Transductive segmentation

Olivier Duchenne[1], Jean-Yves Audibert[2] and Renaud Keriven[3]

CERTIS Research Report 06-27
also Willow Technical report 04-06
December 2006

Ecole des ponts - Certis
6-8 avenue Blaise Pascal
77420 Champs-sur-Marne
France

Inria - Rocquencourt
Domaine Voluceau-Rocquencourt
78153 Le Chesnay Cedex
France

Ecole Normale Supérieure - DI
45, rue d'Ulm
75005 Paris
France

[1]École Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France
[2]Willow Team, CERTIS-ENPC, 77455 Marne la Vallée cedex, France
[3]CERTIS-ENPC, 77455 Marne la Vallée cedex, France

Transductive segmentation

Segmentation transductive

Olivier Duchenne[1], Jean-Yves Audibert[2] et Renaud Keriven[3]

[1]École Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France
[2]Willow Team, CERTIS-ENPC, 77455 Marne la Vallée cedex, France
[3]CERTIS-ENPC, 77455 Marne la Vallée cedex, France

Abstract

We consider a multi-zone segmentation of a single image when user-supplied seeds are provided in each region. We view this task as a statistical *transductive inference*, in which some pixels are already associated with given zones and the remaining ones need to be classified. Our method relies on the Laplacian graph regularizer, a powerful manifold-learning tool that is based on the estimation of variants of the Laplace-Beltrami operator and that is tightly related to diffusion processes. Our segmentation is modeled as the task of finding matting coefficients for unclassified pixels given known matting coefficients of seed pixels. The resulting segmentation procedure is simple, fast, and accurate. Comparison with other methods on natural images databases are given.

Résumé

Nous considérons le problème de segmentation multi-zone d'une image lorsque l'utilisateur fournit des graines pour chaque région. Des pixels sont donc déjà étiquetés par la zone à laquelle ils appartiennent et nous souhaitons trouver à quelles zones appartiennent les autres pixels. Pour résoudre ce problème, nous utilisons un outil puissant d'inférence transductive : le laplacien de graphe. Cet outil puissant d'apprentissage est basé sur l'estimation de variantes de l'opérateur de Laplace-Beltrami d'une variété - opérateur fortement lié à des processus canoniques de diffusion sur la variété.

Notre algorithme permet de trouver les coefficients de mélange ("matting") de chaque pixel non étiqueté à partir des coefficients de mélange des pixels étiquetés fournis par l'utilisateur. La procédure en découlant est simple, rapide et précise et peut s'interpréter comme une diffusion partant des pixels étiquetés. Nous fournissons une comparaison avec les autres méthodes existantes sur des bases d'images naturelles.

Contents

1 Introduction **1**

2 Transductive viewpoint of image segmentation **2**
2.1 Transductive vs Inductive inference 2
2.2 The segmentation input space 2
2.3 Graph Laplacian method: the state-of-the-art transductive inference algorithm 3

3 Our segmentation algorithm **6**
3.1 Two-zone segmentation 6
3.2 Multi-zone segmentation 7
3.3 Segmentation with prior knowledge of the zones 7
3.4 Computational complexity 7

4 Link with previous segmentation approaches **8**
4.1 Min-cuts methods 8
4.2 Iterative segmentation methods 8
4.3 Guan and Qiu's approach 8

5 Experimental results **9**
5.1 Comparison with state-of-the-art algorithms 9
5.2 Discussion 10
5.2.1 geometric neighborhood 10
5.2.2 comparison with graphcut 10
5.2.3 computation time 11

6 Conclusion **11**

Bibliography **11**

1 Introduction

Image segmentation, the process of automatically partitioning an image into different homogeneous regions, is a fundamental task in a large number of applications in computer vision, medical imaging, ...For instance, it may be used to separate an object from its background (e.g. identification of specific anatomical structures in medical images, tracking of persons or objects in video sequences, ...), to identify different areas in images (forests, fields, mountains, towns in satellite images), it has direct applications in painting software (alpha matting for landscape recomposition, virtual reality, ...). More generally, image segmentation is one of the first fundamental steps toward better scene understanding and object recognition.

Yet, state-of-the-art algorithms do not segment images as efficiently as humans do. To identify objects in images, human beings rely on a combination of low-level information (e.g. color and texture) with high-level information (shape priors, semantic cues, ...). Images are corrupted by various perturbing factors such as noise, occlusions, missing parts, cluttered data, among others. The incorporation of high-level knowledge, which is often necessary to resolve the ambiguity inherent to the segmentation of complex images, is still a challenging and active research area.

The difficulty of the task is also increased by the non-uniqueness of a segmentation, i.e. there is no unique way of segmenting a given image. In this paper, we circumvent this problem by taking a *transductive learning* viewpoint; we follow the same approach as Blake et al. in [3]. A human user identifies small patches representative of the regions he wants to segment. Given those, our algorithm generates a meaning-consistent segmentation of the "entire" image that is coherent with the user-supplied patches.

Figure 1: a) original image to be segmented. (b) The resulting segmentation with our method

The paper is organized as follows. Section 2 presents our transductive view-

point of image segmentation and Section 3 describes our segmentation algorithm. Section 4 discusses the links with previous works. Section 5 gives experimental results.

2 Transductive viewpoint of image segmentation

2.1 Transductive vs Inductive inference

One of the main issues addressed by machine learning is labeling of new data points given set of labeled examples; classically, one observes input-output points and wants to derive from this database (i.e. the training set) the outputs associated with new inputs.

We usually distinguish two types of learning, inductive inference vs transductive inference. In inductive inference, the new points are not known beforehand, so that the algorithm has to deduce from the database a mapping from (all) the input space to the output space. When new inputs come in, the learned function maps them to corresponding outputs.

In transductive inference, the setup is different. At the beginning, we get simultaneously the training set and the input test set. As a consequence, the two steps, which consist in learning the input-to-output mapping and in consecutively using it on new test points, can be replaced by a single one: learning the output associated with the input test points. This more-direct approach follows Vapnik's principle: *do not try to learn more than necessary as an intermediate step*. In addition, one can use the input test set to "estimate" the input distribution, which turns out to be often useful, as this second key principle highlights: *outputs vary a lot only on input regions having low density*.

The example of figure 2 points out the advantage of transductive segmentation toward inductive one. We have to differentiate 2 families of points in $\mathbb{R}^2$. The learning sets are represented in 2.a. An inductive classifier would find a separation border in the middle of the two classes (2.c). Thanks to unlabeled points position shown in 2.b, the inductive method will find a separator localized at low density area. And the resulting edge is quite more satisfactory (2.d). Indeed, in natural images, unlabeled points features can change slowly as in 2.b, especially due to luminance effects.

2.2 The segmentation input space

In our multi-zone segmentation problem, the outputs associated with seed pixels are given; the inference problem consists in determining the outputs associated

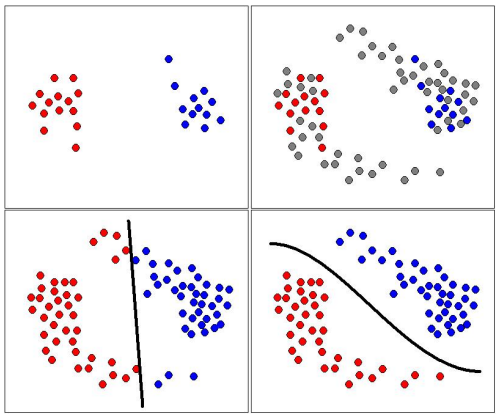

Figure 2: a) top-left, the learning sets of points. b) top-right, the unlabeled points in gray. c) bottom-left, the transductive algorithm separator and results. d) bottom-right, the transductive algorithm separator and results.

with the remaining pixels of the image. This is basically a transductive classification task in which the number of classes is the number of selected zones in the segmentation. In machine learning, the question of representation is of crucial importance. This is the first question we address, i.e. how should we represent image pixels? Our choice for the input space is motivated by the following conditions:

- pixels coming from the same zone should be well clustered,
- clusters coming from different zones should be well separated,
- geometric (position) as well as photometric (color, texture) information should be used to cluster pixels.

In order to capture the *texture* of the different objects, we associate to each pixel a local patch centered around it: the texture is then encoded by the color level of the patch. Finally, the pixel geometrical position is just encoded by the row and column of the pixel.

2.3 Graph Laplacian method: the state-of-the-art transductive inference algorithm

Graph Laplacian-based methods have emerged recently and have been successfully used in transductive inference ([2]), (spectral) clustering ([12]) and and dimensionality reduction ([1]).

The underlying assumption of these methods is the following: *the (input) points are generated by a probability distribution with support on a submanifold of the Euclidean space*. Let M denote this submanifold. Let p be the density of the input probability distribution with respect to the canonical measure on M (i.e. the one associated with the natural volume element dV). Note that M could be all the Euclidean space (or a subset of it of same dimension) so that p can simply be viewed as a density with respect to the Lebesgue measure on the Euclidean space.

In transductive inference, one searches for a smooth function f from the input space into the output space such that $f(X_i)$ is close to the associated output Y_i on the training set and such that the function is allowed to vary only on low density regions of the input space. Let $s > 0$ be a parameter characterizing how low the density should be to allow large variations of f, i.e. we consider a s-weighted version of the density p. *One of the main contributions of this work consists in carefully choosing the parameter s as well as the correct graph Laplacian.*

For the sake of clarity, let us consider a real-valued output space (such as the space of alpha-matting coeffents in a two-zone segmentation task [10]). Depending on the confidence we assign to the training outputs, we obtain the following optimization problem:

$$\begin{array}{l} \min\limits_{f} \sum\limits_{i \in \{\text{train pixels}\}} c_i [Y_i - f(X_i)]^2 \\ \qquad\qquad + \int_M \|(\nabla f)(x)\|^2 p^s(x) dV(x), \end{array} \tag{1}$$

where the c_i's, $\forall i \ c_i > 0$, are regularization coefficients measuring how much we want to fit the training point (X_i, Y_i). Typically, $c_i = +\infty$ imposes a hard constraint on the function f so that $f(X_i) = Y_i$. The s-th weighted Laplacian operator is characterized:

$$\int_M f \times (\Delta_s g)\, p^s dV = \int_M \langle \nabla f, \nabla g \rangle \, p^s dV,$$

where f, g are infinitely smooth real-valued functions defined on M and with compact support. By the law of large numbers, the integral in (1) can be then approximated by

$$\tfrac{1}{n} \textstyle\sum_{i=1}^n f(X_i) \Delta_s f(X_i) p^{s-1}(X_i).$$

Unfortunately, the direct computation of $\Delta_s f(X_i)$ for every possible function f is not possible and solving (1) is intractable.

Graph Laplacian methods, which are based on a discrete approximation of the s-th weighted Laplacian operator, propose a discrete alternative to this problem. The method is based on a neighborhood graph in which the nodes are the input points coming from both the training and test sets. Let $X_1, \ldots, X_n$ denote these points. Let $\tilde{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetrical function giving the similarity between two input points. The typical kernel k is the gaussian kernel

$\tilde{k}(x', x'') = e^{-\frac{\|x'-x''\|^2}{2\sigma^2}}$. The degree function associated with this first kernel is $\tilde{d}(x) = \sum_{i=1}^{n} \tilde{k}(X_i, x)$. Let $\lambda \geq 0$. General graph Laplacian methods use the normalized kernel defined as

$$k(x', x'') = \frac{\tilde{k}(x', x'')}{[\tilde{d}(x')\tilde{d}(x'')]^{\lambda}}. \tag{2}$$

For $\lambda = 0$, no normalization is done. For $\lambda = 1/2$, perfect normalization occurs. The degree function associated with this first kernel is $d(x) = \sum_{i=1}^{n} k(X_i, x)$.

The kernel k induces a weighted undirected graph in which the nodes are $X_1, \ldots, X_n$ and in which any two nodes are linked with an edge of weight $k(X_i, X_j)$. The degree of a node is defined by the sum of the weights of the edges at the node, i.e. $d(X_j)$.

Let W be the $n \times n$ matrix in which the generic element is $k(X_i, X_j)$. Let D be the diagonal $n \times n$ matrix for which the i-th diagonal element is $d(X_i)$. Finally let I be the identity matrix of size $n \times n$.

In the literature, three kinds of graph Laplacian are defined through their associated matrices:

$$\begin{array}{ll} \text{the random walk matrix:} & L_{\text{rw}} = I - D^{-1}W, \\ \text{the unnormalized matrix:} & L_{\text{un}} = D - W, \\ \text{the normalized matrix:} & L_{\text{n}} = I - D^{-1/2}WD^{-1/2}. \end{array}$$

For a given function $f : \mathcal{X} \to \mathbb{R}$, let F be the vector defined as $F_i = f(X_i)$. The main result of [7] is essentially

$$\begin{array}{rcl} (L_{\text{rw}}F)_i & \rightsquigarrow & \big(\Delta_{2(1-\lambda)}f\big)(X_i) \\ (L_{\text{un}}F)_i & \rightsquigarrow & [p(X_i)]^{1-2\lambda}\big(\Delta_{2(1-\lambda)}f\big)(X_i) \\ (L_{\text{n}}F)_i & \rightsquigarrow & [p(X_i)]^{\frac{1}{2}-\lambda}\Big[\Delta_{2(1-\lambda)}\big(\frac{f}{p^{1/2-\lambda}}\big)\Big](X_i) \end{array} \tag{3}$$

where $\rightsquigarrow$ means convergence almost sure when the sample of size n goes to infinity and the kernel bandwidth h goes to zero not too rapidly (e.g. $h = (\log n)^{-1}$), up to normalization of the left-hand side by an appropriate function of n and h. Besides one can understand the role of the degree functions through the convergences:

$$\begin{array}{rcl} \tilde{d}(x) & \rightsquigarrow & p(x) \\ d(x) & \rightsquigarrow & [p(x)]^{1-2\lambda} \end{array}$$

For instance, for $\lambda = 1/2$, the three graph Laplacians are essentially the same since the matrix D converges to the identity matrix.

The above analysis shows that instead of focusing on the intractable optimization (1), one should solve the simple quadratic problem (with possibly linear equality constraints if some c_i's are infinite):

$$\min_{F \in \mathbb{R}^n} \sum_{i \in \{\text{train pixels}\}} c_i(Y_i - F_i)^2 + F^t L_{\text{un}} F \tag{4}$$

where $\lambda = 1 - s/2$. Let C be the diagonal $n \times n$ matrix for which the i-th diagonal element is c_i or 0 depending whether i corresponds to a training or test point. Similarly let Y be the n-dimensional vector for which the i-th diagonal element is Y_i or 0 depending whether i corresponds to a training or test point. (4) reduces to

$$\min_{F \in \mathbb{R}^n} \ (F-Y)^t C (F-Y) + F^t L_{\text{un}} F,$$

whose solutions are the solutions of the linear system

$$(L_{\text{un}} + C)F = CY. \tag{5}$$

For infinite regularization coefficient c_i, we have $F_i = Y_i$, while the other F_j's are the solutions of the system:

$$\forall k \ \textstyle\sum_{j \in \{\text{test pixels}\}} L_{k,j} F_j = - \sum_{i \in \{\text{train pixels}\}} L_{k,i} Y_i, \tag{6}$$

where $L_{i,j}$ are the coefficients of the matrix L_{un}.

3 Our segmentation algorithm

3.1 Two-zone segmentation

Our segmentation algorithm is parameterized by

$s \in [1,2]$:	measures how much we believe in "outputs should vary only on input regions having low density" (see (1))
$\sigma_g > 0$:	scale of geometric neighbourhoods (see (7))
$\sigma_c > 0$:	scale of chrometric neighbourhoods (see (7))
$m \in \mathbb{N}$:	size of the local patch (see below)

Let $C(i)$ denote the RGB levels of a square patch of size $2m+1$ around the pixel i. Let x_i denote the geometric position (row+column) of the pixel i. We use the following kernel between pixels

$$k(i,j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma_g^2} - \frac{\|C(i) - C(j)\|^2}{2\sigma_c^2}}. \tag{7}$$

The labels of the training pixels are fixed to 0 or $+1$ depending which zone the pixel i belongs to. Finally, our segmentation method consists in:

(1) computing L_{un} (see (3))
(2) solving the sparse linear system (6)
(3) thresholding the output to $1/2$:
pixel j is assigned to zone $\mathbf{1}_{F_j \geq 1/2}$.

3.2 Multi-zone segmentation

The previous procedure can be simply extended to a multi-zone segmentation with more than 2 zones. Let d denote the number of zones. The output value associated with the k-th zone is defined to be the vector of $\mathbb{R}^d$ having all zero coefficients except its k-th coefficient being equal to one. Then, in order to produce a smooth function that is coherent with the observed outputs on the training points (especially on high-density regions), we solve (6) for each coordinate f_k, $k = 1, \ldots, d$. This procedure can be viewed as a simple one-vs-all segmentation method; at the end, a pixel is assigned to the zone l where $l = \text{argmax}_{k=1,\ldots,d} f_k$.

3.3 Segmentation with prior knowledge of the zones

One might want to use the algorithm without even giving the seeds. Consider a two-zone segmentation problem: object vs background. If absolutely no information is given on the zones, then one can use the prior proposed in [6, Section 2.2].

In some situations, the user has prior knowledge of the following form: each pixel i has a score s_i measuring the likelihood that the pixel belongs to the object zone. This is in particular the case when the user wants to segment an object with a known texture previously-learned from a database of objects of the same category. This information can be directly plugged into the segmentation method by adding the term: $c \sum_{i \in \{\text{test pixels}\}} (s_i - F_i)^2$. The numerical complexity of the method remains unchanged (see below).

3.4 Computational complexity

The computational complexity of the algorithm comes from the resolution of the linear system (7). Using a truncated version of the gaussian kernel, the resulting matrix becomes sparse. Solving such a system can be computed in $O(n.p)$ where n corresponds to the matrix dimension (the number of unlabelled points) and p the number of non-zero entries in the matrix (the number of neighbors for all the points). Efficient computing methods based on multicore processors can be used to speed up the process. In comparison, graphcut algorithms which are known to be fast use the Ford-Fulkerson min-cut max-flow algorithm. Its complexity is $O(E.f)$, where E is the number of edge and f the max flow. In our case, E is equivalent to s and f is proportional to n, showing that the two algorithms have the same complexity.

4 Link with previous segmentation approaches

4.1 Min-cuts methods

If the labels Y_i's take their value in $\{-1;+1\}$, then a variant of our segmentation algorithm would be to replace (4) with

$$\min_{F\in\{-1;1\}^n} \sum_{i\in\{\text{train pixels}\}} c_i(Y_i - F_i)^2 + F^t L_{\text{un}} F \tag{8}$$

so that steps (2) and (3) are replaced with the combinatorial task

$$\min_{\substack{F\in\{-1;1\}^n \\ F_i=Y_i \text{ on train pixels}}} F^t L_{\text{un}} F. \tag{9}$$

This problem can be efficiently solved by a graph-cut algorithm in which the edge between two pixels i and j is weighted by four times the (i,j)-element of the matrix W.

For $s = 1$ (equivalently $\lambda = 1/2$ in (2)), the matrix which appears is exactly the one of the normalized cut eigenvalue problem ([11]). Besides, the discrete version of our algorithm for $s = 2$ (equivalently $\lambda = 0$ in (2)) is comparable to the regularization used in [5, 3, 4].

4.2 Iterative segmentation methods

Several segmentation procedures (e.g. with level sets [9], with iterated graph-cuts [8]) rely on the evolution of a curve or region in which, at each step, the visual properties (e.g. color and texture) of the regions are updated. This leads to generally slow methods because of the iterative aspects of the problem. We believe that our viewpoint is conceptually and practically more satisfactory since the "chicken-and-egg" aspect is directly encoded in our global optimization problem.

4.3 Guan and Qiu's approach

Our framework enables to tackle the energy underlying Guan and Qiu's approach ([6]). The minimized energy functional is the following:

$$\min_{F\in\mathbb{R}^n} \sum_{i\in\{\text{train pixels}\}} c_i(Y_i - F_i)^2 + F^t L_{\text{rw}}^2 F \tag{10}$$

with $c_i = +\infty$ and the normalizing parameter (used in (2)) $\lambda = 0$. In other words, they solve the discrete version of

$$\min_{f:Y_i=f(X_i) \text{ on train pixels}} \int_M \|(\Delta f)(x)\|^2 p(x) dV(x) \tag{11}$$

Considering the regularizer $\int_M \|(\nabla f)(x)\|^2 p(x) dV(x)$ (and its variants) seems to us more adequate than the above regularizer since the latter gives no penalties for linearly-varying functions.

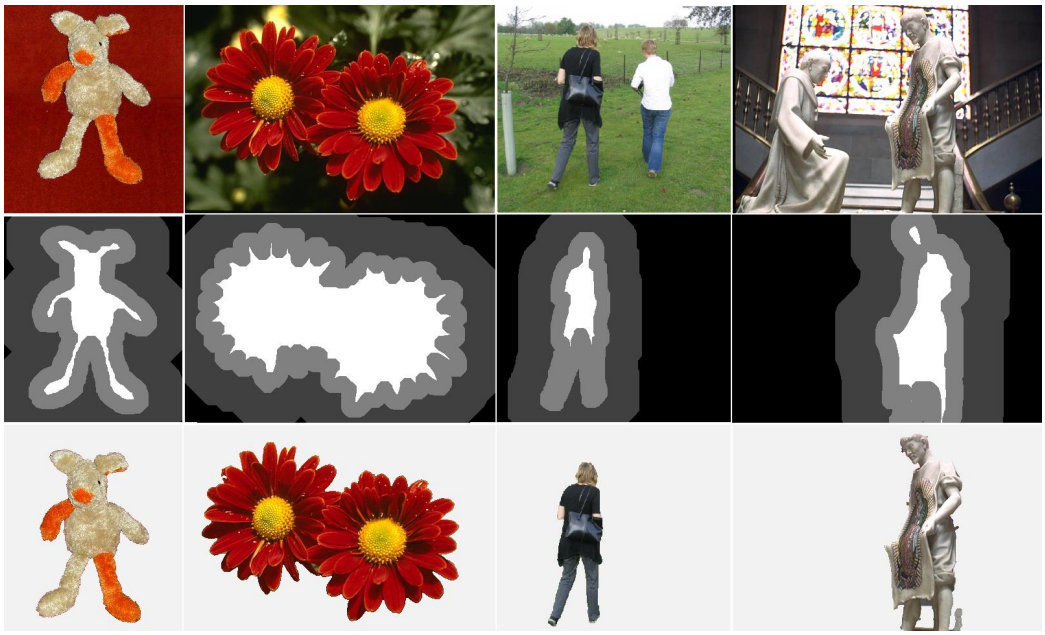

Figure 3: Top row) Initial images. Middle row) Masks provided for the segmentation. Bottom row) Results of our segmentation with $s = 2$. Our method outperformed other segmentation algorithms with a score of 3.3%. The last column corresponds to the worst segmentation with 9.15%.

5 Experimental results

5.1 Comparison with state-of-the-art algorithms

Despite the numerous segmentation databases with ground truths, there is, to our knowledge, only one segmentation database `http://research.microsoft.com/vision/cambridge/i3l/segmentation/GrabCut.htm` for which seed points are given. This database contains seeds of a very particular type since all pixels are labeled except for a narrow-band around the contour of the segmented object.

As such, this database appears to be of limited interest since one can exploit this particular type of geometric information to improve the results of a segmentation. An "almost-naive" segmentation approach that would simply track the

skeleton of the unlabeled points would perform superbly on this data set. The adaptative thresholding (AT) introduced by Guan and Qui in [6, Section 3.1] is somehow designed to produce the same effect. Using the same adaptive filter, we also greatly improved our segmentation results leading to significantly better results than the ones reported in [6]. Yet, we argue that such post-processing step should not be used to evaluate the quality of a segmentation method. Our results are reported in tab.1.

Figure 3 illustrates some typical outputs of our segmentation (with adaptative filtering and $s = 2$). The last column, which corresponds to the worst score (i.e. 9.25%) of our segmentation algorithm is still of great quality. Results on the same dataset with a different parameter $s = 1$ lead to similar results (3.8%).

Segmentation model	Error rate
GMMRF ([3])	7.9%
Our method without AT	5.2%
Square Laplacian regularizer ([6])	4.6%
Our method with AT	3.3%

Table 1: Pourcentage of mislabeled pixels in the region to be classified. Note that the two last scores correspond to algorithms dedicated to segmentation with contour information (using an adaptative filter [6]).

5.2 Discussion

5.2.1 geometric neighborhood

In the experiments, we considered a relatively small geometric neighborhood ($\sigma_c = 10$). Our experiments showed that the graph laplacian is not efficient with a too large σ_c. Isolated pixels appear, large zones influence become too important. So the algorithm cannot use long range pixel similarity. To tackle this problem, we think that this algorithm needs to be plugged with long range or global methods such as Gaussian Mixture Color Model or SVM classifier. This could be computed using the method explained in section 3.3. The graph laplacian would help to diffuse the prior knowledge of a smart classifier.

5.2.2 comparison with graphcut

Graph laplacian based algorithms and graphcut algorithms have the same complexity. They can minimize the same Energy function, by modifying the kernel. However graph cut provide labels and our algorithm provide a real-valued score

function. It can be used to perform alpha matting. Or it could be used as a confidence score. Indeed zones different from both learning zones received an almost null score.

5.2.3 computation time

The segmentation process last between 2 seconds and a minute on the database images, on a Pentium 1.7 MHz. Real time is not yet reachable. But this time could be improved by standard multicore methods for sparse system.

6 Conclusion

This work presents a simple, yet accurate segmentation procedure based on the transductive viewpoint. We clearly illustrated the link between the continuous formulation and its discrete counterpart, introducing a parameter $\lambda = 1 - s/2$ as a measure of the output variations on low density input regions. Our discrete formulation leads to an energy minimization which reduces to a linear system of size the number of pixels to be labeled. Comparison with methods was provided, and segmentation results on natural images clearly demonstrated the quality of our approach. Applied to the data set provided in [3], our method outperformed other approaches. Future work will focus on improving the design of kernel to make it better adapted to the segmentation task at hand, and on reducing the complexity of the algorithm and making it real-time in high-resolution images.

References

[1] Belkin and Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comp.*, 15(6):1373–1396, 2003.

[2] Belkin and Niyogi. Semi-supervised learning on manifolds. *Machine Learning*, 56:209–239, 2004.

[3] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *ECCV06*, pages I: 428–441, 2006.

[4] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision (IJCV)*, 70(2):109–131, 2006.

[5] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *International Conference on Computer Vision (ICCV)*, volume I, pages 105–112, 2001.

[6] J. Guan and G. Qiu. Interactive image segmentation using optimization with statistical priors. In *International Workshop on The Representation and Use of Prior Knowledge in Vision, In conjunction with ECCV 2006, Graz, Austria*, 2006.

[7] M. Hein, J.-Y. Audibert, and U. von Luxburg. Graph laplacians and their convergence on random neighborhood graphs, 2006. Preprint, `http://arxiv.org/abs/math.ST/0608522`.

[8] C. Rother, V. Kolmogorov, and Blake A. Grabcut - interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2004*, 2004.

[9] Mikaël Rousson, Thomas Brox, and Rachid Deriche. Active unsupervised texture segmentation on a diffusion based space. In *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 699–704, Madison, Wisconsin, USA, June 2003.

[10] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Computer Vision and Pattern Recognition*, pages 18–25, 2000.

[11] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[12] D. Spielman and S. Teng. Spectral partioning works: planar graphs and finite element meshes. In *37th Ann. Symp. on Found. of Comp. Science (FOCS)*, pages 96–105. IEEE Comp. Soc. Press., 1996.