Spectral Embedding and Min-Cut for Image Segmentation

Francisco Estrada, Allan Jepson, and Chakra Chennubhotla
Department of Computer Science,
University of Toronto
[strider,jepson,chakra]@cs.utoronto.ca

Abstract

Recently it has been shown that min-cut algorithms can provide perceptually salient image segments when they are given appropriate proposals for source and sink regions. Here we explore the use of random walks and associated spectral embedding techniques for the automatic generation of suitable proposal regions. To do this, we first derive a mathematical connection between spectral embedding and anisotropic image smoothing kernels. We then use properties of the spectral embedding and the associated smoothing kernels to select multiple pairs of source and sink regions for min-cut. This typically provides an over-segmentation, and therefore region merging is used to form the final image segmentation. We demonstrate this process on several sample images.

1 Introduction

Interactive min-cut has been shown to be a practical and useful tool for image segmentation. Boykov and Jolly [4] investigated min-cut image partitions generated from graphs formed using each pixel as a node, along with two additional source and sink nodes. In their work, two image regions are given, and the pixels of one region are connected to the sink node, while those from the other region are connected to the source node. Every pixel is also connected to its neighbours with edges weighted by their affinities. The weights of edges joining pixels to source and sink regions are set to be larger than the sum of the weights of all other edges in the graph so that they will never be cut.

A minimal cut separating the source from the sink is computed, where the cost of any cut is the sum of the weights of all the edges that are cut. The most appealing property of this approach is that the cut is guaranteed to be a global minimum. Moreover, with the recent development of min-cut algorithms, this process is computationally feasible even for reasonably large images [3, 6]. The problem, then, is to generate suitable proposals for source and sink regions. Boykov et. al. [3, 4] rely on user interaction, while Veksler [14] places sink regions outside the image, and uses the image pixels, one at a time, as source regions.

Here we propose an automatic procedure for generating a small set of region proposals, as well as a method for assigning these proposals to source and sink regions. The main contributions of our paper are: 1) A mathematical connection between spectral embedding and anisotropic image smoothing kernels. 2) The use of the embedding to select

kernels to generate source and sink regions for min-cut. 3) An implementation of the approach, along with a brief evaluation of the results.

2 Image Segmentation using Min-Cut

Given an $n \times m$ image $I(\vec{x})$, we assume we are given an $nm \times nm$ (sparse) symmetric affinity matrix A whose elements satisfy $A_{i,j} \in [0,1]$, with the specific value of $A_{i,j}$ indicating the similarity of image pixels $\vec{x}_i$ and $\vec{x}_j$. Perfect similarity is represented by $A_{i,j} = 1$, and we set $A_{i,i} = 1$ for all i. We choose to use sparse affinity matrices formed using the standard image 8-neighbourhood, although any neighbourhood structure (including the whole image) could also be used.

Our general goal is to study the process of image segmentation given such an affinity matrix A and, for now, we set aside the important issue of how to define appropriate affinities in the first place. For the purpose of experimentation, we use a simple affinity measure based on the gray-level difference of neighbouring pixels $\vec{x}_i$ and $\vec{x}_j$, namely

$$A_{i,j} = e^{-(I(\vec{x}_i)-I(\vec{x}_j))^2/(2\sigma^2)}, \tag{1}$$

where σ is a parameter representing the typical gray-level variation of similar pixels.

These affinities are used as weights in a graph formed using the image pixels as nodes. There are two general types of min-cut problems that can be considered for such graphs. The first problem is to find a two-way partitioning of the graph nodes which minimizes the sum of the weights of the cut edges. We refer to this as the graph partitioning problem. A related but computationally much simpler problem is the min-cut/max-flow problem, which we refer to here simply as min-cut. To understand the difference between these two problems, consider that the cut resulting from the min-cut/max-flow formulation is not necessarily the smallest cut that can be obtained in a particular image (i.e. not a solution of the graph partitioning problem). Different pairs of sources and sinks will yield minimum cuts with different values. To approximate the solution of the graph partitioning problem Gdalyahu, et. al. [8] use a probabilistic method first described by Karger and Stein in [9] to obtain cut proposals, then average many such proposals to generate a typical cut used to segment the image. Alternatively, Wu and Leahy [15] solve the min-cut/max-flow problem between every pair of pixels in the image, and use the K smallest cuts to partition an image into K regions.

Here the min-cut problems we consider are specified using the affinities in A as inter-pixel edge weights, along with suitable source and sink regions. The criteria for selecting suitable regions are quite intuitive, namely: 1) source and sink regions must each be sufficiently large; 2) the source and sink regions should come from disjoint unions of natural image segments; and 3) the set of all source and sink region pairs used must be rich enough to cause each natural image segment to be separated from the other segments. The first point here arises due to the fact that the minimum-cost cut is bounded above by the cost of cutting the source region from the neighbouring image pixels, and similarly for the sink region. Therefore large image segments will only appear in the min-cut partition when the cost of cutting them from the image is smaller than the cost of cutting out the source or sink regions themselves. Secondly, in order to detect salient boundaries, the partitioning of the source and sink regions should match a partitioning of the natural image segments. Finally, the set of all partitions generated from source/sink pairs must

(a) (b) (c)

Figure 1: a) A 100×100 fractal image with four surfaces. b) Two projected blur kernels for initial pixels $\vec{x}_i$ at red dots. c) Linear-anisotropically smoothed image. For (b,c) we used $d = 10$ and $t = 750$.

be sufficient to pick out each desired image segment. In the next sections we explore the use of random walks and spectral embedding for the automatic construction of source and sink regions, with the explicit goal of meeting these three criteria.

3 Random Walks biased by Pixel Similarity

We follow [11] in defining a Markov chain representing a random walk over image pixels. In particular, suppose a particle is at pixel $\vec{x}_j$ at time t of the random walk. Then the probability that the particle jumps to pixel $\vec{x}_i$ at the subsequent time $t+1$ is taken to be proportional to the affinity $A_{i,j}$. The appropriate normalization constant is $D_j \equiv \sum_{k=1}^{nm} A_{k,j}$, and the transition probability is therefore $A_{i,j}/D_j$.

Let $p_t(\vec{x}_j)$ denote the probability that the particle undergoing the random walk is at pixel $\vec{x}_j$ at time t. To write the random walk in matrix form, we use the nm-dimensional vector $\vec{p}_t$ to represent $p_t(\vec{x})$. Here the vector $\vec{p}_t$ is formed from the values of $p_t(\vec{x})$ using a raster ordering of the image pixels. Defining D to be the diagonal matrix formed from the normalization factors D_j, the probability distribution for the particle at time $t+1$ is then given by

$$\vec{p}_{t+1} = M\vec{p}_t, \quad \text{where } M = AD^{-1}. \tag{2}$$

Note $M_{i,j} = A_{i,j}/D_j$, in agreement with the transition probability we constructed above.

Given an initial distribution $\vec{p}_0$, it follows from (2) that the distribution after t steps of the random walk is $\vec{p}_t = M^t\vec{p}_0$. Here the matrix power M^t can be simply represented in terms of the eigenvectors and eigenvalues of M, which are conveniently obtained from the *similar, symmetric* matrix $L = D^{-1/2}MD^{1/2} = D^{-1/2}AD^{-1/2}$. Since L is symmetric, the eigen-decomposition has the form $L = U\Lambda U^T$, where U is an orthogonal matrix whose columns are the eigenvectors of L, and Λ is the diagonal matrix of eigenvalues λ_i, $i = 1,\ldots,nm$. We assume that the eigenvalues have been sorted in decreasing order of their absolute value, so $|\lambda_i| \geq |\lambda_{i+1}|$ for all $i \geq 1$. Moreover, from the form of A and D it can be shown that $\lambda_i \in (-1,1]$ and there is at least one eigenvalue equal to one [10]. Therefore, without loss of generality, we can take $\lambda_1 = 1$. Finally, it can be shown that the associated eigenvector $\vec{u}_1$ with eigenvalue 1 is given by

$$\vec{u}_1^T = (u_{1,1}, u_{1,2}, \ldots, u_{1,nm}), \quad \text{with } u_{1,j} = \frac{1}{\alpha}D_j^{1/2}, \text{ and } \alpha = \sqrt{\sum_{k=1}^{nm} D_k}. \tag{3}$$

Using this factorization of L we can write

$$\vec{p}_t = M^t \vec{p}_0 = D^{1/2} L^t D^{-1/2} \vec{p}_0 = (D^{1/2} U) \Lambda^t \vec{c}_0, \quad \text{with } \vec{c}_0 = U^T D^{-1/2} \vec{p}_0. \tag{4}$$

There are two other properties of this construction that we will need. First, by (3),

$$\vec{1}^T D^{1/2} U = \alpha \vec{u}_1^T U = \alpha(1, 0, \ldots, 0), \tag{5}$$

where $\vec{1}$ is the vector of all ones. Secondly, from (5) it can be shown that $\vec{1}^T \vec{p}_t = \vec{1}^T \vec{p}_0 = 1$, so the distribution $\vec{p}_t$ remains properly normalized.

3.1 Linear Anisotropic Smoothing

Here we investigate the properties of $\vec{p}_t$ further, and show how these distributions can be used to describe anisotropic image neighbourhoods. Suppose we start the random walk from pixel $\vec{x}_i$, so the initial distribution is given by $\vec{p}_{0,i} \equiv \vec{e}_i$, where $\vec{e}_i$ is the standard unit vector with the value 1 in the i^{th} row and zeroes elsewhere. Then after t steps we obtain the diffused distribution $\vec{p}_{t,i} = M^t \vec{e}_i$ (i.e. the i^{th} column of M^t), which we refer to as an averaging (or blur) kernel. In Fig. 1b we show two of these kernels $\vec{p}_{t,i}$ for different initial pixels $\vec{x}_i$. Here the affinities have been computed as in (1) for the image in Fig. 1a. Note the significant spread in $\vec{p}_{t,i}$ across regions with relatively small brightness variations. Moreover, this spread is reduced across weak image edges (eg. see the top-left and bottom-right corners in Fig. 1b), and nearly eliminated across strong image edges.

It is instructive to form an anisotropically smoothed image using these blur kernels. Let $\vec{I}$ be the rasterized image, and $\vec{B}$ the rasterized smoothed image. Define B_j, the value of the smoothed image at pixel $\vec{x}_j$, to be the weighted average of image gray-levels, where the weights are given by the blur kernel initialized at pixel $\vec{x}_j$. That is, $B_j \equiv \vec{I}^T \vec{p}_{t,j}$, where $\vec{p}_{t,j}$ is given by (4) for $\vec{p}_0 = \vec{e}_j$. Fig. 1c shows the result of this averaging process. Note that strong image discontinuities are preserved due to the anisotropic nature of the blur kernels, but weaker local brightness variations are smoothed out. This is similar to standard anisotropic smoothing [13], except there the affinities are iteratively updated as the image is smoothed. Here, instead, the affinities are held fixed, according to the values dictated by the original image.

It turns out that most of the eigenvalues λ_i have magnitudes significantly smaller than one and therefore, for sufficiently large t, the product $\Lambda^t \vec{c}_0$ in (4) can be approximated by a projection onto the first few coefficients (recall that we sorted $|\lambda_i|$ in decreasing order). In particular, let R_d be the $d \times mn$ rectangular matrix with the $d \times d$ identity matrix in the left-most block, and zeros elsewhere. Then, by neglecting the powers λ_i^t for $i > d$ and using (4), we find

$$\vec{p}_{t,i} \approx \vec{q}_{t,i} \equiv (D^{1/2} U_d) \vec{w}_{t,i}, \quad \text{with } \vec{w}_{t,i} = \Lambda_d^t U_d^T D^{-1/2} \vec{e}_i. \tag{6}$$

Here $U_d = U R_d^T$ is the first d columns of U, and Λ_d is the $d \times d$ diagonal matrix formed from the first d eigenvalues λ_i. We refer to $\vec{q}_{t,i}$ as a projected blur kernel, and note that it is an element of a d-dimensional linear subspace. The use of $\vec{q}_{t,i}$ in place of $\vec{p}_{t,i}$ provides an efficient way to compute the anisotropically smoothed image $\vec{B}$. The error in the approximation of $\vec{p}_{t,i}$ by $\vec{q}_{t,i}$ depends on the magnitude of the terms $|\lambda_i|^t$ for $i > d$, the largest of which is $|\lambda_{d+1}|^t$. In practice, we find a good approximation when d and t are chosen such

that $|\lambda_{d+1}|^t \leq 1/3$. For smaller values of d or t ringing effects due to the projection can be significant in $\vec{q}_{t,i}$, much like the ringing due to the severe truncation of a Fourier series.

Given t and d, each pixel $\vec{x}_i$ can be mapped to the d-dimensional vector $\vec{w}_{t,i}$ according to equation (6). This vector $\vec{w}_{t,i}$ provides coefficients for the projected blur kernel $\vec{q}_{t,i}$ centered at that pixel. For $t = 0$ this mapping is closely related to the standard Laplacian embedding [1]. The derivation here extends Laplacian embedding to positive values of t and, moreover, explicitly makes the connection between the embedding and the (projected) blur kernels. We will make considerable use of this connection.

4 Spectral Embedding

Two additional properties of the embedded vectors $\vec{w}_{t,i}$ are of interest. First, it follows from (5) and (6) that the first component of $\vec{w}_{t,i}$ equals $1/\alpha \equiv 1/\sqrt{\sum_{k=1}^{nm} D_k}$, a constant. Secondly, if two kernels $\vec{p}_{t,i}$ and $\vec{p}_{t,j}$ have their probability masses on two different subsets of pixels, then their inner-product $\vec{p}_{t,i}^T \vec{p}_{t,j}$ must vanish. This inner product can be approximated using the projected kernels as $\vec{p}_{t,i}^T \vec{p}_{t,j} \approx \vec{q}_{t,i}^T \vec{q}_{t,j} = \vec{w}_{t,i}^T Q \vec{w}_{t,j}$, where $Q = U_d^T D U_d$. We can simplify this $\vec{w}$-space geometry by using the coordinate transform

$$\vec{z} = Q^{1/2} \vec{w}. \tag{7}$$

It follows that $\vec{q}_{t,i}^T \vec{q}_{t,j} = \vec{z}_{t,i}^T \vec{z}_{t,j}$.

The values of second and third components of $\vec{z}_{t,i}$ are shown in Fig. 2a (the first component is nearly constant and is omitted). In general, there are several important observations to make. First, small isolated regions show up as tight clusters in this embedding (see also [12]) indicating that the blur kernels for all the pixels in the region are very similar. This is exhibited to some extent by region 11 in Fig. 2a,b, although the separation is much larger in other dimensions of $\vec{z}$ and is not so apparent in this projection. Secondly, for smooth image ribbons (or regions) the points in the embedding are clustered around 1D (or 2D) sets. For example, in Fig. 2a there appears to be a smooth path from region 7 to 3 to 9 to 4. This corresponds to the top-most surface visible in the image (see Fig. 1a), along with the 'leak' to region 4. Third, image brightness discontinuities which induce small affinities on edges which span them appear in the embedding as sparsely populated valleys separating two more populated sets. For example, the area in Fig. 2a between regions 7 and 2 is sparsely populated by points on the corresponding surface boundary apparent in Fig. 1a. If the affinities spanning the edge are small enough, then the blur kernels on either side of such an image edge will be nearly orthogonal, and therefore the width of this valley in $\vec{z}$-space will be nearly 90 degrees. Weaker edges will generate blur kernels on either side which overlap more, and their relative angles will be smaller than 90 degrees. In some cases, such as the transition between seeds 9 and 4 in Fig. 2a,b, weak edges may appear to be blended in the embedding.

Finally, we note that the embedding $\vec{z}_{d,t,i}$ computed for a particular dimension d and scale t is related to the embedding $\vec{z}_{d',t',i}$ for some other pair (d',t') with $d' \leq d$ by a linear transformation, $\vec{z}_{d',t',i} = Z \vec{z}_{d,t,i}$ where Z is a $d' \times d$ constant matrix (i.e. it does not depend on i, see (6) and (7)). This implies that $\vec{z}$-space is only distorted in an affine manner when the scale t and dimension d are changed. Thus, while relative angles between various points may change with t and d, the global arrangement of clusters, ridges and valleys in the general distribution of points can only change in a linear manner.

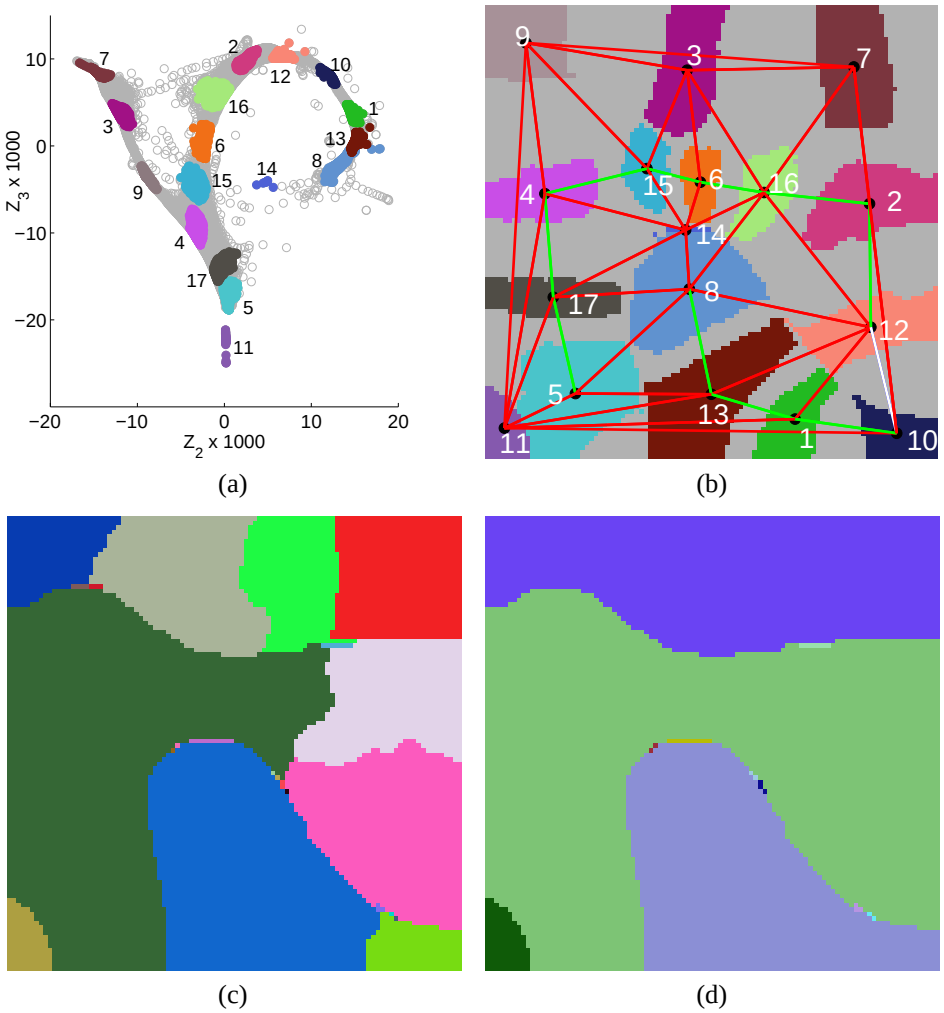

Figure 2: a) Embedded coordinates (z_2, z_3) for the image in Fig. 1a. The clusters found using seed selection are shown. b) Seed regions in the image corresponding to z-space clusters, along with the Delaunay triangulation of seed centroids. The edge colour (red, green, light gray) indicates the edge weight $f_{k,j}$ (region discontinuity, continuous blend, ambiguous, respectively). Note that this is a 2D-projection of a plane in 10D so, for example, the connections between regions 9, 3, and 7 are not as straight as they look here. c) Segmentation provided by the intersection of all min-cut partitions generated by the source/sink proposals using the seed regions and edge classification shown in (b). d) Segmentation after region merging. Throughout we used $d = 10$ since this generated an appropriate seed size. We used $\lambda_d^t = 1/3$ to select $t = 750$.

5 Source and Sink Region Selection

According to the criteria listed in Sec. 2 for the selection of source and sink regions for min-cut, we wish to choose sufficiently large regions which are contained within natural image segments. We propose to do this by locating clusters of high density in $\vec{z}$-space. These clusters correspond to pixels which have similar blur kernels, and thus are likely to avoid strong image boundaries.

We do the clustering using a simplification of a mixture model density estimation procedure [2] applied to the $\vec{z}$ points mapped to the unit sphere, namely $\vec{s}_{t,i} \equiv \vec{z}_{t,i}/||\vec{z}_{t,i}||$. For each component density in this mixture model we use $K(\vec{s},\vec{m}) = \beta \max(\vec{s}^T\vec{m} - \tau_0, 0)$, where β is a normalization constant, $\vec{m}$ is a unit vector representing the mean direction of the cluster, and τ_0 is a fixed parameter which determines the maximum angular width of the kernel. We use $\tau_0 = 0.8$. The density estimate is then $p(\vec{s}) = \sum_{k=1}^{M} \pi_k K(\vec{s},\vec{m}_k)$, where π_k are the mixing coefficients. In general, the mixing coefficients π_k must be non-negative and sum to one, but here, for simplicity, we take the specific values $\pi_k = 1/M$. The only unknowns in $p(\vec{s})$ are then the cluster centers $\vec{m}_k$, and these are fitted to the observations $\{\vec{s}_{t,i}\}_{i=0}^{nm}$ using a standard EM-algorithm.

The initial guess for this algorithm is obtained by randomly sampling points $\vec{s}_{t,i}$, with successive samples constrained to have an inner-product of at least τ_0 from previous samples. The number of clusters, M, is chosen to be the maximum number of $\vec{m}_k$ that can be chosen before all points $\vec{s}_{t,i}$ have an inner-product of at least τ_0 with some center $\vec{m}_k$.

Given the cluster centers $\{\vec{m}_k\}_{k=1}^{K}$ fitted with the EM-algorithm, we determine corresponding image "seed" regions S_k by thresholding the inner-products of $\vec{s}_{t,j}$ with $\vec{m}_k$ as follows

$$S_k \equiv \{\vec{x}_j \,|\, \vec{z}_{t,j}^T\vec{m}_k \geq \tau_1||\vec{z}_{t,j}||\}, \tag{8}$$

where $\tau_1 > \tau_0$ is taken to be 0.985 (see Fig. 2a,b for a typical result from this process).

The source and sink regions we choose to use for min-cut are disjoint unions of the seed regions $\{S_k\}_{k=1}^{M}$. The difficulty here is that several seed regions often belong to the same image segment (see Fig. 2b) and, according to criterion 2 in Sec. 2, these should not be split between the source and sink regions. When they are split we have observed that the min-cut algorithm often fails to find a salient cut.

We therefore developed a method for generating source/sink proposals which considers evidence on whether or not two seed regions belong to the same image segment. This is based on a Delaunay triangulation of the centroids of the seed regions S_k. We classify edges between regions S_k and S_j in this triangulation one of three possible ways based on evidence from the embedding on whether or not there is a smooth continuation between S_k and S_j. The specific evidence we use is the relative density of points in $\vec{z}$-space along a path between the seed centers $\vec{m}_k$ and $\vec{m}_j$. The reason for this choice is the observation made in the previous section that image boundaries show up as rapid changes in the blur kernel coefficients, generating valleys in the density of points in the embedding (see Fig. 22a). The specific figure of merit we use is

$$f_{k,j} = \min_{r\in[0,1]} \frac{|\{\vec{x}_i \,|\, \vec{z}_{t,i}^T\vec{m}_{k,j}(r) > \tau_1||\vec{z}_{t,i}||\}|}{|S_k|r + |S_j|(1-r)}, \tag{9}$$

where $|\cdot|$ denotes set cardinality and $\vec{m}_{k,j}(r)$ is the projection of a convex combination of $\vec{m}_k$ and $\vec{m}_j$ onto the unit sphere (i.e. $\vec{m}_{k,j}(r) = \vec{n}_{k,j}(r)/||\vec{n}_{k,j}(r)||$ for $\vec{n}_{k,j}(r) = r\vec{m}_k + (1-$

Figure 3: (Left column). Test images of indoor scenes. The top image is 239×138, and the two below are 160×120. (Center column) Segmentation results for our algorithm. (Right column) Segmentations using Felzenszwalb and Huttenlocher's algorithm.

$r)\vec{m}_j$).[1] For $f_{k,j} < 0.2$ we classify the link between S_k and S_j as having evidence of an image region discontinuity. For $f_{k,j} > 0.3$ we consider the evidence to be in favour of a smooth continuation between the regions, and for intermediate values of $f_{k,j}$ we classify the evidence as ambiguous.

The resulting classification is depicted in Fig. 2b. Note that the derived classification is appropriate, other than for the edges between regions 9, 3, and 7, which belong to the same surface. The issue here is that the embedding between these regions is curved in dimensions that have been projected out in Fig. 2b. Since the seed regions themselves are relatively far apart, the linear interpolation used in (9) causes a gap to be detected in the embedding. However, as we show below, the results using this simple classification appear to be adequate.

We then generate source/sink proposals by selecting each seed region S_k in turn. Given S_k and its immediate neighbours in the Delaunay triangulation, say $S_{j(n)}$ for $n = 1, \ldots, N$, we generate all combinations of source and sink proposals (using only these neighbouring seed regions) which satisfy the following: 1) S_k is in the source; 2) if the edge between S_k and $S_{j(n)}$ is classified as a clear image boundary (i.e. $f_{k,j(n)} < 0.2$) then $S_{j(n)}$ is in the sink; and 3) if the edge between S_k and $S_{j(n)}$ is classified as a smooth continuation (i.e. $f_{k,j(n)} > 0.3$) then $S_{j(n)}$ is either in the source or omitted. This results in $2^{N_s}3^{N_a}$ distinct source/sink proposals for seed region S_k, where N_s and N_a are the number of edges in the Delaunay triangulation with one endpoint on region S_k that are classified as smooth or ambiguous, respectively. In practice we have found this to be tractable since the total

[1] We estimate the minimum in (9) simply by sampling r at 11 equally spaced points.

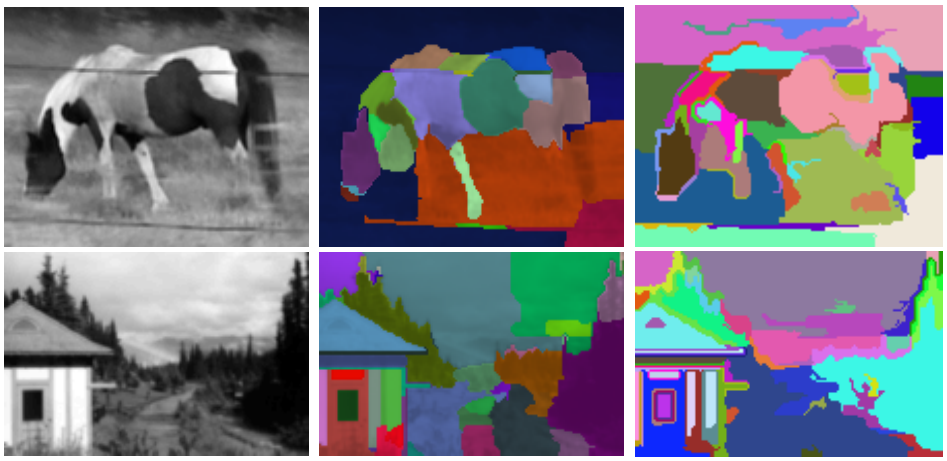

Figure 4: (Left column). Test images of outdoor scenes. The top image is 154×121, and the one below is 148×98. (Center column) Segmentation results for our algorithm. (Right column) Segmentations using Felzenszwalb and Huttenlocher's algorithm.

number of neighbours, N, is typically small and $N_s + N_a \leq N$. For example, in Fig. 2b the most complex case occurs for region 12, for which $N_s = N_a = 1$.

The min-cut algorithm is run on each such proposal which has a non-empty sink region. The resulting partitions are intersected to form an over-segmented image (see Fig. 2c). The resulting segments are then merged based on the observation that a cut through an image boundary should consist mostly of links with small weights. We check the distribution of affinities at the boundary between two regions, and if less than $1/4$ of these links are reasonably weak, the regions are merged (see Fig. 2d).

6 Examples

This approach was implemented in Matlab using the push-relabel min-cut code described in [6][2]. Also, we used a Matlab implementation of the multi-scale SVD algorithm [5] to generate eigenvectors U_d and eigenvalues Λ_d of the large sparse symmetric matrix L (see eqn. (6)). In generating the seed regions we observed that increasing the dimension d generally caused more and smaller regions to be selected. We found $d = 25$ was sufficient in all the above examples to form seeds within each salient region. Finally, t, the number of steps in the random walk, was set according to the equation $\lambda_d^t = 1/3$ mentioned at the end of Sec. 3.1. Other parameters of the method were fixed at the values specified previously.

Sample results are shown in Figs. 3 and 4 where, for comparison purposes, we also show results from Felzenszwalb and Huttenlocher's algorithm [7]. The free parameter in this latter algorithm was selected to give the most favorable comparison with our result. Overall results are generally comparable, with our algorithm respecting image borders to a greater degree. We find this extremely encouraging since here we are using the simple default image affinities described in (1), while Felzenszwalb and Huttenlocher's algorithm

[2]We thank A.V. Goldberg for supplying C-code (i.e. `hi_pr.c`) for this algorithm.

uses information about the gray-level variation within each region.

For the images in Figs. 3 and 4, and others of similar sizes, our algorithm running on a 2.5 GHz. P4 machine requires roughly 30 secs. to compute the 40 leading principal vectors of L (roughly a 20K $\times$ 20K matrix!), seed region generation takes between 30 sec. and 1 min., the computation of minimum cuts between source and sink regions takes from 1 to 3 minutes, and the final merging stage consumes between 1 and 2 minutes. The variability in run-times is the result of having a different number of seed region proposals for each image. In contrast, Felzenszwalb and Huttenlocher's algorithm takes less than 5 sec.

In terms of our overall objectives, the results indicate that we have been successful in automatically generating proposals for source and sink regions which satisfy the criteria, as listed in Sec. 2, for the automatic application of the min-cut image segmentation approach.

Acknowledgements We thank Sam Roweis for his useful comments.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In "T.G. Diettrich, S. Becker, and Z. Ghahramani", editors, *"Adv. in Neural Inf. Proc. Sys. 14"*, pages 585–591. MIT Press, 2002.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford Univ. Press, 1995.

[3] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, pages 359–374, 2001.

[4] Yuri Y. Boykov and Marie-Pierre Jolly. *Interactive Graph Cuts* for optimal boundary & region segmentation of objects in n-d images. In *IEEE International Conference on Computer Vision*, pages 105–112, 2001.

[5] S. C. Chennubhotla. *Spectral Methods for Multi-Scale Feature Extraction and Data Clustering*. PhD thesis, Dept. of Computer Science, Univ. of Toronto, 2004.

[6] Cherkassky and Goldberg. On implementing push-relabel method for the maximum flow problem. In *Proc. IPCO-4*, pages 157–171, 1995.

[7] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *Proc. of IEEE CVPR*, pages 98–104, 1998.

[8] Yoram Gdalyahu, Daphna Weinshall, and Michael Werman. Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1053–1074, Oct. 2001.

[9] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, 1996.

[10] J. R. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand, 1960.

[11] M. Meila and J. Shi. A random walks view of spectral segmentation. In *Proc. International Workshop on AI and Statistics*, 2001.

[12] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Proc. NIPS*, 2001.

[13] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[14] Olga Veksler. Image segmentation by nested cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 339–344, 2000.

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