Model Order Selection and Cue Combination for Image Segmentation

Andrew Rabinovich[1] amrabino@ucsd.edu
Tilman Lange[2] langet@inf.ethz.ch
Joachim M. Buhmann[2] jbuhmann@inf.ethz.ch
Serge Belongie[1] sjb@cs.ucsd.edu

[1]Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA

[2]Institute of Computational Science
ETH Zurich
CH-8050 Zurich, Switzerland

Abstract

Model order selection and cue combination are both difficult open problems in the area of clustering. In this work we build upon stability-based approaches to develop a new method for automatic model order selection and cue combination with applications to visual grouping. Novel features of our approach include the ability to detect multiple stable clusterings (instead of only one), a simpler means of calculating stability that does not require training a classifier, and a new characterization of the space of stabilities for a continuum of segmentations that provides for an efficient sampling scheme. Our contribution is a framework for visual grouping that frees the user from the hassles of parameter tuning and model order selection: the input is an image, the output is a shortlist of segmentations.

1. Introduction

The goal of clustering (or segmentation, or grouping) is to partition n objects into k groups so as to optimize an objective function. One way of thinking of the objective function is that it imposes a ranking on the set of all partitions. While this is a convenient tool for intuition, when k is unknown, the size of this set grows exponentially in n. Compounding the problem is the fact that most clustering algorithms possess a variety of parameters on the objective function that weight different features (or cues) of the objects. In the case of image segmentation, these features include position, color, texture, motion, and so on. As such, the problems of choosing k (model order selection) and the relative parameter weightings (cue combination) are difficult open problems.

Fortunately, the various domains in which clustering is applied often enjoy properties that can be leveraged against the above problems. In this work, our domain of interest is visual grouping. In this setting, k is often fairly small, e.g., 20, and the various parameters can be restricted into narrow valid ranges. Nonetheless, depending on the number of cues employed and the granularity of their variation, this can still present substantial problems both in the sense of computation and of usability.

The present work addresses both of these problems. We begin with the observation that no single value of k is correct in general. The literature on model order selection is perhaps surprisingly focused on selecting one 'best' value of k [2, 13] (and references therein). A similar situation exists in the scale selection literature [10, 15]. To consider multiple values of k, we stop short of exhaustively searching the space of parameter values, however, by observing that this space has structure that allows one to characterize the space via an efficient sampling scheme. Our contribution is a framework for visual grouping that frees the user from the hassles of parameter tuning and model order selection: the input is an image, the output is a shortlist of segmentations. The formalism we adopt in pursuit of this framework is a quantification of a *stable clustering*, corresponding to the intuition that a clustering is good if it is repeatable in the face of perturbations.

As a preview of this idea, consider the dataset shown in Figure 1. This stimulus consists of four clumps of points. For simplicity, we consider two cues: proximity and density, where the latter is measured by counting the number of points that fall inside a box centered on each point.[1] We show three representative stable clusterings, two for $k = 2$ and one for $k = 4$. Depending on the relative cue weighting, one can obtain two very different stable clusterings for $k = 2$. For $k = 4$, however, there is only one stable clustering. Associated with each case is a range of parameter values (cue weighting and box width) that lead to the same result. The myriad other unstable clusterings are not of interest to us. In this work we aim to select only meaningful clusterings for a given dataset.

The organization of this paper is as follows. In Section 2 we review relevant work on stability based clustering and

[1]This measure of local density is a simplified instance of a texture descriptor; it generalizes to a local texton histogram [16].

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

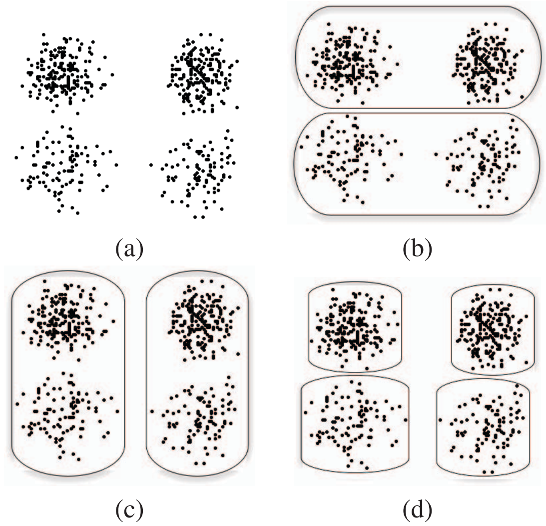


Figure 1. (a) Original stimulus of four clumps of points with varying density. Two stable clusterings for $k = 2, 4$, shown in (c) and (d), are based on Euclidean distances between points. Stable clustering for $k = 2$, based on point density (texture descriptor), is shown in (b). There are two other trivial stable solutions for $k = 1$ and N, where N is the cardinality of the set.

suggest a simplification to an existing model that will allow for multiple stable solutions and reduce the algorithmic complexity. In Section 3, we motivate the cue combination problem in a framework of stability based segmentation evaluation. We demonstrate the behavior of the proposed approach on images of tissue samples and examples from the Berkeley Segmentation Database in Section 4. We conclude in Section 5 with the discussion and future work.

2. Stability Based Clustering

Stability based clustering is a relatively new approach to model order selection. In late 1980s Jain and Dubes [9] discussed the validity of a given clustering structure based on hypothesis testing. The boom of work on finite mixture models in the 1990s gave rise to numerous approaches based on information theoretic criteria such as MDL, AIC, and BIC [1]. More recently a class of approaches based on stability have shown great promise. Our work falls into this category; we provide a brief review of it next.

2.1. Model Order Selection

The approach of stability based model order selection (from [13]) is as follows. Given a dataset, the data points are split into two disjoint subsets $\mathcal{A}$ and $\mathcal{B}$. Using some clustering method, cluster $\mathcal{A}$ into k groups. Once the clustering for a given k is found and all the points in $\mathcal{A}$ are labeled, a classifier ϕ is chosen and is trained using the labels from the clustering algorithm. Once the classifier (the predictor) is trained, the subset $\mathcal{B}$ is considered. The data in $\mathcal{B}$ is clustered into k groups and independently labeled using ϕ. Then the labels from the clustering and classification are compared to determine the stability. Care must be taken here since the labeling is arbitrary up to a permutation. To address this, one can perform pairwise comparison between points (i.e., are the two points in the same group or not) or find an optimal label permutation, e.g., using the Hungarian method [11]. Finally, the number of points with the same label provide a stability measure for that value of k. This procedure is repeated for a range of k's.

This approach is well motivated and we adopt it with the following modifications:

1. Instead of splitting the data in two, cluster the entire set with a given k. The clustering engine can be a central method such as k-means if the clusters are spherical, or a pairwise method such as NCut [5, 25] if not; for generality assume the data is not spherical and use a pairwise method.
2. Once the data is clustered and the data points labeled, add noise (proportional to the variance of the data, discussed in Section 3.2.2) to slightly perturb the pairwise distances.[2] Then, perform clustering for the same number of groups and assign new labels to the data. Such a labeling scheme avoids the use of a classifier, and reduces the algorithmic complexity. This perturbation is performed many times; here we re-clustered the data 50 times, yielding 50 different labelings for the data points.
3. Given all of the labelings, permute all but one of them to best match the labeling held constant, an anchor, and compute the stability according to the following definition:

$$S(k) = \frac{1}{n - \frac{n}{k}} \left(\sum_{i=1}^{n} s_i - \frac{n}{k} \right) \quad (1)$$

where $s_i \in [0, 1]$ indicates the agreement between the labels over all perturbed restarts for the ith point in the data, the fraction n/k prevents a bias for a particular value of k, and $n - n/k$ is the normalization coefficient. Since any given anchor could be suboptimal, we try all possible anchors and pick the one that yields highest stability.

3. Visual Cue Combination

3.1. Segmentation Stability with Given Parameters

Unlike the case of point sets, where Euclidean distance may be used to assess the similarity between data points, image segmentation is best performed using multiple visual cues [16, 22, 27]. How to choose which cues to consider for a given segmentation problem and how to weigh their importance is unclear. This is known as the cue combination problem in computer vision. Traditionally, supervised

[2] One could instead perturb the positions of each data point. See Section 3.2.2 for details.

learning approaches are used to address cue combination in a given application. Based on labeled data, a classifier is trained to choose the appropriate weighting for each cue [17, 18]. In other approaches, the combination of cues is set manually [8]. Since we do not assume human labeled examples are available, we propose to use the stability based model order selection approach to identify all possible combinations of cues and numbers of groups that lead to a stable segmentation. The stability calculation process remains unchanged; however, with every new combination of cues, the grouping criterion changes.

One possible approach to combining cues is to construct an affinity matrix using a convex combination:

$$W_{ij} = e^{-(C_{1,ij} \cdot p + C_{2,ij} \cdot (1-p))} \tag{2}$$

where $p \in [0, 1]$ specifies the cue weighting, C_1 and C_2 are the individual dissimilarities for each of the cues, and W is the overall affinity between points based on both cues. Each of the dissimilarities C_1, C_2 and affinity W have an internal scaling parameter, σ, that is used to maximally separate the dissimilar entries and group the similar entries in the matrix. We discuss the selection of this parameter next.

3.2. σ Estimation and Re-Sampling using Non-Parametric Density Estimates

In this section we estimate the scaling parameter for each cue individually and propose a re-sampling scheme for data perturbation. We assume that similarities C_{ij} correspond to *squared Euclidean* distances $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ in a suitable (embedding) space and that we have access to a representation of the data (one for each cue). If only similarities between data points are available, a vectorial representation in $\mathbb{R}^d$, $d \leq n-1$, can be obtained given that the similarity matrix fulfills *Mercer's condition*, i.e., if we have a kernel matrix [3]. Then, the application of Kernel Principal Component Analysis (kPCA) [23] results in an *isometric* embedding of the corresponding distances into an $n-1$ dimensional space (after centering the kernel matrix).

3.2.1 σ Estimation

For each cue we obtain a set of n realizations $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ of the random variable X with unknown density $\gamma(\mathbf{x})$. The guiding principle of the stability approach is to require segmentations to be robust with respect to fluctuations in the source, i.e., in $\gamma(\mathbf{x})$. If the density were known, this condition would have been easily checked for by drawing multiple samples from γ. However, in practice, we do not have access to γ. Thus, we adopt the following strategy: Instead of using the density γ, we construct a non-parametric density estimate q, the latter being used for the re-sampling and the subsequent stability assessment.

To obtain a non-parametric estimate of the density $\gamma(\mathbf{x})$ a *Parzen window* estimator is used. The density is approximated by a super-position of basis functions, being centered around the realizations $\mathbf{x}_i$, $i \in \{1, \ldots, n\}$ [4, 7, 26]. One possible choice for the underlying kernel function is a Gaussian kernel: the kernel centered at $\mathbf{x}_i$ reads

$$k_{\mathbf{x}_i,\sigma}(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right). \tag{3}$$

The density estimate q_σ is then a super-positon of the individual density estimates:

$$\gamma(\mathbf{x}) \approx q_\sigma(\mathbf{x}) := \frac{1}{n} \sum_{1 \leq i \leq n} k_{\mathbf{x}_i,\sigma}(\mathbf{x}). \tag{4}$$

The estimate depends on a smoothness (a.k.a. bandwidth) parameter, σ, whose choice greatly influences the shape of the density estimate q_σ. It is well known, that doing (neg. log-) likelihood-cross-validation (asymptotically) leads to a consistent estimate of σ [28]. In essence, one tests $-\sum_{\mathbf{x}^{(t)} \in T} \log q_\sigma(\mathbf{x}^{(t)})$ for different choices of σ and a set of "test" points T. Finally, a σ is picked, for which this test quantity becomes minimal.

3.2.2 Re-sampling

$k_{\mathbf{x}_i,\sigma}(\mathbf{x})$ is a Gaussian density with variance σ^2 and the corresponding density estimate q_σ is a mixture of Gaussians with n modes, each having weight $\frac{1}{n}$ and the common variance with d-dimensional covariance matrix $\mathrm{diag}(\sigma^2, \ldots, \sigma^2)$. This Gaussian mixture is used to sample "noisy" versions of $\mathbf{x}_i$. In particular, we get a noisy version $\tilde{\mathbf{x}}_i$ of each *original* point $\mathbf{x}_i$ by sampling a substitute from the Gaussian $k_{\mathbf{x}_i,\sigma}$. Note that, in contrast to the original stability approach of [13], the point correspondence problem is automatically resolved in this case, as one can identify $\tilde{\mathbf{x}}_i$ sampled from $k_{\mathbf{x}_i,\sigma}$ with $\mathbf{x}_i$.

3.3. Just Noticeable Difference

Besides the scaling parameter, some cues have another internal parameter. For example the density cue has a window in which the density is computed. The size of this window, ω, is an internal parameter of the density cue (other cues such as proximity do not have such internal parameters and simply use the raw coordinates of points to determine similarity). Varying the values of such internal parameters has a particular effect on the overall stability of the clustering. If for example the window size ω is changed by a small fraction to $\omega + \epsilon$, the stability of a given grouping will not change as texture is captured equally with windows of negligibly different sizes. This is related to the phenomenon of the *Just Noticeable Difference* and *Weber's Law* [21]. Slight variations of the value of ω do not result in perceptually distinctive textures [12]. Thus, the segmentations produced with such different values of ω will be the same. However, such a rule does not apply to the variation of number of groups. If the stability of segmentations with all possible cue combinations is known for k groups, in general there can be nothing said about the stability behavior of grouping with $k \pm 1$ groups. In the example in Figure 1 there are stable solutions for $k = 2$ and $k = 4$ groups, however, no clustering with $k = 3$ is stable.

3.4. All Possible Segmentations for a Set of Cues

As discussed earlier, there may be more than one stable segmentation of a given image. Segments may be formed based on different cue combinations and/or model orders. To identify all parameter settings for which a segmentation is stable, it is intuitive to consider all possible combinations of parameter values: different numbers of groups, different cue weightings, and finally the internal parameters for each of the cues. Even if we restrict the range of parameters, e.g., 10 different values for k, 20 values for window size, and 10 for cue combination, for the example in Figure 1, there are still 2,000 segmentations to consider; see Figure 2. Although such a representation of the space of possible segmentations is very thorough and potentially useful, the brute force computation of these segmentations and stability values associated with them becomes computationally infeasible. Instead, we propose a sampling-based approach for approximating the space of segmentation stabilities. Since the behaviors of segmentations for different k's are decoupled, we must sample p and ω for every desired k.

To be able to analyze the behavior of the parameters independently, as was discussed in Section 3.3, the overall stability of a given segmentation is modeled as a product of stabilities of each individual cue. With A, a matrix of sampled segmentation stabilities, constructed, we use Non-negative Matrix Factorization (NMF) to decompose the overall segmentation stability values into segmentation stabilities of individual cue parameters.[3] NMF [14, 20] is a recently introduced method for finding non negative basis functions (vectors) that represent the data. Using an iterative approach with non-negativity constaints, a data mixture A is factored into constituent components S and the weights B for each component. Repeated iteration of the update rules is only guaranteed to converge to a locally optimal matrix factorization, however, practical applications of NMF indicate suitability of the approach. In its usual form, this decomposition is an additive one in terms of the learned components. Here, we set up the problem as multiplicative (this holds true in the rank-one case) and consider B and S to be the two basis functions. In doing the NMF, there is a constraint on non-negativity, yet there is no upper bound on the individual entries of the basis functions. Since the basis functions that we extract correspond to the stability value for individual cues for a given k, the entries in vectors B and S must be constrained to $[0, 1]$. To enforce the bounds on the values, we introduce an extension to the general NMF.

[3] In the presence of more than two cues, the use of NMF is generalized to Non-negative Tensor Factorization (NTF) [24, 30]. Each combination of parameters for each value of k becomes a tensor and is decomposed with NTF.

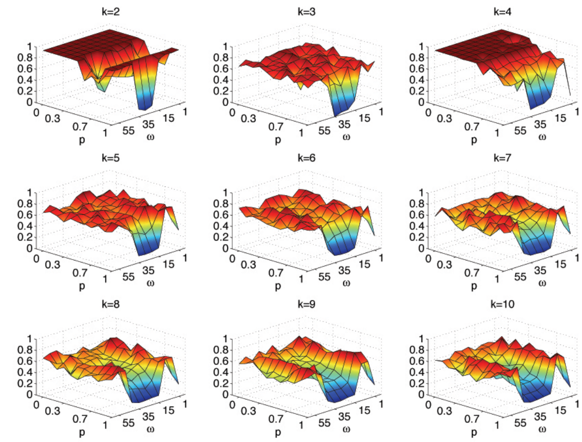


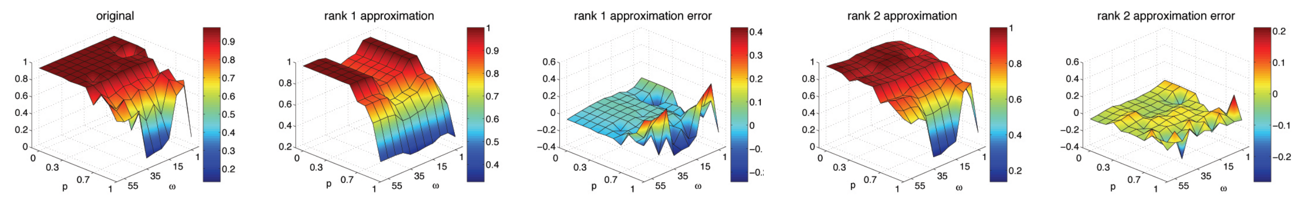
Figure 2. Slices of the cube of all possible segmentations for the 4 clumps stimulus (shown in Figure 1); the number of groups is indicated on top of every slice. All the stability values are in the range of $[0, 1]$. p is the cue combination coordinate in $[0, 1]$, ω is the window radius, an internal texture parameter, in $[1, 55]$. As expected, there are stable solutions for a range of cue parameters when grouping into $k = 2$ and $k = 4$ groups. It is important to note that although the slices for $k = 2$ and $k = 4$ show high stability, the slice for $k = 3$ is unstable. This underlines the decoupled behavior of the order in model selection.

3.5. Bounded Non-negative Matrix Factorizations

To achieve the desired bounds on the elements of B and S, the following procedure is based on a rank-1 decomposition; however, it is possible to achieve rank k decompositions of A with k rank-1 consecutive decompositions. Given $A = BS$, subject to $0 \leq B_{ij} \leq 1$, $0 \leq S_{ij} \leq 1$, the decomposition is an outer product of B and S assuming $rank(A) = 1$. To constrain the upper bound of elements of B and S, we wish to re-write $A = BS$ in terms of a function that is restricted on the interval $[0, 1]$. For example e^{-y}, if $y \geq 0$, is constrained in $[0, 1]$. Let $A' = -\log(A)$, $B' = -\log(B)$, $S' = \log(S)$, thus $A' = B' + S'$ is an instance of a least squares problem.[4] In particular, let $V = A'(:)$ (concatenated), $b = [B'(:, 1); S'(1, :)]$, and let

[4] Since A' is $m \times n$ and both B' and S' are vectors, the entries of B' and S' are repeated to achieve the $m \times n$ dimensions.

0-7695-2597-0/06 $20.00 © 2006 IEEE



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

Figure 3. Accuracy of bNMF approximation of the stability matrix A from Figure 1 for $k = 4$. (a) Original; (b) Rank 1 approximation of A using bNMF; (c) Error of rank 1 approximation; (d) Two successive rank 1 approximations; (e) Error of rank 2 approximation.

$$F = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ \dots & & & & & \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

F is $mn \times (m+n)$, where m and n are the lengths of B and S (the above example of F is shown for $m = 3$ and $n = 3$). Thus, $A' = B' + S'$ becomes:

$$V = Fb \tag{5}$$

In order to satisfy initial constraint of $0 \leq B_{ij} \leq 1$, $0 \leq S_{ij} \leq 1$, this least squares problem must be solved with the constraint that $b_i \geq 0$ (`lsqnonneg` in Matlab). By performing the above substitutions in reverse, B' and S' are recovered. Finally, we exponentiate $A' = B' + S'$ to obtain the bounded decomposition into B and S.

$$e^{-A'} = e^{-B'(:,1)} e^{-S'(1,:)} \tag{6}$$

$$A = BS \tag{7}$$

This is Bounded Non-negative Matrix Factorization with a rank-one constraint.

3.5.1 Projection onto the approximation subspace

Rank 1 approximation of the stability matrix may not be sufficient to represent the structure of A accurately; a higher order approximation may be required. Thus, it is necessary to quantify the performance of such an approximation.

The subspace spanned by the original full rank matrix A is projected onto the subspace spanned by the eigenvectors of the approximated matrix Q. In the case of rank 1 approximation, A is projected onto a line — Q's only eigenvector. More formally, we would like to find a combination of $\sum z_i q_i = A$, where each column of A is projected onto the subspace spanned by eigenvectors of Q, q_i. Let the set of eigenvectors spanning Q's subspace be Q_e. In matrix form:

$$Q_e Z = A \tag{8}$$

$$Q_e^\top (A - Q_e Z) = 0 \tag{9}$$

$$Q_e^\top Q_e Z = Q_e^\top A \tag{10}$$

$$Z = (Q_e^\top Q_e)^{-1} Q_e^\top A \tag{11}$$

Z is the projection matrix of A onto the eigenspace of Q. Thus, the projection of A onto Q's eigenspace is $P = Q_e Z$ and the residual of the projection P and original matrix A is $E = A - P$. Figure 3 illustrates the rank 1 and rank 2 approximations of the original matrix A, for the stimulus in Figure 1, and the residuals between A and the approximations. With a rank 1 approximation the residual error is 31.7%, while the residual error of the rank 2 approximation is only 4.26%.

3.5.2 Cue interpolation and approximation accuracy

Unlike k, a cue parameter such as point density within a window is not independent of its "neighbors". Visual cues may have a piecewise constant or monotonically changing behaviors and may be modeled as such. By having only a few stability values along the discrete interval of values for a given cue, it is possible to use a simple model to interpolate to the rest of the desired values for a particular cue. Since the actual cue combination is modeled as a convex combination, the behavior of that "axis" is continuous and smooth, and was fit using a simple bicubic interpolation. The behavior of stability as a function of box size for point density was modeled as piecewise constant. Once the vectors B and S are filled, their outer product fills the entire space of stability values for all segmentations for a given k.

To measure the quality of such a sampled approximation, we compare the approximated cube of stabilities to the actual one, where each stability value is computed according to our definition of stability. The approximation evaluation was carried out using the stimulus in Figure 1. As shown in Figure 4, with less than 20% of all possible combination of parameters for each k, an approximation of 90% accuracy is achieved. Accuracy was calculated via the projection procedure in Section 3.5.1.

3.6. Shortlist of stable segmentations

The set of stability values provides valuable information about the associated segmentations, but such a representation still requires manual "sifting" of the stable segmentations from the ones with low stability. Our aim is to output

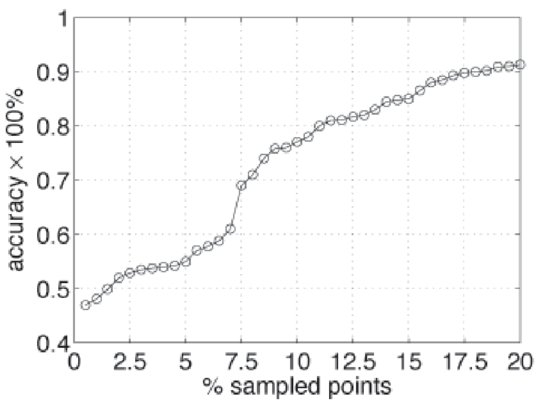


Figure 4. Evaluation of the accuracy of sampling the cube of stabilities to approximate its dense representation. The curve illustrates the agreement between two dense cubes of stability values, where the entries in the first cube are all explicitly computed and the entries in the second are the result of sampling and interpolating. By sampling less than 20% (out of 200 points in each plane of the cube) of the full cube, the sampling approach is able to achieve an accuracy of 90%.

only a small set of parameter values that lead to stable segmentations. To identify stable solutions, we adopt a hysteresis based thresholding approach. Due to the continuous nature of the behavior of visual cues, we only consider regions of high stability values rather than individual points of high stability in this space of segmentations, to avoid noise. We begin pruning the cube by choosing a point of high stability with an assumption that every image has at least one stable segmentation; the stability values in the neighborhood are grouped into plateaux (each plateau represents a unique segmentation) by region growing [6]. We enforce that at least 2 neighboring positions have a high value to consider this region to be stable and result in a plateau. Sequentially, another point of high stability, outside of the explored plateau, is considered, and the region exploration is repeated. Such a process is done until all values of high stability (above a certain threshold) have been considered. Currently, we set the upper, τ_u, and lower, τ_l, thresholds manually. In the "ropes" stimulus for example, we set $\tau_u = 0.974$ and $\tau_l = 0.691$.

Finally a set of parameters for cue combination, texture window size, and model order are output as a shortlist. This is the list of all possible parameters that provide stable clusterings. With images presented in this work, the shortlist of all possible stable segmentations reduced the size of the entire space of possible parameter combinations by more than 95%. The shortlist is a highly compact summary of the entire space of all segmentations. Some combinations of parameters may result in redundant segmentations and some segmentations may be stable but meaningless. Removing the parameters that yield incorrect and redundant segmentation is a subject of ongoing work.

stimulus	max k	total possible param. comb.	total # stable plateaux	% stable segmentations out of all possible	mean # of parameter combinations per plateau
ropes	20	5500	19	0.31%	29.4
clouds	20	5500	14	0.25%	45.7
flowers	50	13750	86	0.62%	8.1
tissue 1	100	27500	218	0.79%	11.6
tissue 2	100	27500	109	0.39%	38.9
tissue 3	100	27500	236	0.86%	12.2

Figure 5. Segmentation statistics. The percent is computed assuming that each of the plateaux of stable parameter combinations represents only 1 segmentation. In reality there could be more than 1, however, even if there are a few different segmentations per plateau, the fraction of stable ones will still be less than 5%.

4. Experimental Results

There are a number of domains where the existence of multiple segmentations for a given image is natural. The domain of biomedical images is one of them. Due to the hierarchical structure of the images, segmentations with multiple numbers of groups are natural. Also it is desirable to be able to identify segments based on different criteria, e.g., DNA content, protein expression and brain activity. Here we present examples of multiple stable segmentations of images of tissue biopsy samples. To explore the generality of our framework, we apply it to images from the Berkeley Segmentation Database (BSD) as well. In Figure 6 are segmentations of three images from BSD and three images of tissue samples. In all six examples the different segmentations are the results of varying the number of groups and the cue weightings (using texture and color). Averaged segment boundaries, in the 4^{th} column of the first 3 rows, from multiple subjects from BSD (darker boundaries indicate higher probability for a given set of human segmentations) further illustrate the presence of multiple stable segmentations and exhibit a high correlation with segmentations produced by our method. Table in Figure 5 shows the segmentation statistics of our method for the images from Figure 6.

Similarity of texture was measured using texton histograms with an internal parameter of texture window radius [16]. Similarity of color was based on the Euclidean distance of the hue channel in HSV color space. Binning kernel density estimates of the color distribution in CIELAB color space using a Gaussian kernel, and comparing histograms with the χ^2 difference may be perceptually more meaningful; however, the choice of color description is not central here. We chose the HSV representation for its simplicity. Given the similarities for each cue, the overall pairwise pixel affinity was computed according to Equation 2.

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

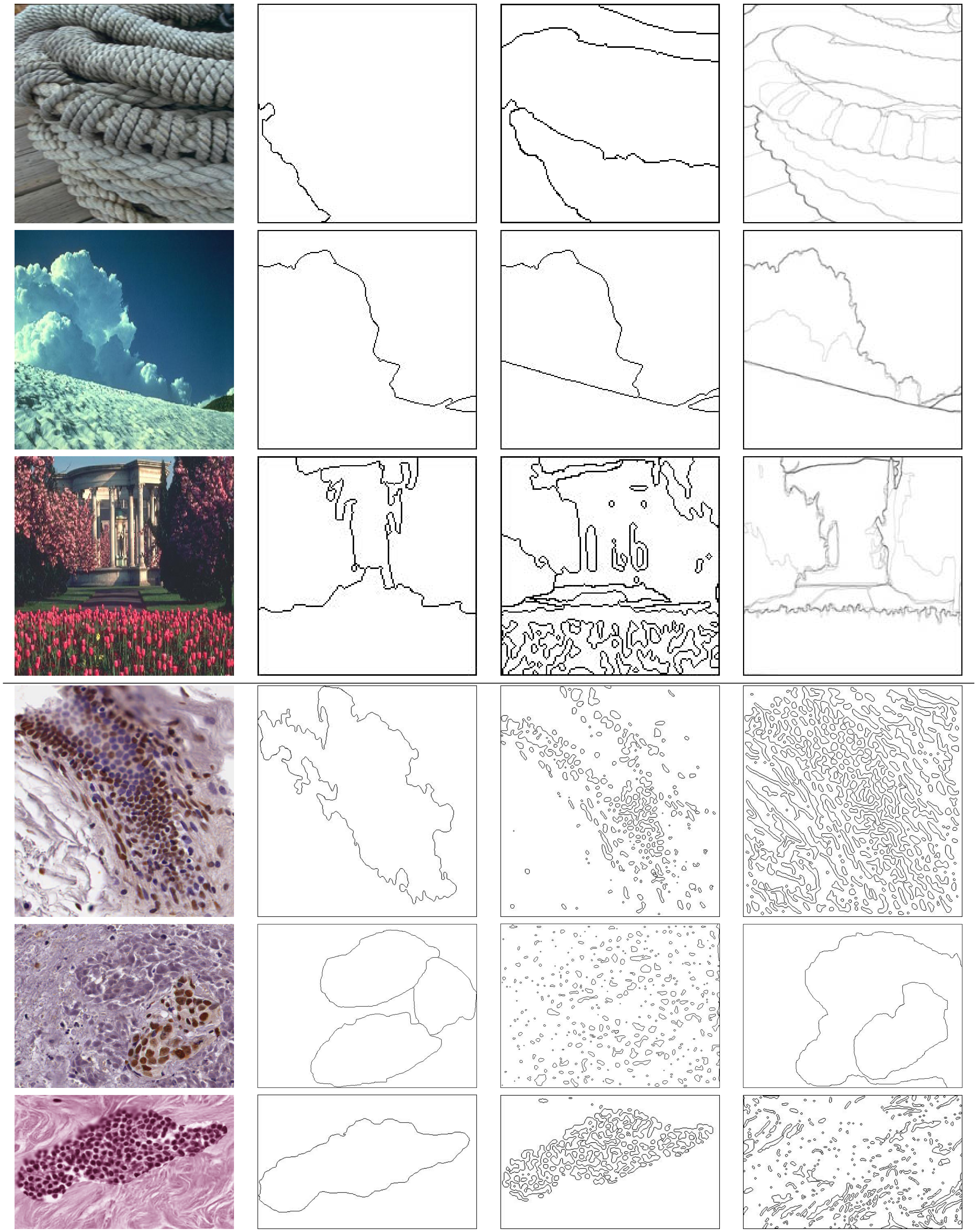


Figure 6. Examples of stable segmentations. Each is a result of a different choice of k, texture and color weighting. Only two and three stable solutions are shown for the BSD and tissue examples, respectively. In all examples, over 95% of all possible segmentations have low stability and are discarded. In column 4 of the first 3 rows we show averaged segment boundaries from multiple subjects from the BSD (darker boundaries indicate higher probability for a given set of human segmentations).

Once the combined affinity matrix W is constructed, using the proposed rank-one sampling approach twice, its entries are treated as edge weights of an undirected graph. A number of approaches, such as spectral clustering, cut such a graph based on some criterion [19, 25, 29]. In this work, we use Normalized Cuts [25], where the number of leading eigenvectors were set to k and were further thresholded using k-means clustering. The current algorithm is implemented in Matlab and on average takes 2.51 seconds of processing for each stability value on a dual 3.2 GHz processor with 2 GB of RAM. For example, the "ropes" image is 256×255 pixels and the full process took 0.76 hours (46 minutes) (without sampling it takes 3.83 hours). Note that current algorithm is highly parallelizable; we intend to exploit this in future work.

5. Discussion

In this work we proposed a framework that frees the user from the burden of manual parameter tuning and model order selection in the task of image segmentation. We leverage the observation that the number of possible segmentations of a dataset is significantly smaller than the number of parameter combinations of the segmentation algorithm; furthermore the number of stable segmentation is much smaller than the total number of possible segmentations. With an image as input, our method generates a shortlist of stable segmentations. The proposed approach performs well on medical images and examples from the Berkeley Segmentation Database.

After sampling a small fraction of the possible cue weightings and internal parameters, the sparsely populated space of segmentation stabilities is filled in using a novel extension to NMF that constrains both the upper and lower bounds of the elements of the extracted basis functions. Stabilities for individual cues and the weights are interpolated to the desired resolution and the full space of segmentation stabilities is reconstructed. Finally, only parameters (k, cue weights and cue internal parameters) that result in stable segmentations are returned. The selected segmentations are stable, but not all may be unique. In future work we will address the problem of only including highly stable segmentations that are not redundant.

Acknowledgments

This work was funded in part by the Chris and Warren Hellman Foundation, the NSF CAREER #0448615, the Alfred P. Sloan Research Fellowship and was partially supported under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

References

[1] C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, 1995.

[2] K.J. Cho and P. Meer. Image segmentation from consensus information. *CVIU*, 68(1):72–89, October 1997.

[3] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, 2000.

[4] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[5] Bernd Fischer and Joachim M. Buhmann. Bagging for path-based clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1411–1415, November 2003.

[6] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[8] D. Hoiem, A.A. Efros, and M. Hebert. Geometric context from a single image. In *Proc. 10th Int'l. Conf. Computer Vision*, pages I: 654–661, 2005.

[9] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[10] J.J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[11] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 2:83–97, 1955.

[12] M.S. Landy and I. Oruc. Properties of second-order spatial frequency channels. *Vision Research*, 42:2311–2329, 2002.

[13] T. Lange, V. Roth, M.L. Braun, and J.M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16:1299–1323, 2004.

[14] D. D. Lee and H. S. Seung. Learning the parts of objects with non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[15] T. Lindeberg. Principles for automatic scale selection. Technical Report ISRN KTH/NA/P-98/14-SE, Royal Inst. of Tech., 1998.

[16] J. Malik, S. Belongie, J. Shi, and T. Leung. Contour and texture analysis for image segmentation. *IJCV*, 43(1):7–27, 2001.

[17] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, May 2004.

[18] M. Meilă and J. Shi. Learning segmentation with random walk. In *Proc. of NIPS*, pages 873–879, 2001.

[19] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. of NIPS*, 2002.

[20] P. Paatero and U. Tapper. Least squares formulation of robust non-negative factor analysis. *CILS*, 37:23–35, 1997.

[21] S. E. Palmer. *Vision Science*. MIT Press, 1999.

[22] T. Poggio, E. Gamble, and J. Little. Parallel integration of vision modules. *Science*, 242:436–440, 1988.

[23] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[24] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *ICML*, 2005.

[25] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, August 2000.

[26] M. Singh and N. Ahuja. Regression based bandwidth selection for segmentation using parzen windows. In *ICCV*, pages 2–9, 2003.

[27] Z.W. Tu and S.C. Zhu. Image segmentation by data driven markov chain monte carlo. In *ICCV*, pages 131–138, 2001.

[28] M.J. van der Laan, S. Dudoit, and S. Keles. Asymptotic optimality of likelihood-based cross-validation. *Statistical Applications in Genetics and Molecular Biology*, 3(1), 1998.

[29] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proc. 7th Int'l. Conf. Computer Vision*, pages 975–982, 1999.

[30] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recogn. Lett.*, 22(12):1255–1261, 2001.

0-7695-2597-0/06 $20.00 © 2006 IEEE