Learning and Inferring Image Segmentations using the GBP Typical Cut Algorithm

Noam Shental Assaf Zomet Tomer Hertz Yair Weiss
School of Computer Science and Engineering,
The Hebrew University of Jerusalem,
91904, Jerusalem, Israel

Abstract

Significant progress in image segmentation has been made by viewing the problem in the framework of graph partitioning. In particular, spectral clustering methods such as "normalized cuts" (ncuts) can efficiently calculate good segmentations using eigenvector calculations. However, spectral methods when applied to images with local connectivity often oversegment homogenous regions. More importantly, they lack a straightforward probabilistic interpretation which makes it difficult to automatically set parameters using training data.

In this paper we revisit the typical cut criterion proposed in [1, 5]. We show that computing the typical cut is equivalent to performing inference in an undirected graphical model. This equivalence allows us to use the powerful machinery of graphical models for learning and inferring image segmentations. For inferring segmentations we show that the generalized belief propagation (GBP) algorithm can give excellent results with a runtime that is usually faster than the ncut eigensolver. For learning segmentations we derive a maximum likelihood learning algorithm to learn affinity matrices from labelled datasets. We illustrate both learning and inference on challenging real and synthetic images.

1. Introduction

Many authors have pointed out that the problem of image segmentation can be formulated as a graph partitioning problem (e.g. [2, 8, 11, 9]). In the typical conversion every pixel corresponds to a node in the graph and pixels are connected to nearby pixels with a weighted edge, where the weight often depends on the similarity of a local image feature at the two pixels. Segmenting the image is equivalent to finding a partition of the graph vertices.

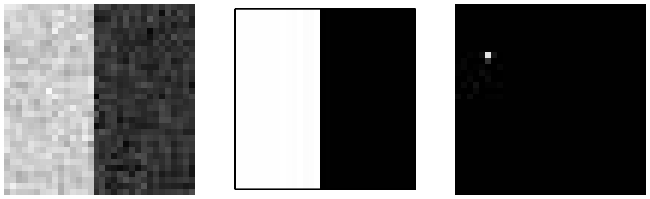


Figure 1. The problem with minimal cut segmentations. The trivial segmentation shown on the right has lower cut value than the desired segmentation shown in the middle. Normalized cut and typical cut are two criteria that avoid these trivial segmentations.

How would we define a good segmentation? Wu and Leahy [11] suggested the minimal cut criterion. Define:

$$cut(A,B) = \sum_{i\in A, j\in B} W(i,j) \tag{1}$$

where $W(i,j)$ is the affinity between node i and j in the graph. The minimal cut criterion finds segmentations that minimize $cut(A,B)$.

The advantage of using the minimal cut criterion is that the optimal segmentation can be computed in polynomial time. A disadvantage, pointed out by Shi and Malik [9], is that it will often produce trivial segmentations. For example, since the cut grows linearly with the number of edges cut, a single pixel cut from its four nearest neighbor will often have a lower cut value than a large foreground separated from background. Figure 1 shows an example.

In order to avoid these trivial segmentations, Shi and Malik suggested the normalized cut (ncut) criterion:

$$Ncut(A,B) = \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)} \tag{2}$$

where $asso(A,V) = \sum_j \sum_{i\in A} W(i,j)$.

Since $asso(A,V)$ is related to the size of group A, the Normalized cut criterion directly penalizes partitions where one of the groups is small. Thus unlike the minimal cut

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

criterion, a segmentation of a single, noisy pixel from the entire image will, in general, not be optimal.

Minimization of the ncut criterion is NP Complete but Shi and Malik showed that an approximate solution can be found by computing the eigenvectors of the matrix $N = D^{-1/2}WD^{-1/2}$, where D is a diagonal degree matrix $D(i,i) = \sum_j W(i,j)$.

1.1 Typical cuts

The typical cut criterion avoids trivial segmentations using a rather different method. The criterion was first introduced by Blatt et al. [1] in the framework of statistical physics, and was reformulated by Gdalyahu et al. [5] in terms of graph partitioning and image segmentation. Unlike most graph partitioning algorithms, this one is directly based on a *probability distribution over partitions*. For example, Blatt et al. define a probability distribution over possible partitions by:

$$\Pr(A,B) = \frac{1}{Z}e^{-cut(A,B)/T} \tag{3}$$

where T is a "temperature" parameter that serves as a free parameter.

Using this probability distribution, the most probable partition is simply the minimal cut. Thus performing MAP inference under this probability distribution will still lead to trivial segmentations. However, as pointed out by [1, 5], there is far more information in the full probability distribution over partitions than solely in the MAP partition. For example, consider the pairwise correlation $p(i,j)$ defined for any two neighboring nodes in the graph as the probability that they belong to the same segment:

$$p(i,j) = \sum_{A,B} \Pr(A,B) SAME(i,j;A,B) \tag{4}$$

with $SAME(i,j;A,B)$ defined as 1 iff $i \in A$ and $j \in A$ or $i \in B$ and $j \in B$.

Referring again to the single pixel in figure 1, while that pixel and its neighbors do not appear in the same segment in the most probable partition, they *do* appear in the same segment for the vast majority of partitions. Thus we would expect $p(i,j) > 1/2$ for that pixel and its neighbors.

Gdalyahu et al. defined the *typical cut* partition as the one defined by the connected components of the graph where all edges for which $p(i,j) < 1/2$ have been removed. Gdalyahu et al. showed encouraging results on image segmentation problems using this criteria.

In summary, both the normalized cut and the typical cut criteria are promising, principled approaches to segmentation. However, while normalized cut has been used widely for many image segmentation problems, the typical cut has not. Mostly this is due to algorithmic considerations. Just as exact minimization of the normalized cut criterion is NP complete, exact calculation of $p(i,j)$ (equation 4) is exponential in the size of the image. While efficient approximate algorithms for normalized cut are available through eigensolvers, to this day, there has been no similar algorithm for typical cuts. Both [1, 5] used sophisticated stochastic algorithms whose behavior for finite samples is difficult to analyze. In fact, the problem of determining the equilibrium distribution over segmentations from which the algorithm used in [5] samples from, is an open one.

In this paper, we show an equivalence between calculating typical cuts and inference in an undirected graphical model. This equivalence allows us to use the powerful machinery of graphical models for inference and learning. For inference, we show that generalized belief propagation (GBP) leads to a simple, deterministic segmentation algorithm whose run time is usually faster than the eigensolver used in ncuts. For learning, we derive a maximum likelihood algorithm to learn affinity matrices from labeled datasets. We illustrate both inference and learning on challenging real and synthetic images.

2 Typical cuts and graphical models

This paper is based on the observation that equation 3 defines an undirected graphical model and hence algorithms for approximate or exact inference in graphical models can be used to calculate $p(i,j)$ and can also be used to learn affinities.

An undirected graphical model with pairwise potentials (see [13] for a review) consists of a graph G and potential functions $\Psi_{ij}(x_i, x_j)$ such that the probability of an assignment x is given by:

$$\Pr(x) = \frac{1}{Z}\prod_{<ij>} \Psi_{ij}(x_i, x_j) \tag{5}$$

where the product is taken over nodes that are connected in the graph G.

To relate this to typical cuts we first define for every partition (A,B) a binary vector x such that $x(i) = 0$ if $i \in A$ and $x(i) = 1$ if $i \in B$. We then define:

$$\Psi_{ij}(x_i, x_j) = \begin{pmatrix} 1 & e^{-W(i,j)/T} \\ e^{-W(i,j)/T} & 1 \end{pmatrix} \tag{6}$$

Observation 1: The probability distribution (equation 3) is equivalent to that induced by a pairwise undirected graphical model (equation 5) whose graph G is the same as the graph used for graph partitioning and whose potentials are given by equation 6.

This equivalence described above, holds for any number of segments q. Let $(A_1, A_2, \cdots, A_q)$ be a partitioning of the

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

graph into q segments. Define $cut(A_1, A_2, \cdots A_q)$ in direct analogy to equation 1, and:

$$\Pr((A_1, A_2, \cdots, A_q) = \frac{1}{Z} e^{-\frac{1}{T} cut(A_1, A_2, \cdots, A_q)} \quad (7)$$

In the equivalent graphical model formalism, x is no longer a binary vector but rather takes on q possible discrete values and the potentials are analogous to equation 6 with 1 along the diagonal and $e^{-W(i,j)/T}$ in all of the off diagonal elements. The graphical model defined this way defines a probability distribution over q-way cuts that is equivalent to equation 7.

2.1 Inferring image segmentations using GBP

The observation means that algorithms for approximate inference in graphical models can be used to devise a typical cut algorithm. We used the generalized belief propagation (GBP) ([13]) algorithm which has been shown to give excellent results on similar problems. We refer the reader to [13] for a full description of the GBP algorithm.

Specifically, the GBP typical cut algorithm is given an affinity matrix $W(i, j)$ between all neighboring pixels in the image, and does the following:

1. Construct a graphical model with potentials given by $e^{-W(i,j)/T}$ in the off diagonal terms and 1 for the diagonal terms. (see eq 6)

2. Use generalized belief propagation to compute marginal probabilities over pairs of adjacent pixels $b_{ij}(x_i, x_j)$. Derive $p(i, j)$ from these marginal probabilities using:

$$p(i, j) = \sum_{x_i = x_j} b_{ij}(x_i, x_j) \quad (8)$$

3. Remove from the graph any edges for which $p(i, j) < 1/2$ and find the connected components of the graph.

We now give the specific GBP updates for the problem of image segmentation. Our input is a 2D grid in which each pixel is connected to its four nearest neighbors. The algorithm described here is called the "two-way" GBP algorithm in [12].

To apply GBP to our problem, one first forms a *region graph*. In our case, this graph contains all quartets of neighboring nodes and all pairs of neighboring nodes. In general, each quartet is connected to 4 pairs and each pair is connected to two quartets. Figure 2 shows a small 4×4 grid and a portion of the associated region graph. In the general GBP algorithm for 2D grids, there will also be regions for single pixels, but due to the symmetry of the potentials in our problem, they can be ignored.

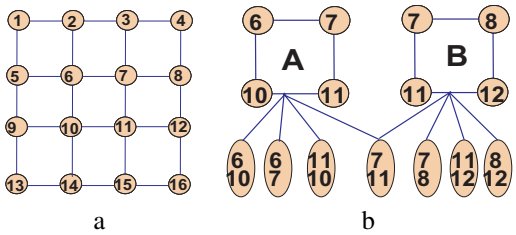


Figure 2. **a.** A small 4×4 image. **b.** A portion of the associated region graph. In general all quartets of neighboring pixels are connected to four pairs of pixels and all pairs are connected to two quartets. GBP passes messages along both directions of the graph.

Along each edge in the region graph, messages are passed in both directions. These messages are probability distributions over pairs of pixels. At every iteration, the message sent by a node in the region graph is updated based on the messages it received at the previous iteration. When all messages have converged, every pair of nodes forms a "belief" $b_{ij}(x_i, x_j)$: an approximation to the marginal probability of pixels x_i and x_j.

The message updates involve multiplying and summing other messages. We give the update rules for a specific pair and quartet. All other message updates are identical up to a permutation. Referring to figure 2 we denote by $m_{A \to 7,11}(x_7, x_{11})$ the message that the quartet $A = (6, 7, 10, 11)$ sends to the pair $(7, 11)$ and by $m_{7,11 \to B}(x_7, x_{11})$ the message that the pair (x_7, x_{11}) sends to the quartet $B = (7, 8, 11, 12)$.

The update rules are:

$$\begin{aligned} m_{A \to 7,11}(x_7, x_{11}) &\propto \textstyle\sum_{x_6, x_{10}} (m_{6,10 \to A}(x_6, x_{10}) \\ &\quad \cdot m_{6,7 \to A}(x_6, x_7) m_{10,11 \to A}(x_{10}, x_{11})) \\ m_{7,11 \to B}(x_7, x_{11}) &\propto m_{A \to 7,11}(x_7, x_{11}) \Psi_{7,11}(x_7, x_{11}) \\ b_{7,11}(x_7, x_{11}) &\propto m_{A \to 7,11}(x_7, x_{11}) m_{B \to 7,11}(x_7, x_{11}) \\ &\quad \cdot \Psi_{7,11}(x_7, x_{11}) \end{aligned}$$

In a simple implementation of the above algorithm, each message is a $q \times q$ matrix explicitly representing the joint probability of a pair of pixels. For example $m_{A \to 7,11}$ explicitly represents the joint labelings of pixels 7 and 11. However, due to the symmetry of the potentials, each such matrix actually only contains two distinct values: one along the diagonal (corresponding to the pairs where $x_7 = x_{11}$) and another off the diagonal (corresponding to pairs where $x_7 \neq x_{11}$). Thus each vector of probabilities can be represented by a single scalar number (e.g. the ratio of the probabilities of $x_7 = x_{11}$ and $x_7 \neq x_{11}$). By taking advantage of this scalar representation, we can rewrite the algorithm in the following way.

Assume the message $m_{k \to l}(x_i, x_j)$ is represented by 1 along the diagonal, and a certain value g_{ij} in all off diagonal

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

elements. Hence the update rule for $m_{A\to 7,11}(x_7, x_{11})$ is given by $g_{7,11} = NE_{7,11}/E_{7,11}$ [1]

where

$$\begin{aligned} E_{7,11} &= 1 + (q-1)\cdot[g_{6,7}\cdot g_{10,11} + g_{6,10}\cdot g_{10,11} \\ &\quad + g_{6,7}\cdot g_{6,10} + (q-2)\cdot g_{6,7}\cdot g_{6,10}\cdot g_{10,11}] \\ NE_{7,11} &= g_{6,7} + g_{10,11} + (q-2)\cdot g_{6,7}\cdot g_{10,11} \\ &\quad + g_{6,10}\cdot[1 + (q-2)\cdot g_{10,11} + (q-2)\cdot g_{6,7} \\ &\quad + ((q-2)^2 + q - 1)\cdot g_{6,7}\cdot g_{10,11}] \end{aligned}$$

Since the potentials $\Psi_{ij}(x_i, x_j)$ are also governed by a single parameter, g_{ij}^{Ψ}, the 'g' for $m_{7,11\to B}(x_7, x_{11})$ is simply $g_{7,11}\cdot g_{7,11}^{\Psi}$. The update rule for $b_{7,11}(x_7, x_{11})$ is also a multiplication of the three corresponding g's.

For a given $W(i,j)$ the algorithm has a single free parameter T. This parameter implicitly defines the number of segments in the final segmentation. Note that the final outputted partition may contain any number of segments, regardless of q. For $T \to 0$ all the probability mass in equation 7 is centered on the solution where all entries of x are identical and thus the algorithm returns one big segment. For $T \to \infty$ the probability mass in equation 7 is uniformly distributed and thus $p(i,j)$ approaches $1/q$ and the algorithm returns many small segments.

Note that every iteration of the GBP algorithm is linear in the number of edges in the graph, or equivalently, for nearest neighbor connections, linear in the number of pixels. We are not aware of a bound on the number of iterations needed until convergence but we found that for half sized images ($240x360$) 10 iterations were sufficient. Also note that while our algorithm deals with probabilities it is a *deterministic* algorithm: unlike former typical cut algorithms which are stochastic algorithms, two runs of the GBP typical cut algorithm with the same input will give exactly the same answer.

2.2 Learning image segmentations using maximum likelihood

Many authors have pointed out that a major problem in segmentation using graph partitioning is how to define the affinities between pixels (e.g. [4]). There are many different gestalt cues for segmentation including color, texture, contour etc., and different weights for these cues will lead to very different segmentations. We would like to use a labeled dataset to learn the "right" affinities.

More specifically let us assume the "correct" affinity is a linear combination of a set of known affinity functions $\{f_k\}_{k=1}^K$. Hence the affinity between neighboring pixels i and j, is defined by: $W(i,j) = \sum_{k=1}^K \alpha_k f_k(i,j)$. The "basis" affinity functions f_k can either correspond to different cues (in which case the final affinity is a weighted linear combination of cue affinities) or to nonlinear functions of affinities (in which case the final affinity is a *nonlinear* combination of cue affinities).

In addition assume we are given a labeled training sample (similar to the Berkeley segmentation database used in [4]) in which images are segmented by hand. For each image in the training set, we can compute the basis affinity values $f_k(i,j)$ between neighboring pixels. Our goal is to estimate the affinity mixing coefficients α_k.

As we now show this problem can be solved using the graphical model defined by the typical cut probability distribution (Equation 7). The probability of the partition x is defined

$$\begin{aligned} P(x) &= \frac{1}{Z}e^{-cut(x)} = \frac{1}{Z}e^{-\sum_{<ij>}(1-\delta(x_i-x_j))W(i,j)} \\ &= \frac{1}{Z(\alpha)}e^{-\sum_{k=1}^K \alpha_k \mathsf{fcut}_k(x)} \end{aligned}$$

Where we have defined: $\mathsf{fcut}_k(x) = \sum_{<ij>}(1-\delta(x_i - x_j))f_k(i,j)$. $\mathsf{fcut}_k(x)$ is the cut value defined by x when only taking into account the affinity function f_k, hence it can be computed using the training sample.

Differentiating the log likelihood with respect to α_k gives the familiar exponential family equation:

$$\frac{\partial \ln P(x)}{\partial \alpha_k} = -\mathsf{fcut}_k(x) + <\mathsf{fcut}_k>_\alpha \qquad (9)$$

Equation 9 gives an intuitive definition for the optimal α: the optimal α is the one for which $<\mathsf{fcut}_k>_\alpha = \mathsf{fcut}_k(x)$. That is, the optimal α is the one for which the expected values of the cuts for each feature separately, match exactly the values of these cuts in the training set.

Since we are dealing with the exponential family, the likelihood is convex and the ML solution can be found using gradient ascent. To calculate the gradient explicitly, we use the linearity of expectation:

$$\begin{aligned} <\mathsf{fcut}_k>_\alpha &= \sum_{<ij>} <(1-\delta(y_i - y_j)>_\alpha f_k(i,j) \\ &= \sum_{<ij>}(1 - p(i,j)_\alpha) f_k(i,j) \end{aligned}$$

Where $p(i,j)_\alpha$ are the pairwise correlations for given values of α.

3 Experimental results

3.1 Inference results

By using GBP to compute the pairwise correlations $p(i,j)$ we obtain an approximation of the true pairwise cor-

[1] $E_{7,11}$ denotes the sum of over all assignments in which $x_7 = x_{11}$, and $NE_{7,11}$ denotes the sum of all assignments in which $x_7 \neq x_{11}$. It should also be noted that we assume that $q \geq 4$

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

relations. In order to evaluate the quality of these approximations we compare the correlations $p(i,j)$ calculated using an extensive MCMC sampling procedure [10] to those calculated using GBP with the clusters being four neighboring pixels in the graph. Figure 3 shows results of GBP approximations for a $30x30$ 2D uniform grid. The clique size in a junction tree is of order 2^{30} and hence exact inference is impossible. GBP converged in only 10 iterations and gives an excellent approximation.

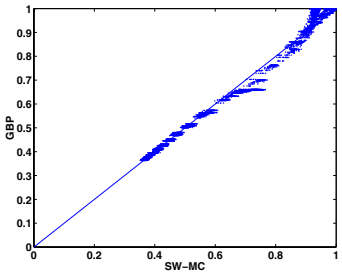
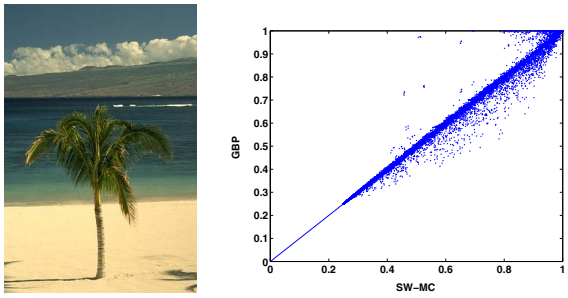


Figure 3. Scatter plot of pairwise correlations in a $30x30$ grid, using MCMC [10] and GBP. Each dot corresponds to the pairwise correlation of one edge at a specific temperature. Notice the excellent correspondence between GBP and MCMC

Figure 4(b) presents a comparison of the MCMC correlations with those calculated by GBP on a real 120 by 80 image (see Figure 4(a)) with affinity based on color similarity.



(a) (b)

Figure 4. (a) Original image. (b) Scatter plot of pairwise correlations in the image using MCMC [10] and GBP, at one specific temperature. Each dot corresponds to one pair of pixels.

We now compare ncuts and GBP typical cuts on real and synthetic images. Both algorithms received as input exactly the same affinity matrix $W(i,j)$. Every pixel was connected to its 4 nearest neighbors, and the weight was only a function of the absolute difference between the RGB values in the two pixels:

$$W(i,j) = \frac{1}{\epsilon + \|(r_i,g_i,b_i) - (r_i,g_i,b_i)\|_1}$$

We should emphasize that these affinities are certainly not the optimal affinities to use for image segmentation: they only take into consideration color and not texture or contour and the radius of connections is very small. Our goal here was not to build a state-of-the-art segmenter but rather to understand the differences between the various algorithms, and to demonstrate their capabilities. Undoubtedly, both ncuts and GBP typical cut would improve with better $W(i,j)$ and with higher neighbor connectivity.

For GBP typical cut we chose $q = 6$ for all images and the temperature was adjusted manually to achieve the desired number of clusters. For ncuts we used the k-way algorithm described in [9]. Specifically, we calculate the k largest eigenvectors of the matrix $D^{-1/2}WD^{-1/2}$ and embed every pixel in a k dimensional vector space. We then ran the Kmeans algorithms in the vector space to give the Normalized cuts segmentation. We ran Kmeans 10 times and chose the clustering that gave lowest distortion. The free parameter k was adjusted manually to achieve the desired number of clusters.

When running Both ncut and GBP typical cut on the noisy edge image described in the introduction (Figure 1) both algorithms avoid the trivial segmentation of the minimal cut algorithm and output the correct segmentation.

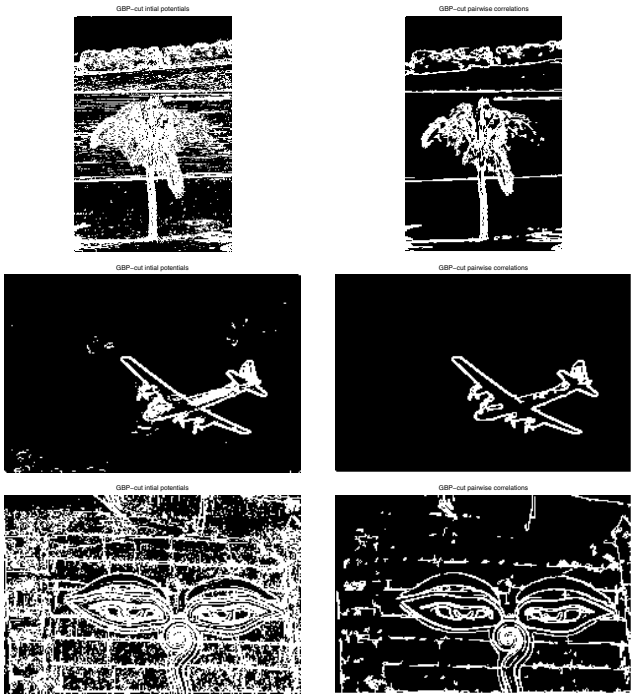


Figure 6. A comparison of the edge probability maps before and after running the GBP typical cut algorithm. Edges in the graph appear in locations where $p(i,j) < 0.5$. Note that edges that do not denote a large region but have a strong gradient (e.g. the ripples on the waves) have $p(i,j) > 0.5$ and therefore disappear from the resulting edge map.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

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

Figure 5. A comparison of segmentation results using GBP typical cut (columns (a) and (b)), and Normalized cut (columns (c) and (d). Both algorithms used the same affinity matrices. (a) and (c) show the 15 largest cluster boundaries superimposed on the original image.(c) and (d) show a gray level map of the 15 largest clusters. As may be seen, the Normalized cut algorithm tends to split large homogeneous regions.

Figure 5 shows the results [2] on three real images taken from [6]. We emphasize that we chose images that contain large homogeneous regions to highlight the distinction between the two algorithms. It is known that ncuts with local connectivity will tend to favor splitting homogeneous regions (e.g. [3]). In the appendix, we analyze ncuts and typical cuts for a simple synthetic image with homogenous regions and show that typical cuts will not oversegment the large homogenous region. What happens in real images with homogenous regions ?

As shown in figure 5, ncuts indeed tends to favor splitting of large homogeneous regions while the typical cut does not. At the same time, the GBP typical cut algorithm also outputs some small noisy clusters which the ncut algorithm avoids. Which of these two errors is more important is, of course, application dependent. It can be seen that, the segmentations obtained by using the GBP typical cut are more meaningful: one can recognize the airplane, palm tree and eyes in the second column of figure 5 but not in the fourth column. It should be noted that for these images, the GBP implementation was an order of magnitude faster than Matlab's eigs eigensolver.

To get a better understanding of how GBP typical cut works, we display in figure 6 the optimal edge guesses before and after running GBP. The "before" column shows a white pixel wherever the probability of an edge based only on the local color gradient is greater than 0.5. The "after" column shows a white pixel whenever $p(i,j)$ calculated using GBP is greater than 0.5. Note that edges where there is a significant color gradient but do not denote a boundary between two large regions are suppressed (e.g. the ripples on the waves).

[2] We thank Marshall Tappen for providing a C++ GBP implementation.

3.2 Learning results

We experimented with the ML learning algorithm on the problem of ignoring shadows. When one uses color similarity between pixels, the largest differences between pixels may be due to shadows. Whether we want our algorithm to indeed segment based on shadows or ignore the shadows is, of course, application dependent. This preference can be communicated using a training set: when the training set ignores shadows we would like our algorithm to learn to ignore them, but if the training set segments based on shadows, we would like the algorithm to do the same.

Fig 7 shows a synthetic example. There is one training image (fig 7a) but two different segmentations (fig 7b,c). The first training segmentation is based on the shadows and the second training segmentation ignores shadows.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

The three affinity functions used here are based on intensity differences in the three R, G, B channels. The affinity is an affine function of the intensity difference. We used gradient ascent as given by equation 9. Figure 7c shows a novel image and figures 7d,e show two different pairwise correlations of this image using the learned α. Indeed, the algorithm learns to either ignore or not ignore shadows, based on the training set.

Figure 8 shows results on real images. For real images, we found that a preprocessing of the image colors is required in order to learn shadow-invariant linear transformation. This was done by saturating the image colors. The training segmentation (shown in figure 8a-c) ignores shadows. On the novel image (shown in figure 8d) the most salient edge is a shadow on the face. Nevertheless, the segmentation based on the learned affinity (figure 8e) ignores the shadows and segments the facial features from each other. In contrast, a typical cut segmentation which uses a naive affinity function (combining the three color channels with uniform weights) segments mostly based on shadows (figure 8f).

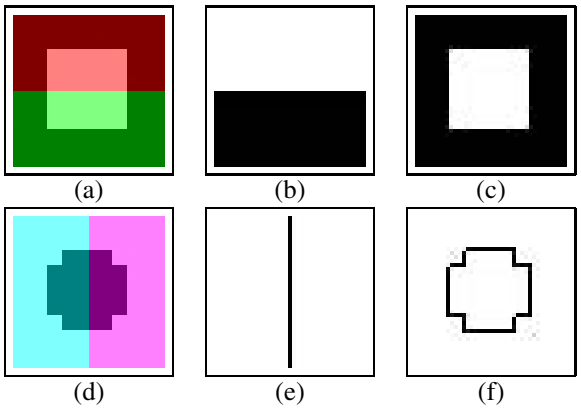


Figure 7. A synthetic example for learning the affinity function. The top row presents the training set: The input image (a), the clusters of the first experiment (b), and the clusters of the second experiment (c). The bottom row presents the result of the learning algorithm: The input image (d), the marginal probabilities $p(i, j)$ (Eqn. 4) in the first experiment (e) and the marginal probabilities $p(i, j)$ in the second experiments(f). The illustration is better viewed in color.

4 Discussion

The introduction of graph partitioning techniques into image segmentation has been tremendously helpful. In particular, the ncuts algorithm has given excellent performance over a wide range of images. This is due both to the criterion which avoids trivial segmentations and to the simple eigenvector algorithm. In this paper we have presented a new algorithm: GBP typical cut which in terms of complexity is usually faster than the eigenvector ncut. We showed

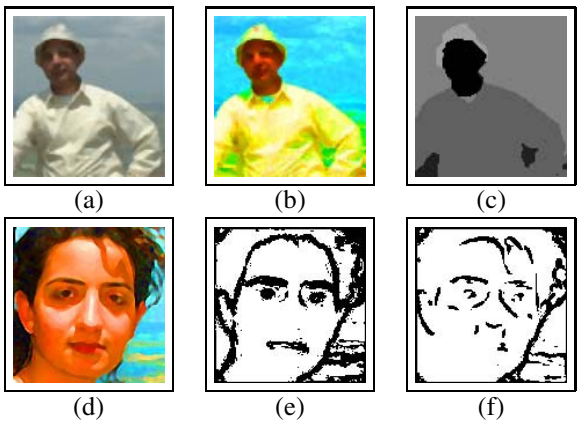


Figure 8. Learning a color affinity function which is invariant to shadows. The top row shows the learning data set: The input image(a), the pre-processed image (b) and the clustering to regions (invariant to shadows) (c). The bottom row presents, from left to right, the pre-processed input image for the classification stage (d), an edge map produced by learning the shadow-invariant affinity (e) and an edge map produced by a naive affinity function, combining the 3 color channels with uniform weights (f). The edge map was computed by thresholding the pairwise correlations p(i,j) (Eqn. 4). The illustration is better viewed in color. See text for details.

that the algorithm avoids the trivial segmentations of min cuts but also avoids the oversegmentation of homogeneous regions that may plague ncuts. Another important advantage of the GBP typical cut, is that it comes with a simple, probabilistic framework, which naturally allows using various simple algorithms for estimating the model parameters as we have demonstrated in this paper.

Although promising results have been presented for the problem of learning affinities from ground truth segmentations in the ncut framework [7, 4] it is far from obvious how to change $W(i, j)$ so that the ncut segmentations will give boundaries that agree with the human boundaries. This is true even when one adopts the random walk view of ncuts proposed in [7]. In contrast ML estimation for undirected graphical models is very well understood and gives a straightforward method to learn affinity matrices from labeled data.

Our current work of maximum likelihood estimation for the affinity matrix $W(i, j)$ is just an example of the type of research directions that open up once we have a graphical model and a criterion that is optimal for segmentation given that model. We view the GBP typical cut algorithm as a bridge between the image segmentation problem and the powerful machinery of graphical models.

References

[1] M. Blatt, S. Wiseman, and E. Domany. Data clustering using a model granular magnet. *Neural Computation*, 9:1805–1842, 1997.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

[2] I. Cox, S. Rao, and Y. Zhong. Ratio regions: A technique for image segmentation. In *Proc. 13th Int'l Conf. Pattern Recognition*, 1996.

[3] C. Fowlkes, S. Belongie, and J. Malik. Efficient spatiotemporal grouping using the nystrom method. In *Proc. CVPR 2001*, 2001.

[4] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: Combinging patch-based and gradient-based approaches. In *Proceedings CVPR 2003*, 2003.

[5] Y. Gdalyahu, D. Weinshall, and M. Werman. Self organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(10):1053–1074, 2001.

[6] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[7] M. Meila and J. Shi. Learning segmentation by random walks. In *Advances in Neural Information Processing Systems*, 2001.

[8] S. Sarkar and K. Boyer. quantitative measures of change based on feature organization: eigenvalues and eigenvectors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1996.

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

[10] J. Wang and R. Swendsen. Cluster monte carlo algorithms. *Physica A*, 167:565–579, 1990.

[11] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[12] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *Technical Report Number TR2002-35*, 2002.

[13] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*. Morgan Kaufmann, 2003.

A Appendix: Analytical comparison on imbalanced segmentations

In this section we analytically compare the performance of optimal ncuts to optimal typical cuts on a class of imbalanced images with nearest neighbor connectivity. It is known that ncuts will have trouble under these conditions (e.g. [3]) but will these problems also plague typical cuts?

Suppose our input image is as shown in figure 9. There is a single, square figure C of size $m \times m$ embedded in a uniform background D of size $n \times n$. Assume $W(i,j) = 1$ for any two neighboring pixels with the same intensity and $W(i,j) = 1/4$ for two neighboring pixels across an intensity boundary. Consider segmenting the background into two regions A, B then we have (ignoring boundary effects):

$$\begin{aligned} Ncut(A,B) &= \frac{n \cdot 1}{4n^2/2} + \frac{n \cdot 1}{4n^2/2} \\ Ncut(C,D) &= \frac{4m \cdot \frac{1}{4}}{4m^2} + \frac{4m \cdot \frac{1}{4}}{4(n^2 - m^2)} \end{aligned}$$

Observation 1: For $n >> m$ optimal Normalized cut with these weights will prefer cutting the background into equal pieces over segmenting the figure from the background.

The proof is straightforward: for a fixed m as n grows $Ncut(A,B)$ will approach zero while $Ncut(C,D)$ will approach $1/4m$.

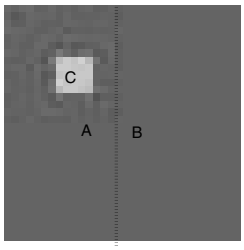


Figure 9. A simple image on which we can calculate the performance of optimal Normalized cuts and typical cuts when both algorithms have nearest neighbor connectivity. As the size of the image grows, optimal Normalized cuts will prefer splitting the background in half instead of segmenting the figure from ground. Typical cuts will always prefer segmenting the figure from ground.

How dependent is this result on the nearest neighbor connections? Note that as long as the number of connections of each node is bounded above (or even grows sublinearly with n) the same argument holds. Note also that this result is not restricted to normalized cuts. It also holds for ratio cuts [2] (where one minimizes $cut(A,B)/weight(A)$) as long as weight A increases with the number of elements in A.

Observation 2: As n grows optimal typical cut will *never* prefer to split the background into two rather than segmenting the figure from the background.

To prove this consider a point $i_A \in A$ and a neighboring point $j_B \in B$. Denote their correlation as a function of n by $p(i_A, j_B; n)$. Similarly, consider a point $i_C \in C$ $j_D \in D$ and denote their correlation by $p(i_C, j_D; n)$. What happens to these correlations as n grows? Note that as we increase n the local subgraph around these 4 points does not change: neither the topology nor the potentials. We are simply adding nodes to the graph that are increasingly far away from these 4 points. Since the additional node are increasingly far away from the 4 points, their influence decreases as n increases and $p(i_A, j_B; n)$,$p(i_C, j_D; n)$ will tend to an asymptotic value that is independent of n.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE