which can be viewed as the distance to the boundary in an appropriate feature space.

The gaussian kernel $k_\sigma : (x, x') \mapsto e^{-\|x-x'\|^2/\sigma^2}$ of width $\sigma > 0$ is the Mercer kernel we use in our experiments. With this kernel, a new vector very different from all the support vectors would have a score close to b Now b could have very different values and points which are far from any point in the training set should not be classified with high scores. So we modify the algorithm to optimize the SVM problem with the $b = 0$ constraint. This ensures that the answer of a SVM to unknown patches is the score meaning 'I do not know how to classify this patch'. For these pixels, the MRF model plays a central role and leads to a propagation of the labels between neighbouring pixels. Technically, the problem (2) becomes

$$\max_{0 \le \alpha_i y_i \le C} \sum_{i=1}^n \alpha_i y_i - 1/2 \sum_{1 \le i,j \le n} \alpha_i \alpha_j k(x_i, x_j). \tag{3}$$

In our experiments, this last problem is solved efficiently by gradient descent in the directions associated with each α_i individually.

3.3 The pixel-by-pixel data-dependent energy term

For each label $l \in \mathcal{L}$, consider the training set in which all patches in regions labelled l by the user are of class $+1$ and all the other patches in the regions labelled by the user are of class -1. The previously described SVM is then trained. For any new patch, it associates the quantity

$$\text{score}_l(x) = \sum_{i=1}^{n} \alpha_{i,l} k_\sigma(x_i, x)$$

which is 'all the more' positive as the SVM is confident that the patch should be labelled l. The center of a $2^j \times 2^j$ pixels square is at an interpixel location. On a regular grid of interpixels, we compute the associated score of the square centered at this location. By bilinear interpolation, we obtain the score at any pixel location. For $\gamma > 0$, let us consider the function ϕ_γ defined as for any $u \in \mathbb{R}$,

$$\phi_\gamma(u) = \frac{1}{1 + e^{-\gamma u}}.$$

The pixel-by-pixel data-dependent energy term is defined as

$$D_p(\lambda_p) = \phi_\gamma\big[\text{score}_{\lambda_p}(p)\big].$$

The more confident we are that pixel p should be in the class λ_p the less the energy $D_p(\lambda_p)$ is. The parameter $\gamma > 0$ is chosen empirically (see Section 5).

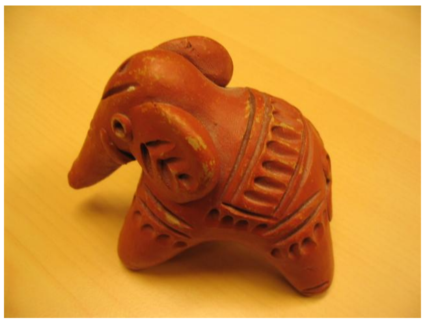
4 The cohesion term

Let us now define the cohesion terms $V_{p,q}(\lambda_p, \lambda_q)$ where p and q are neighbouring pixels. The cohesion term relies on the two following assertions:

- in general, neighbouring pixels have the same label
- for a given image, neighbouring pixels which are both not on a contour extracted by an edge detector are very likely to have the same label

The first assertion is not data-dependent (since it does not depend on the image to be segmented), and is quantified by $E_1 \mathbb{1}_{\lambda_p \neq \lambda_q}$ where E_1 is some positive constant. To take into account the second assertion, a known approach consists in weighting this term by the inverse of gradient intensity to favorize group labelling just for zone without big intensity change. This ameliorates the results but it is hard to calibrate.

For instance, the nearby figure is an easy to segment image. However the gradient norm is three times higher on the top of the elephant than on its bottom. And it is much higher on the marks on the statue. Therefore the gradient information is not very significant. It is rather impossible to give a good formula or threshold to calculate the cohesion term from local gradient norm.



To improve the segmentation results, we use the Canny edge detector which follows the gradient line even if the absolute value of gradient fall low. Let F be the set of pixels on the borders output by a Canny edge detector. We take

$$V_{p,q}(\lambda_p, \lambda_q) = E_1 \mathbb{1}_{\lambda_p \neq \lambda_q} + E_2 \mathbb{1}_{p \notin F, q \notin F, \lambda_p \neq \lambda_q}. \qquad (4)$$

5 Numerical experiments

We have tested our algorithm on the data set used in [1, 6]. This data set available online[3] contains 50 images. As in [1], we choose the parameters of our algorithm using 15 images, and test the resulting algorithm on the remaining 35.

We have used the same control test despite the slight differnce in the segmentation task. In the database, the user has drawn a big gray line on the border of the object he wants to cut. The algorithm uses the interior and the exterior as the learning parts and then segments the gray line. The quality of the segmentation is assessed through

$$\text{Error rate} = \frac{\text{number of misclassified pixels}}{\text{number of pixels in the gray line}}.$$

This problem is not very appropriate for our algorithm. First, in this database, there is often in the outer zone, part very similar to part in the gray zone which have to be classified foreground. Secondly as the learning patches have not been chosen to be representative, there can be parts which are bound to be arbitrarily classified. Finally the database did not consider images in which the background and the foreground have same colors but different textures. Our results are resumed in the following chart.

Segmentation model	Error rate
Optimization with Statistical priors ([6])	4.65%
Our algorithm	5.98%
GMMRF; discriminatively learned $\gamma = 20$ ($K = 10$ full Gaussian) [1]	7.9%
Learned GMMRF parameters ($K = 30$ isotropic Gauss.) [1]	8.6%
GMMRF; discriminatively learned $\gamma = 25$ ($K = 30$ isotropic Gaussian) [1]	9.4%
Strong interaction model ($\gamma = 1000$; $K = 30$ isotropic Gaussian) [1]	11.0%
Ising model ($\gamma = 25$; $K = 30$ isotropic Gaussian) [1]	11.2%
Simple mixture model – no interaction ($K = 30$ isotropic Gaussian) [1]	16.3%

Texture classification The following figure shows our segmentation of an artificial image. composed of three textures having each the same proportion of black and white colors. In this artificial setting, there is no way of segmenting correctly the image by looking at individual pixels, i.e. without taking into account the texture information.

Some other segmentation results are given in appendix.

[3] `http://research.microsoft.com/vision/cambridge/i3l/segmentation/GrabCut.htm`

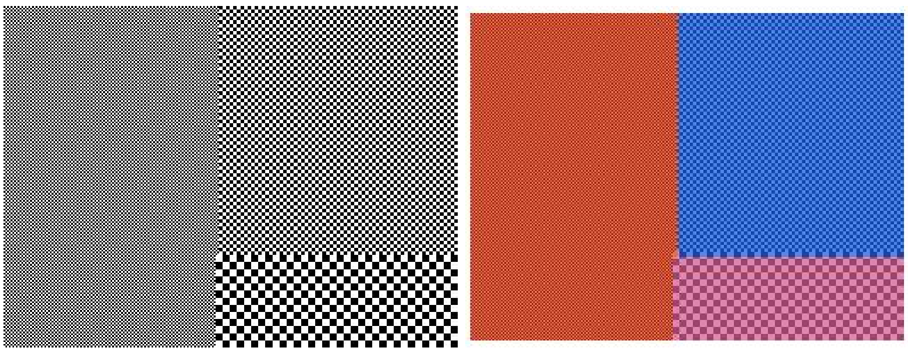


Figure 3: (a) Artificial textures. (b) Segmentation result.

6 Conclusion and future work

We have proposed a segmentation procedure learning simultaneously the color and texture of the different regions through a modified SVM algorithm applied to a compact multiscale representation of local patches. In our method, the gradient information is used through a Canny edge detector. The resulting algorithm gives similar results in the natural images of the benchmark database and it can cope successfully with challenging texture segmentation.

Future work may concentrate on the following weaknesses of our segmentation algorithm. We need

- to handle better the illumination variations. We already tried HSV, LAB and YUV but the resulting segmentations were comparable, and sometimes even worse.
- to better localize the border of the regions. When a patch is astride the border, the SVM scores are generally useless. In most cases, the coherence term allows us to find the correct segmentation. Nevertheless when the edge detector finds out many edges near the border (for instance when there are patterns on a tablecloth), the algorithm is unable to find consistently the edge where it has to stop.
- to have a Canny edge detector for color images. The one we use was considering the grey level images. As a result, on some images of the database, the edge detector failed to localize the border. We think that our results would be highly improved by the use of an color edge detector.
- to use an adaptive edge detector. The Canny edge detector needs thresholds as parameters. If they are too low, borders invade the whole image. When they are too high, there is no border. Worse, it is sometimes impossible to have one border you want, without having unwanted lines. So in our method we first place a high threshold. Then we segment and look for segment borders which are not lying on

a border of Canny edge detector. At this place, we extract a little image where we apply the edge detector at a lower threshold until getting a good border.

A Some segmentation results

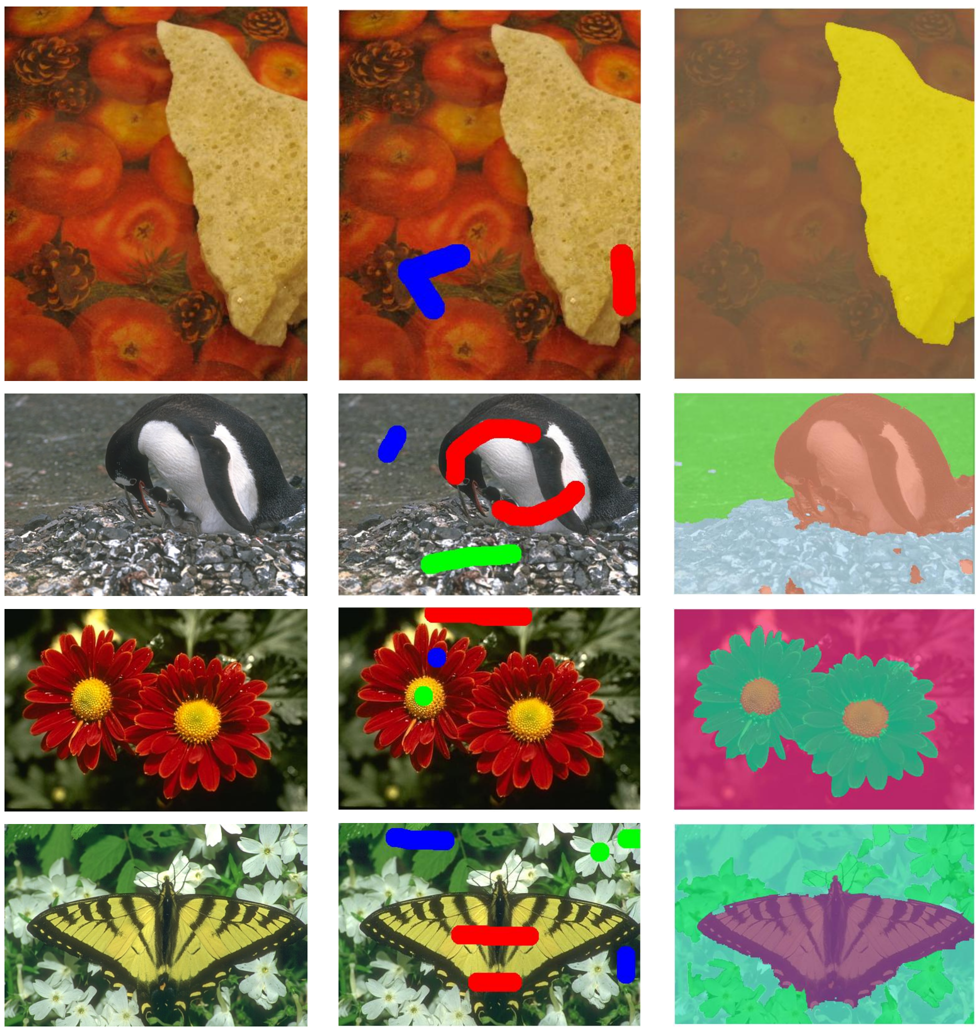


Figure 4: Results. to the left, start Image, then user patches, and finally segmentation

References

[1] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In ECCV, pages Vol I: 428–441, 2004.

[2] Y. Boykov, O. Veksler and R. Zabih, Fast approximate energy minimization via graph cuts. IEEE Transaction on Pattern Analysis and Machine Intelligence 23, 11 (2001), 1222–1239.

[3] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In ICCV, pages I: 105–112, 2001.

[4] J.F. Canny, A computational approach to edge detection. IEEE Trans Pattern Analysis and Machine Intelligence, 8(6): 679-698, Nov 1986.

[5] S. Geman and D. Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, PAMI(6), No. 6, November 1984, pp. 721-741.

[6] J. Guan and G. Qiu. Interactive Image Segmentation using Optimization with Statistical Priors. ECCV 2006.

[7] O. Juan, On some Extensions of Level Sets and Graph Cuts towards their applications to Image and Video Segmentation, PhD thesis, `http://cermics.enpc.fr/~juan/thesis.pdf`, 2006.

[8] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. IEEE PAMI, 26(2):147–159, 2004.

[9] J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press. 2004