Available online at www.sciencedirect.com
ScienceDirect
Computer Vision and Image Understanding 110 (2008) 212–225
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

Unsupervised segmentation of natural images via lossy data compression

Allen Y. Yang [a,*], John Wright [b], Yi Ma [c], S. Shankar Sastry [d]

[a] 333 Cory Hall, UC Berkeley, Berkeley, CA 94720, United States
[b] 146 Coordinated Science Laboratory, 1308 W. Main St, Urbana, IL 61801, United States
[c] 145 Coordinated Science Laboratory, 1308 W. Main St., Urbana, IL 61801, United States
[d] 514 Cory Hall, UC Berkeley, Berkeley, CA 94720, United States

Received 8 March 2007; accepted 29 July 2007
Available online 14 September 2007

Abstract

In this paper, we cast natural-image segmentation as a problem of clustering texture features as multivariate mixed data. We model the distribution of the texture features using a mixture of Gaussian distributions. Unlike most existing clustering methods, we allow the mixture components to be degenerate or nearly-degenerate. We contend that this assumption is particularly important for mid-level image segmentation, where degeneracy is typically introduced by using a common feature representation for different textures in an image. We show that such a mixture distribution can be effectively segmented by a simple agglomerative clustering algorithm derived from a lossy data compression approach. Using either 2D texture filter banks or simple fixed-size windows to obtain texture features, the algorithm effectively segments an image by minimizing the overall coding length of the feature vectors. We conduct comprehensive experiments to measure the performance of the algorithm in terms of visual evaluation and a variety of quantitative indices for image segmentation. The algorithm compares favorably against other well-known image-segmentation methods on the Berkeley image database.

Keywords: Image segmentation; Texture segmentation; Lossy compression; Mixture of Gaussian distributions; Clustering

1. Introduction

Natural-image segmentation is one of the classical problems in computer vision. It is widely accepted that a good segmentation should group image pixels into regions whose statistical characteristics (of the color or texture) are homogeneous or stationary, and whose boundaries are "simple" and "spatially accurate" [11]. Nevertheless, from a statistical viewpoint, natural-image segmentation is an *inherently ambiguous* problem for at least the following two technical reasons:[1]

(1) The statistical characteristics of local features (e.g., color, texture, edge, and contour) of natural images usually do not show the same level of homogeneity or saliency at the same spatial or quantization scale. This is not only the case for different natural images, but also often the case for different regions within the same image. Thus, one should not expect the segmentation result to be unique [34], and instead should prefer a hierarchy of segmentations at multiple scales.
(2) Even after accounting for variations due to the scale, different regions or textures may still have different intrinsic complexities, making it a difficult statistical problem to determine the correct number of segments and their model dimensions. For instance, if we use Gaussian distributions to model the features of different textures, the Gaussian for a simple texture obviously has a higher degree of degeneracy (or a lower dimension) than that for a complex texture.

* Corresponding author.
E-mail addresses: yang@eecs.berkeley.edu (A.Y. Yang), jnwright@uiuc.edu (J. Wright), yima@uiuc.edu (Y. Ma), sastry@eecs.berkeley.edu (S.S. Sastry).

[1] It is arguably true that human perception of an image is itself ambiguous. However, here we are only concerned with ambiguities in computing image segmentation.

1077-3142/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2007.07.005

In the literature, many statistical models and methods have been proposed to address some of these difficulties. In this paper, we are interested in *unsupervised* image segmentation. Popular methods in this category include *feature-based* Mean-Shift [1], *graph-based* methods [31,6], *region-based* split-and-merge techniques [26,39], and global optimization approaches based on either energy functions [41] or *minimum description length* (MDL) [13]. Recent developments have mainly focused on the problem of how to integrate textural information at different scales. For example, one can use more sophisticated *region-growing* or *split-and-merge* techniques [11,33,4,9] to partition inhomogeneous regions; or one can use *Markov random fields* to model textures or other image cues [19,26,34]. For a more detailed survey of these methods, the reader is referred to [42,15,8,27].

1.1. Motivations and contributions

Although the reported performance of image-segmentation algorithms has improved significantly over the years, these improvements have come partly at the price of ever more sophisticated feature selection processes, more complex statistical models, and more costly optimization techniques. In this paper, however, we aim to show that for texture features as simple as fixed-size cut-off windows (Fig. 3), with the choice of a likely more relevant class of statistical models (Fig. 1) and its associated agglomerative clustering algorithm (Algorithm 1), one can achieve equally good, if not better, segmentation results as many of the above sophisticated statistical models and optimization methods. Our approach relies on the following two assumptions about how to segment natural images:

(1) The distribution of texture features in a natural image is (approximately) a mixture of Gaussians that may be *degenerate and of different dimensions* (see Fig. 1 right), one for each image segment.
(2) At any given quantization scale, the *optimal* segmentation is the one that gives the most compressed representation of the image features, as measured by the number of binary bits needed to encode all the features.

It is evident that we have chosen to measure the "goodness" of image segmentation in terms of a coding-theoretic criterion: *minimum coding length*.[2] Our earlier work in [16] revealed a strong relationship between lossy data compression and clustering of mixed data. We derived an effective clustering algorithm for mixtures of degenerate or non-degenerate Gaussian distributions. By minimizing the overall coding length of the given mixed data subject to a given distortion, the algorithm automatically merges the data points into a number of Gaussian-like clusters.

Be aware that, although we have adopted the lossy data compression paradigm for image segmentation, we are not suggesting compressing the image *per se*. Instead of pixel values, we compress and segment texture features extracted from the image. Our method bears resemblance to some global optimization approaches, such as using region-merging techniques to minimize a (lossless) MDL cost function [13]. However, the new method significantly differs from existing maximum-likelihood (ML) or MDL-based image segmentation algorithms in the following two main aspects:

(1) We allow the distributions to be *degenerate*, and introduce a new clustering algorithm capable of handling the degeneracy. Extant image-segmentation methods that segment features based on the cluster centers (e.g., K-Means) or density modes (e.g., Mean-Shift) typically work well for low-level segmentation using low-dimensional color features with blob-like distributions (Fig. 1 left) [31]. But for mid-level segmentation using texture features extracted at a larger spatial scale, we normally choose a feature space whose dimension is high enough that the structures of all textures in the image can be *genuinely represented*.[3] Such a representation unavoidably has redundancy for individual textures: The cluster of features associated with one texture typically lies in a low-dimensional submanifold or subspace whose dimension reflects the complexity of the texture (Fig. 1 right). Properly harnessed, such low-dimensional structures can be much more informative for distinguishing textures than the centers of the clusters.
(2) We consider *lossy coding* of the image features, up to an allowable distortion. Varying the distortion provides a simple but effective means of considering textural information at different *quantization* scales.[4] Compressing the image features with different distortions, we naturally obtain a hierarchy of segmenta-

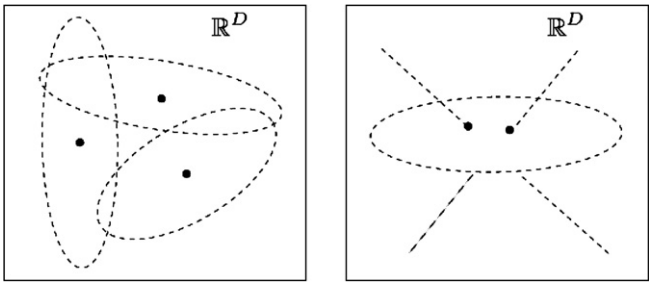

Fig. 1. Mixture of regular (left) or degenerate (right) Gaussians.

[2] It is debatable whether this is how humans segment images. Coding length is an objective measure while human segmentation is highly subjective – much prior knowledge is incorporated in the process. Later we will quantitatively evaluate the extent to which our segmentation results emulate those of humans, in fair comparison with other unsupervised image-segmentation techniques.

[3] Here a genuine representation means that we can recover every texture with sufficient accuracy from the representation.

[4] In this paper, we do not consider varying the spatial scale as we will always choose a fixed-size window as the feature vector. Nevertheless, as we will demonstrate, excellent segmentation can already be obtained.

tions: the smaller the distortion, the more refined the segmentation is (see Fig. 7). In a way, the distortion also plays an important role in image segmentation as a measure of the *saliency* of the segments in an image: First, how small the distortion needs to be in order for certain regions to be segmented from the background, and second, how much we can change the distortion without significantly altering the segmentation (see Fig. 7 again). Thus, lossy compression offers a convenient framework for diagnosing the statistics of a natural image at different quantization scales for various segmentation purposes. This feature is absent in applications of traditional (lossless) MDL to image segmentation.

Through a thorough evaluation, we will demonstrate that lossy coding provides a much more relevant tool for extracting low-dimensional/degenerate structures than traditional (lossless) ML/MDL. The classical likelihood or coding length functions are typically not well-defined for such degenerate structures. While lossy coding, up to an allowable distortion, has been shown to induce a regularization effect that renders the decision rule well-defined, and it also improves finite-sample performance [38]. This ability to seamlessly handle both generic and degenerate distributions renders our approach especially appropriate for segmenting image textures.

1.2. Organization

This paper is organized as follows: Section 2 briefly reviews the coding-based clustering algorithm [16], which minimizes the coding length of data drawn from a mixture of (possibly degenerate) Gaussians. Section 3 introduces the proposed image-segmentation algorithm. Particularly, we discuss how to adaptively select the distortion threshold to achieve good segmentation over a large image database. Section 4 gives experimental results on the Berkeley segmentation database, and compares to other existing algorithms. Finally, Section 5 concludes the paper. We have made all the algorithms public for peer evaluation at: http://www.eecs.berkeley.edu/~yang/software/lossy_segmentation/.

2. Segmentation of mixtures of Gaussians via lossy compression

Once one adopts the assumption that image feature vectors are drawn from a mixture of (possibly degenerate) Gaussians, the problem of image segmentation reduces to that of segmenting such mixed data into multiple Gaussian-like clusters. A popular statistical method for segmenting mixed data is the *expectation-maximization* (EM) algorithm [3,21], which is essentially a greedy descent algorithm to find the maximum-likelihood (ML) estimate of the mixture of Gaussian distributions [7,10,32].

However, notice that here we might be dealing with degenerate Gaussians with unknown dimensions, and furthermore, we do not even know how many of them. Conventional EM-based clustering algorithms do not address these problems, and must be modified to perform well in this domain [16]. In this paper, we adopt a new but simple clustering method introduced in [16], which is especially adept at handling unknown number of possibly degenerate Gaussians. For completeness, in this section, we give a brief overview of this method and the associated clustering algorithm. Readers who are already familiar with [16] may skip this section without loss of continuity.

The new clustering method follows the principle of *lossy minimum description length* (LMDL):[5]

Principle 1 (*Data segmentation via lossy compression*)**.** We define the optimal segmentation to be the one that minimizes the number of bits needed to code the segmented data, subject to a given distortion.

To apply this principle to our problem, we require an accurate measure of the coding length of data drawn from a mixture of Gaussians. We begin by examining the coding length of data from a single Gaussian. Suppose we are given a random vector $v \in \mathbb{R}^D$ with a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, which we wish to encode such that the original vector can be recovered up to a given distortion ε^2, i.e., $\mathbb{E}[\|v - \hat{v}\|^2] \leqslant \varepsilon^2$. From information theory [2], the average number of bits needed to code v is given by the *rate-distortion function* of the Gaussian, which is well approximated as

$$R(\varepsilon) = \frac{1}{2}\log_2 \det\left(I + \frac{D}{\varepsilon^2}\Sigma\right), \tag{1}$$

where I is an identity matrix, and Σ is the covariance.[6]

Now consider a set of N i.i.d. samples $V = (v_1, v_2, \ldots, v_N) \in \mathbb{R}^{D\times N}$ drawn from the Gaussian distribution. Let $\mu \doteq \frac{1}{N}\sum_{i=1}^{N} v_i$, and $\overline{V} \doteq V - \mu \cdot 1_{1\times N}$. As $\widehat{\Sigma} = \frac{1}{N}\overline{V}\overline{V}^{\mathrm{T}}$ is an estimate of Σ, an estimate of the rate-distortion function $R(\varepsilon)$ is

$$R(\varepsilon, V) \doteq \frac{1}{2}\log_2 \det\left(I + \frac{D}{\varepsilon^2 N}\overline{V}\overline{V}^{\mathrm{T}}\right). \tag{2}$$

Encoding the N vectors in V therefore requires $N \cdot R(V)$ bits. Since the codebook is adaptive to the data V, we must also represent it with $D \cdot R(V)$ bits, which can be viewed as the cost of coding the D principal axes of the data covariance $\frac{1}{N}\overline{V}\overline{V}^{\mathrm{T}}$. As the data are in general not zero-mean, we need additional $\frac{D}{2}\log_2(1 + \frac{\mu^{\mathrm{T}}\mu}{\varepsilon^2})$ bits to encode the mean vector μ. This leads to the following estimate of the total number of bits needed to encode the data set V:

[5] For a theoretical characterization and comparison of (lossy) ML estimate and (lossy) MDL estimate, one may refer to [17].

[6] Strictly speaking, the rate-distortion function for the Gaussian source $\mathcal{N}(\mu, \Sigma)$ is $R(\varepsilon) = \frac{1}{2}\log_2 \det(\frac{D}{\varepsilon^2}\Sigma)$ when $\frac{\varepsilon^2}{D}$ is smaller than the smallest eigenvalue of Σ. However, when $\frac{\varepsilon^2}{D}$ is larger than some eigenvalues of Σ, the rate-distortion function becomes more complicated [2]. Nevertheless, the approximate formula $R(\varepsilon) = \frac{1}{2}\log_2 \det(I + \frac{D}{\varepsilon^2}\Sigma)$ can be viewed as the rate distortion of the "regularized" source that works for all range of ε.

$$L(V) \doteq \frac{N+D}{2}\log_2 \det\left(I + \frac{D}{\varepsilon^2 N}\overline{V}\overline{V}^{\mathrm{T}}\right) + \frac{D}{2}\log_2\left(1 + \frac{\mu^{\mathrm{T}}\mu}{\varepsilon^2}\right).$$

Furthermore, although the above formula is derived for a Gaussian source, the same formula gives an *upper bound* of the coding length for any finite number of samples drawn from a subspace, i.e., a degenerate Gaussian. A detailed proof is provided in [16].

Now let us consider the given data set V as drawn from a mixture of Gaussians. In this case, (3) no longer gives an accurate estimate of the minimum coding length for V. It may be more efficient to code V as the union of multiple disjoint subsets: $V = W_1 \cup W_2 \cup \cdots \cup W_K$. If each subset is sufficiently Gaussian, the total number of bits needed to code V is at most

$$L^s(W_1, \ldots, W_K) \doteq \sum_{i=1}^{K} \{L(W_i) - |W_i|\log_2(|W_i|/N)\}. \quad (4)$$

Here the second term counts the number of bits needed to code (losslessly) the membership of the N samples in the K groups, e.g., using the Huffman coding [2]. Notice that the Huffman coding of the membership is optimal only when the membership of the vectors in the K segments is totally random. However, in image segmentation, the membership of pixels is not random – adjacent pixels have higher probability of being in the same segment. In this case, Huffman coding only gives a loose upper bound. Nevertheless, we will demonstrate that minimizing such a function leads to a very simple and effective segmentation algorithm.

To find the optimal segmentation, one essentially needs to compute the coding length for all possible segmentations of the data V, which is combinatorially expensive. To make the optimization tractable, we make use of a *pairwise steepest descent* procedure to minimize the coding length: In the initialization step, each vector v_i is assigned as its own group. At each iteration a pair of groups S_i and S_j is merged such that the decrease in the coding length due to coding S_i and S_j together is maximal. The algorithm terminates when the coding length can no longer be reduced by merging any pair of groups.

Algorithm 1 (Pairwise Steepest Descent)

1: **input**: the data $V = (v_1, v_2, \ldots, v_N) \in \mathbb{R}^{D\times N}$ and a distortion ε^2.
2: initialize $\mathcal{S} := \{S_i = \{v_i\} | i = 1, \ldots, N\}$.
3: **while** $|\mathcal{S}| > 1$ **do**
4: choose distinct groups $S_1, S_2 \in \mathcal{S}$ such that $L^s(S_1 \cup S_2) - L^s(S_1, S_2)$ is minimal.
5: **if** $L^s(S_1 \cup S_2) - L^s(S_1, S_2) \geqslant 0$ **then** break;
6: **else** $\mathcal{S} := (\mathcal{S} \setminus \{S_1, S_2\}) \cup \{S_1 \cup S_2\}$.
7: **end**
8: **output**: $\mathcal{S}$

Notice that the greedy merging process in Algorithm 1 is similar in concept to classical agglomerative clustering methods, especially Ward's method [37,12]. However, by using the coding length as a new distance measure between groups, Algorithm 1 significantly improves these classical methods particularly when the distributions are degenerate or the data contain outliers. Nevertheless, as a greedy descent scheme, the algorithm does not guarantee to always find the globally optimal segmentation for any given (V, ε^2).[7] In our experience, the main factor affecting the global convergence of the algorithm appears to be the density of the samples relative to the distortion ε^2.

Extensive simulations have verified that this algorithm is consistently effective in segmenting data that are drawn from a mixture of Gaussian or degenerate subspace distributions. In addition, the algorithm tolerates significant amounts of outliers, and requires no prior knowledge of the number of groups nor their dimensions. In case that the data structures are nonlinear manifolds, the algorithm further provides an effective way of fitting nonlinear structures with mixture Gaussian/subspace models. Fig. 2 shows a few segmentation results of this algorithm on synthesized data sets. For more detailed analysis of Algorithm 1, the reader is referred to [16].

In the above experiment, the distortion parameter ε^2 was selected to be close to the true noise variance to achieve best results. In practice, there is no universal rule for choosing a good ε for all practical data sets. To apply Algorithm 1 to image segmentation, we need to be able to adaptively choose ε for each image based on its unique texture distribution. We will carefully examine this issue in the next section.

3. Image segmentation via lossy compression

In this section, we describe how the lossy compression-based method in Section 2 is applied to segmenting natural images. We first discuss what features we use to represent textures and why. We then describe how a *low-level segmentation* is applied to partition an image into many small homogeneous patches, known as *superpixels*. The superpixels are used to initialize the *mid-level texture-based segmentation*, which minimizes the total coding length of all the texture features by repeatedly merging adjacent segments, subject to a distortion ε^2. Finally, we study several simple heuristics for choosing a good ε for each image.

3.1. Constructing feature vectors

We choose to represent a 3-channel RGB color image in terms of the $L^*a^*b^*$ color metric, which was specially designed to best approximate perceptually uniform color spaces.[8] While the dependence of the three coordinates

[7] It may be possible to improve the convergence by using more complicated split-and-merge strategies [35].
[8] Equivalently, one can also use the $L^*u^*v^*$ metric.

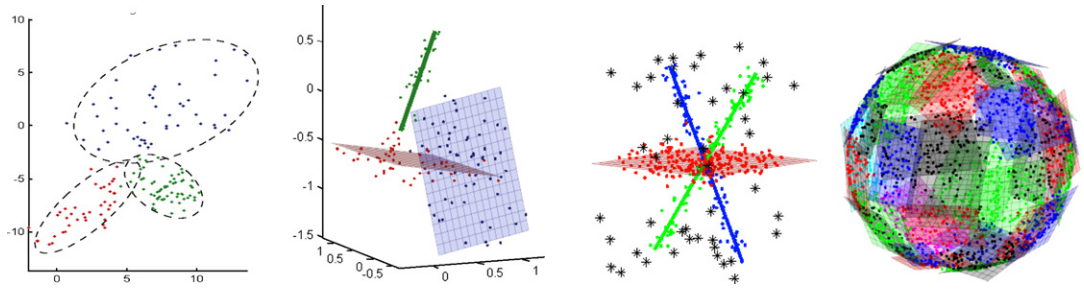

Fig. 2. Simulation results (in color) of Algorithm 1 on four different mixture distributions. Left: Three Gaussian distributions in $\mathbb{R}^2$. Middle left: Three affine subspaces of dimensions $(2,2,1)$ in $\mathbb{R}^3$. Middle right: Three linear subspaces of dimensions $(2,1,1)$ in $\mathbb{R}^3$ with 12% outliers; the algorithm groups all the outliers into one extra Gaussian cluster, in addition to the three subspaces. Right: Approximation of a nonlinear sphere manifold using multiple subspace models; each subspace model locally fits the nonlinear data up to the distortion ε^2. (For interpretation of the references in color in this figure legend, the reader is referred to the web version of this article.)

on the traditional RGB metric is nonlinear [1], the $L^*a^*b^*$ metric better facilitates representing texture via mixtures of Gaussians. Perceptual uniformity renders the allowable distortion ε^2 meaningful in terms of human perception of color differences, tightening the link between lossy coding and our intuitive notion of image segmentation.

In the literature, there have been two major types of features used to capture local textures. The first type considers responses of a 2D filter bank as texture features [18,40]. The second directly uses a $w \times w$ cut-off window around each pixel and stacks the color values inside the window into a vector [25,24]. Each texture window is usually smoothed by convolving with a 2D Gaussian kernel before stacking. Fig. 3 illustrates this process.

Although texture features were traditionally extracted through large-scale 2D filter banks, more recent study has suggested that the texture features from simple cut-off windows may give better performance in terms of image segmentation [36]. We have experimented with using both simple window features and two classical filter banks, namely, the Leung-Malik set [18] and the Schmid set [30], in conjunction with our clustering algorithm. We found the difference in the segmentation result is small despite the fact that filter-bank features are more computationally expensive as they involve convolutions of the image with large number of filters. One likely reason for the similar performance is that the compression-base clustering algorithm is capable of automatically harnessing the low-dimensional linear structures of the features, despite noise and outliers (see Fig. 2 and additional evidence in [16]).

For simplicity, we choose to use the window features in this paper. We find that a 7×7 window provides satisfactory results, although other similar sizes also work well.[9] Finally, to reduce the computational cost, we project the feature vectors into an eight-dimensional space by PCA. This operation preserves all linear structures of dimension

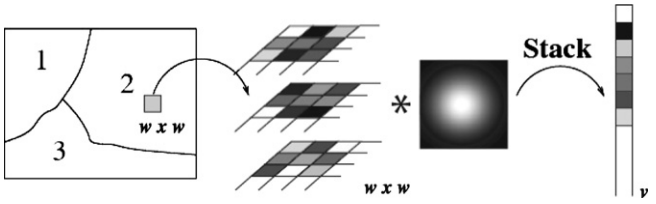

Fig. 3. The construction of texture features: A $w \times w$ window of each of the three $L^*a^*b^*$ channels is convoluted with a Gaussian and then all channels are stacked into a single vector v.

less than 8 in the feature space. Experimentally, we found an eight-dimensional space to be sufficient for most textures from natural images.

3.2. Initialization with superpixels

Given the feature vectors extracted from an image, one "naive" approach would be to directly apply Algorithm 1, and segment the pixels based on the grouping of the feature vectors. Fig. 4 shows one such result. Notice that the resulting segmentation merges pixels near the strong edges into a single segment. This should be expected from the compression perspective, since windows across the boundary of two segments have significantly different structures from the (homogeneous) textures within those segments [13]. However, such a segmentation does not agree well with human perception.

In order to properly group edge pixels, appropriately, we preprocess an image with a low-level segmentation based on local cues such as color and edges. That is, we oversegment the image into (usually several hundred) small, homogeneous regions, known as *superpixels*. This preprocessing step has been generally recommended for all region merging algorithms in [13]. Such low-level segmentation can be effectively computed using K-Means or Normalized-Cuts (NCuts) [31] with a conservative homogeneity threshold. In this paper, we use a publicly available superpixel code [23].

Since the superpixel segmentation respects strong edges in an image (see Fig. 5 middle), it does not suffer from the

[9] We did not test window sizes larger than 9 pixels, as the current MATLAB implementation cannot store all such texture vectors from a typical 320×240 color image. However, this problem can be alleviated by sampling a subset of the texture features from an image.

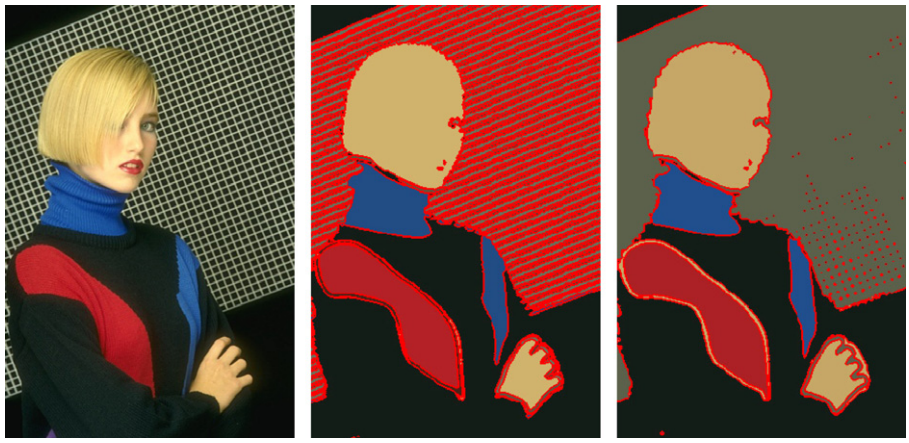

Fig. 4. Two segmentation results of the left original using Algorithm 1 with different ε's. Notice that the pixels near the boundaries of segments are not grouped correctly.

Fig. 5. The segmentation pipeline. Left: Original. Middle: Superpixels obtained from low-level oversegmentation. Right: Segments obtained by minimizing the coding length with $\varepsilon = 0.2$.

misassignment of edge pixels seen in Fig. 4. All feature vectors associated with pixels in each superpixel are initialized as one segment, forcing the subsequent merging process to group boundary pixels together with the interior pixels. An additional benefit from the superpixel preprocessing is a significant reduction in the computational cost. Using superpixel segments as initial grouping, the algorithm only needs to search amongst several hundred of superpixels for the optimal pair to merge, instead of searching amongst all feature vectors (the number of vectors is on the order of tens of thousands).

One may further consider sampling only a portion of the feature vectors associated with each superpixel. For instance, feature vectors at the boundary of a superpixel represent a combination of textures from two adjacent superpixels, and their distribution can be rather complicated compared to the distribution of the feature vectors in the interior of the superpixel. Thus, one may use only feature vectors from the interior of each superpixel.[10] Our experiments show that, under the same distortion parameter ε, this modification tends to partition an image into smaller texture segments. This phenomenon will be discussed in more detail in Section 4.3. For clarity, all segmentation results presented in this paper will use both interior and boundary feature vectors of every superpixel unless stated otherwise.

3.3. Enforcing connected segments

Notice that in the definition of the overall coding length function (4), we use the Huffman coding length to upper bound the number of bits required to encode the membership of the feature vectors. This obviously overestimates the coding length since it does not take into account the fact that in natural images, adjacent pixels have higher probability of belonging to the same segment.

In order to enforce that the resulting segmentation respects spatial continuity and consists of only connected segments, we impose an additional constraint that two segments S_i and S_j can be merged together only if they are spatially adjacent in the 2D image. To this end, we need to

[10] If a superpixel only consists of boundary pixels, these pixels are used anyway.

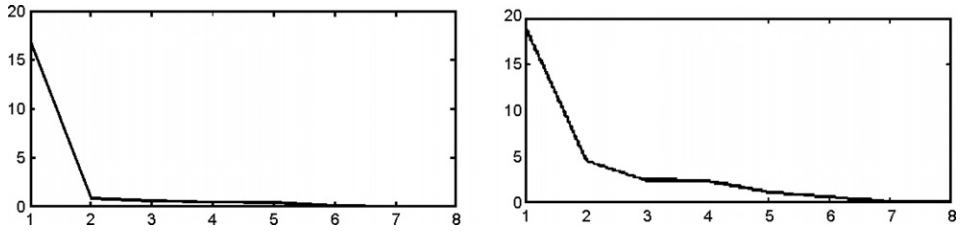

Fig. 6. Singular values of the feature vectors drawn respectively from two image segments in Fig. 5 right. The segment plotted on the left is on the woman's clothes, and the other is from the background.

construct and maintain a *region adjacency graph* (RAG) G in the clustering process. RAG is popularly used in other *merge-and-split* type segmentation methods [15]. We represent an RAG using an adjacency list $G\{i\}$ for each segment S_i. Index j is in the set $G\{i\}$ if the segment S_j is a neighbor of S_i. At each iteration, the algorithm searches for a pair of adjacent segments S_i and S_j which leads to maximal decrease in the total coding length. Note, however, that in some applications such as image compression, disconnected regions may be allowed to be grouped as the same segment. In this case, one can simply discard the adjacency constraint in our implementation.

Fig. 5 shows an example of the two-step segmentation process. For this image, we find that all feature vectors approximately lie in a 6D subspace in the 8D feature space (i.e., the first 8 principal components of the Gaussian windows). Furthermore, feature vectors of each segment can be well modeled as a 1D to 4D subspace. Fig. 6 plots the singular values of two representative segments. This validates our initial assumption that the distributions of texture features are typically (close to) degenerate.

3.4. Choosing the distortion

As discussed in the introduction, the distortion ε effectively sets the quantization scale at which we segment an image. Fig. 7 shows the segmentation of several images under different values of ε. As the figure suggests, a single ε will not give good performance across a widely varying data set such as the Berkeley image-segmentation database. Differences in the contrast of the foreground and background, lighting conditions, and image category cause the distribution of the texture features to vary significantly from image to image.

There are several ways to adaptively choose ε to achieve good segmentation for each image. For example, if a desired number of segments is known *a priori*, we can search a range of ε values for the one that gives the desired number of segments. When such information is not available *a priori*, as is the case for image segmentation, a formal way in information theory to estimate the distortion parameter is to minimize a cost function such as the following one:

$$\varepsilon^* \doteq \min_{\varepsilon \in \mathscr{E}} \{L^s(V,\varepsilon) + \lambda ND \log_2(\varepsilon)\}, \tag{5}$$

where λ is a parameter provided by the user that weighs the two terms $L^s(V,\varepsilon)$ and $ND\log_2(\varepsilon)$. Notice that the first term $L^s(V,\varepsilon)$ decreases as ε increases, as opposed to the second term $ND\log_2(\varepsilon)$. Hence, the expression essentially seeks a balance between the coding length of the data and the complexity of the model measured as $ND\log_2(\varepsilon)$. It is studied in [16] that (5) can accurately recover the true value of ε for the simulated Gaussian mixture models by simply setting $\lambda = 1$. However, when applied to image segmentation on real natural images, the so-estimated ε^* tends to oversegment the images. One reason for this discrepancy between simulation and experiment is that the noise associated with different texture segments can have different covariance.

In this work, we choose to adaptively select the distortion ε by stipulating that feature distributions in adjacent texture regions must be sufficiently dissimilar. In the literature, the similarity measure between two texture distributions has been extensively studied. In information theory, the *Kullback–Leibler* (KL) divergence measures the relative entropy between two arbitrary distribution functions $p(x)$ and $q(x)$ [2]:

$$d_{\mathrm{KL}} = \sum_{x \in \mathscr{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{6}$$

However, the KL divergence is ill-posed for distributions functions $p(x)$ and $q(x)$ that have different supports, where $q(x)$ may be equal to zero as the denominator in the log function. Unfortunately, this is often the case to compare two degenerate distributions (e.g., texture vectors from images).

In computer vision, the heuristic Earth Mover's Distance (EMD) is a metric to measure the similarity of two image distributions [29,28]. Levina and Bickel [14] further show that EMD is equivalent to the Mallows distance when applied to probability distributions.[11] In this paper, since texture segments are modeled by Gaussian distributions, their EMD/Mallows similarity distance has a closed-form expression [5]:

$$d_{\mathrm{M}}(N(\theta_1,\Sigma_1),N(\theta_2,\Sigma_2))^2 = (\theta_1-\theta_2)^{\mathrm{T}}(\theta_1-\theta_2) + Tr(\Sigma_1+\Sigma_2-2(\Sigma_1\Sigma_2)^{\frac{1}{2}}), \tag{7}$$

[11] In general, EMD is equivalent to the Mallows distance when the sums of the probabilities in two clusters are normalized to be equal. In case that the probabilities in two clusters are not normalized, EMD and the Mallows distance behave differently [14].

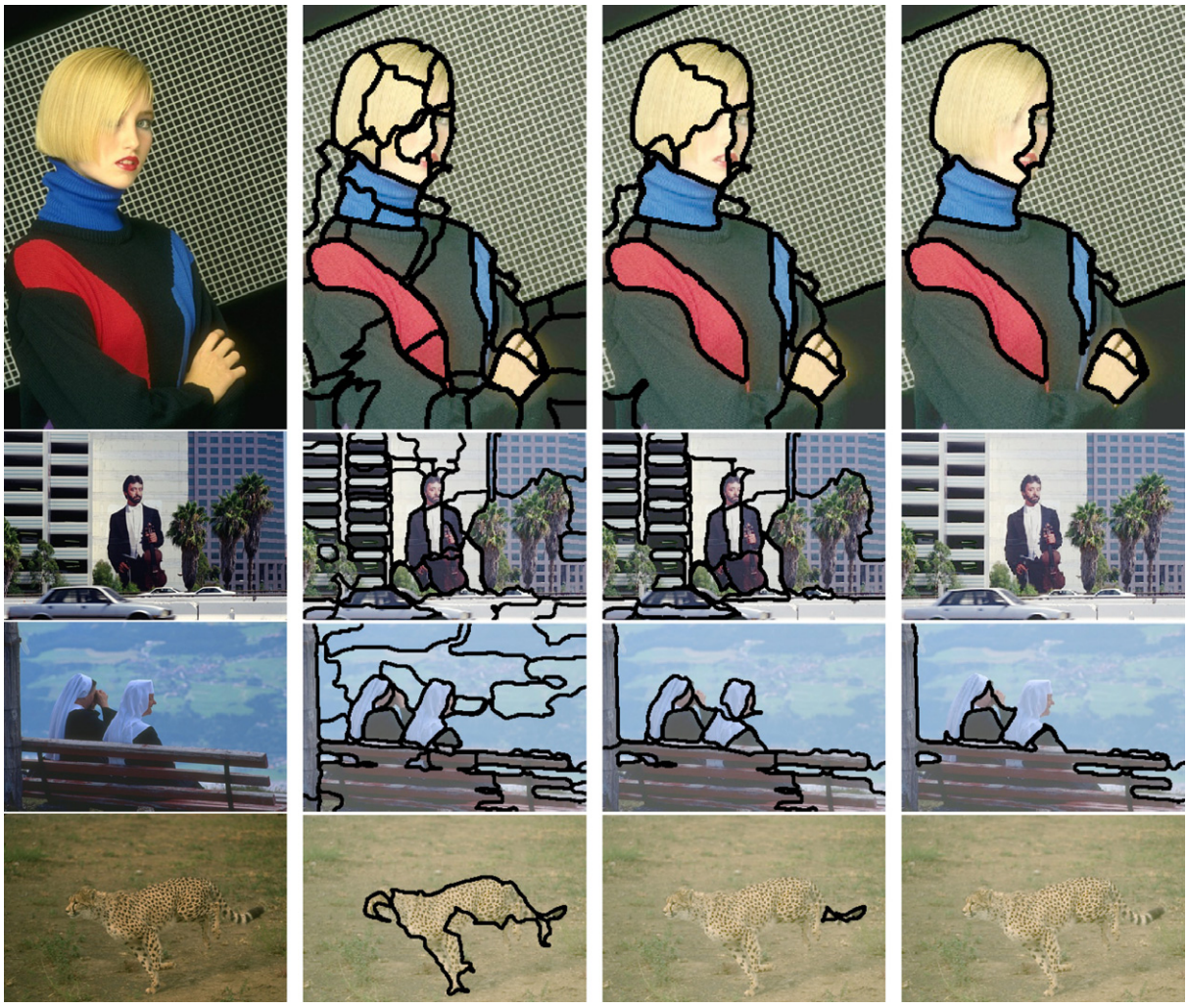

Fig. 7. Segmentation results under different ε's. Left: Originals. Middle left: $\varepsilon = 0.1$. Middle right: $\varepsilon = 0.2$. Right: $\varepsilon = 0.4$.

where the two texture regions are parameterized by Gaussian distributions $N(\theta_1, \Sigma_1)$ and $N(\theta_2, \Sigma_2)$.

Finally, as a reasonable approximation to the Mallows distance, one can measure the similarity of $N(\theta_1, \Sigma_1)$ and $N(\theta_2, \Sigma_2)$ using their mean vectors:

$$d_{\mathrm{m}}(N(\theta_1, \Sigma_1), N(\theta_2, \Sigma_2))^2 = (\theta_1 - \theta_2)^{\mathrm{T}}(\theta_1 - \theta_2). \tag{8}$$

For a given ε and a fixed distance measure that can be either the Mallows distance d_{M} or the mean distance d_{m}, the minimal distance $d(\varepsilon)$ of an image is calculated between all pairs of adjacent segments after the compression-based merging. The selection process gradually increases the value of ε from a list $\mathscr{E}$ of candidate values until the minimal distance $d(\varepsilon)$ is larger than a preselected threshold γ:

$$\varepsilon^* = \min\{\varepsilon : d(\varepsilon) \geqslant \gamma\}. \tag{9}$$

The final segmentation result then gives the most refined segmentation which satisfies the above constraint. We note that increasing ε typically causes the number of segments to decrease and results in a shorter coding length. We may therefore use the segmentation computed with a smaller ε to initialize the merging process with a larger ε, allowing us to search for the optimal ε more efficiently.

It may seem that we have merely replaced one free parameter, ε, with another, γ. This replacement has two strong advantages, however. Experimentally we find that even with a single fixed value of γ the algorithm can effectively adapt to all image categories in the Berkeley database, and achieve segmentation results that are consistent with human perception. Furthermore, the appropriate γ can be estimated empirically from human segmentations, whereas ε cannot. This heuristic thresholding method is similar in spirit to several robust techniques in computer vision for estimating mixture models, e.g., the Hough transform and RANSAC.

The complete segmentation process is specified as Algorithm 2. In terms of speed, on a typical 3 GHz Intel PC, the MATLAB implementation of the CTM algorithm on a 320×240 color image takes about two minutes to preprocess superpixels, and less than one minute to search for the optimal ε^* and minimize the coding length of the features.

(a) Original images.

(b) Segmentation results with $CTM_{+,\gamma=0.1}$.

(c) Segmentation results with $CTM_{-,\gamma=0.1}$.

Fig. 8. Segmentation results on certain animal images. CTM_{+} represents the CTM algorithm applied to all texture vectors including those at the boundaries. CTM_{-} represents the same algorithm without sampling the texture vectors at the boundaries.

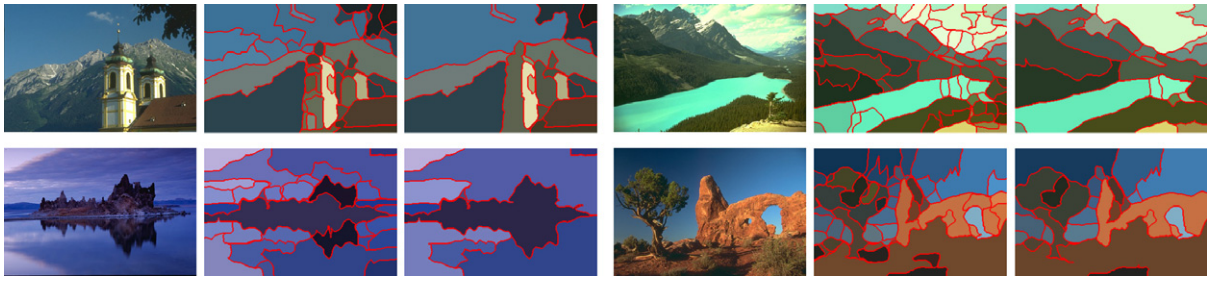

Fig. 9. Examples in category landscape. Left: Original. Middle: $CTM_{\gamma=0.1}$. Right: $CTM_{\gamma=0.2}$.

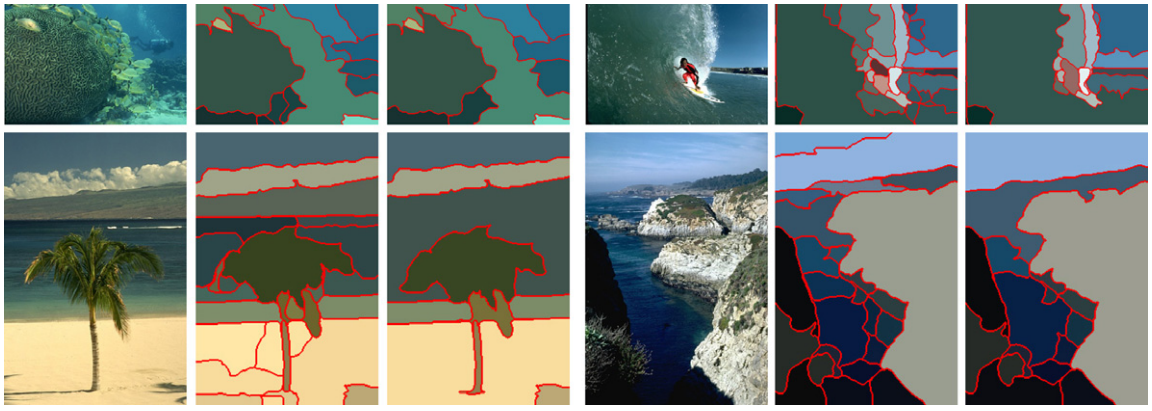

Fig. 10. Examples in category water. Left: Original. Middle: $CTM_{\gamma=0.1}$. Right: $CTM_{\gamma=0.2}$.

Fig. 11. Examples in category urban. Left: Original. Middle: $\mathrm{CTM}_{\gamma=0.1}$. Right: $\mathrm{CTM}_{\gamma=0.2}$.

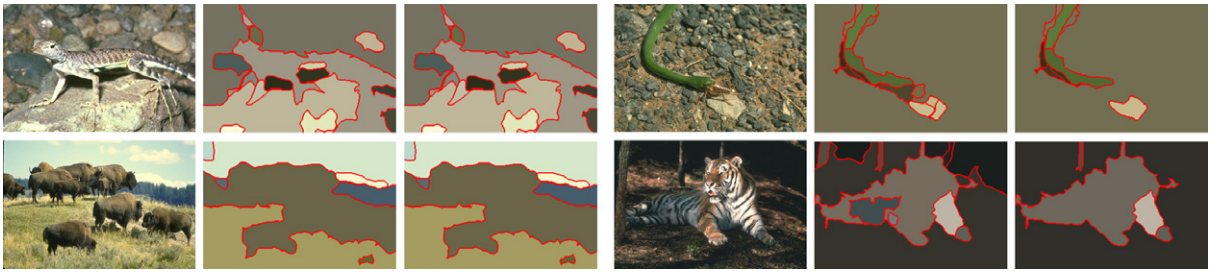

Fig. 12. Examples in category animals. Left: Original. Middle: $\mathrm{CTM}_{\gamma=0.1}$. Right: $\mathrm{CTM}_{\gamma=0.2}$.

4. Experiments

In this section, we demonstrate the segmentation results of Algorithm 2 (CTM) on natural images in the Berkeley segmentation database [20], which also contains benchmark segmentation results obtained from human subjects.

4.1. Visual verification

We first *visually* verify the segmentation results on the Berkeley database. Representative segmentation results of the CTM algorithm with $\gamma = 0.1$ and $\gamma = 0.2$ are shown in Figs. 9–14. The mean distance d_{m} defined in (8) is used to measure the texture similarity between adjacent segments.[12] For better visual evaluation, we have partitioned the database into six different image categories, each of which consists of images that are more relevant, namely, *Landscape* (Fig. 9), *Water* (Fig. 10), *Urban* (Fig. 11), *Animals* (Fig. 12), *People* (Fig. 13), and *Objects* (Fig. 14). By comparing the segmentation results with the two γ values, we conclude that smaller γ's tend to generate more segments and oversegment the images, and larger γ's tend to generate less segments and hence undersegment the images.

[12] The segmentation using the Mallows distance d_{M} is slightly different. However, using the quantitative segmentation measures in Section 4.2, the segmentations using the two distances are very close in terms of the performance.

Algorithm 2 (CTM: Compression-based Texture Merging)

input: Image $I \in \mathbb{R}^{H\times W\times 3}$ in $L^*a^*b^*$ metric, reduced dimension D, window size w, distortion range $\mathscr{E}$, and minimum mean distance γ.

Partition I into superpixels $S_1,\ldots,S_K$. For pixel $p_i \in S_j$, initialize its label $l_i = j$.
Construct RAG $G\{1\},\ldots,G\{K\}$ for the K segments $S_1,\ldots,S_K$.
Sample $w \times w$ windows, and stack the resulting values into a feature vector $v_i \in \mathbb{R}^{3w^2}$.
Replace v_i with their first D principal components.
for all $\varepsilon \in \mathscr{E}$ in ascending order **do**
  for all initial segments S_i, $i = 1,\ldots,K$ **do**
    Compute $L^s(S_i,\varepsilon)$.
    for all $j \in G\{i\}$ **do**
      $U_{ij} \doteq L^s(S_i,\varepsilon) + L^s(S_j,\varepsilon) - L^s(S_i \cup S_j,\varepsilon)$
    end for
  end for
  while $U_{ij} \doteq \max\{U\} > 0$ **do**
    Merge S_i and S_j. Update arrays l, G, L, and U.
    Segment number $K \leftarrow K - 1$.
  end while
  if $\gamma \leqslant \min_{i,j\in G(i)}\{d(S_i,S_j,\varepsilon)\}$ **then**
    break.
  end if
end for

output: Final pixel labels $l_1,\ldots,l_{H\times W}$.

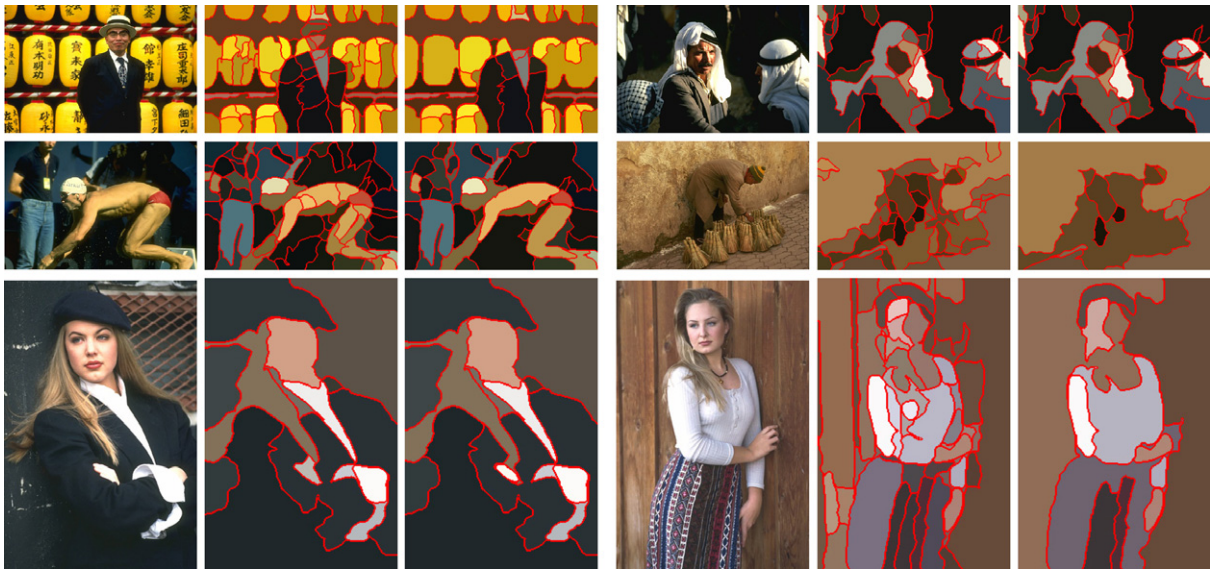

Fig. 13. Examples in category people. Left: Original. Middle: $CTM_{\gamma=0.1}$. Right: $CTM_{\gamma=0.2}$.

Fig. 14. Examples in category objects. Left: Original. Middle: $CTM_{\gamma=0.1}$. Right: $CTM_{\gamma=0.2}$.

4.2. Quantitative verification

We now *quantitatively* compare CTM against three unsupervised algorithms that have been made available publicly: Mean-Shift [1], NCuts [31], and Felzenszwalb and Huttenlocker (FH) [6]. The comparison is based on four quantitative performance measures:

(1) The Probabilistic Rand Index (PRI) [27] counts the fraction of pairs of pixels whose labels are consistent between the computed segmentation and the ground truth, averaging across multiple ground truth segmentations to account for scale variation in human perception.
(2) The Variation of Information (VoI) metric [22] defines the distance between two segmentations as the average conditional entropy of one segmentation given the other, and thus roughly measures the amount of randomness in one segmentation which cannot be explained by the other.
(3) The Global Consistency Error (GCE) [20] measures the extent to which one segmentation can be viewed as a refinement of the other. Segmentations which are related in this manner are considered to be consistent, since they could represent the same natural image segmented at different scales.
(4) The Boundary Displacement Error (BDE) [8] measures the average displacement error of boundary pixels between two segmented images. Particularly, it defines the error of one boundary pixel as the distance between the pixel and the closest pixel in the other boundary image.

Since all methods are unsupervised, we use both the training and testing images for the evaluation. Due to memory issues with the NCuts implementation in MATLAB, all images are normalized to have the longest side equal to 320 pixels. We ran Mean-Shift [1] with parameter settings (h_s, h_r) chosen at regular intervals of $[7,16]\times[3,23]$, and found that on the Berkeley database, $(h_s, h_r) = (13, 19)$

Table 1
Average performance on the Berkeley Database (bold indicates best of all the algorithms)

	PRI	VoI	GCE	BDE
Humans	0.8754	1.1040	0.0797	4.994
$CTM_{\gamma=0.1}$	0.7561	2.4640	**0.1767**	**9.4211**
$CTM_{\gamma=0.15}$	0.7627	2.2035	0.1846	9.4902
$CTM_{\gamma=0.2}$	0.7617	**2.0236**	0.1877	9.8962
Mean-Shift	0.7550	2.477	0.2598	9.7001
NCuts	0.7229	2.9329	0.2182	9.6038
FH	**0.7841**	2.6647	0.1895	9.9497

PRI ranges between $[0,1]$, higher is better. VoI ranges between $[0,\infty)$, lower is better. GCE ranges between $[0,1]$, lower is better. BDE ranges between $[0,\infty)$ in the unit of pixel, lower is better.

gives a good overall tradeoff between the above quantitative measures. We therefore use this parameter choice for our comparison. For NCuts [31], we choose the number of segments $K = 20$ to agree with the average number of segments from the human subjects. For the FH algorithm, we choose the Gaussian smoothing parameter $\Sigma = 0.5$, the threshold value $k = 500$, and the minimal region size to be 200 pixels, as suggested by the authors [6].

Table 1 gives the quantitative comparison of CTM against the other three algorithms on the Berkeley segmentation benchmark. In the experiment, three γ values are tested for CTM, namely, $\gamma = 0.1, 0.15, 0.2$. The texture distance is the mean distance d_{m}. The distortion range $\mathscr{E}$ for the ε value is between 0.01 and 0.5, which are relative scales in terms of the normalized texture vectors.

Table 1 shows that quantitatively, CTM outperforms Mean-Shift, NCuts, and FH in terms of most indices: At $\gamma = 0.15$, CTM is better than Mean-Shift and NCuts in terms of all four indices; and for all chosen γ's, CTM is better than FH except for the PRI index. It is perhaps not surprising that CTM significantly outperforms the other three algorithms in terms of the VoI index, since we are optimizing an information-theoretic criterion. Comparing with the indices of the results by humans, these numbers show that minimizing the coding length leads to segmentation that is closer to human segmentation. It suggests that perhaps human perception also approximately minimizes some measure of the compactness of the representation.

One may also interpret the results in terms of the differences among the four segmentation indices. The GCE and BDE indices penalize undersegmentation more heavily than oversegmentation. In particular, GCE does not penalize oversegmentation at all, i.e., the highest score is achieved by assigning each pixel as an individual segment. As a result, $CTM_{\gamma=0.1}$ has returned the best GCE and BDE values among all the results in Table 1, but its VoI value is one of the worst in the table. From our experience (also shown in Figs. 9–14), PRI and VoI seem to be more correlated with human segmentation in term of visual perception.

To summarize both visual and quantitative comparisons, we notice that on one hand, if we tune the algorithms to give the visually best match with human segmentation, none of the algorithms with different parameters is a clear winner in terms of all four indices; on the other hand, none of the indices seems to be a better indicator of human segmentation than others, which suggests that human segmentation uses much more comprehensive cues. Nevertheless, the extensive visual demonstration and quantitative comparison does serve to validate our hypotheses that the distribution of texture features of natural images can be well approximated by a mixture of (possibly degenerate) Gaussians. As a result, the compression-based clustering algorithm as a powerful tool exploits the redundancy and degeneracy of the distributions for good texture segmentation.

4.3. Difficulties and possible extensions

To fairly assess an image-segmentation algorithm, we also need to investigate examples for which the algorithm has failed to produce good results. In this subsection, we will show a handful of such examples from the Berkeley database, and discuss several possible extensions to the CTM algorithm to further improve the segmentation results.

A particular category that CTM has trouble with is a set of images of animals with very severe camouflage. Fig. 8 shows some representative examples. For these examples, it is difficult for CTM to segment an animal from the background even with very small distortion ε. Comparing with Fig. 7, human figures often endure a larger ε, as human complexion and clothes stand out from the (man-made) surroundings. Thus, in a way, the distortion ε can be interpreted as a measure for how "salient" an object is in an image and how much "attention" is needed to segment the object.

In order to extract severely camouflaged animals from their surroundings, a straightforward extension of the CTM algorithm is to exclude texture vectors at the boundaries of the superpixels. A texture vector, say the Gaussian window, at the boundary contains pixels from the two adjacent superpixels that share the common boundary. By excluding these texture vectors, the set of texture vectors from the superpixel become more homogeneous. Hence, the compression-based algorithm can more effectively distinguish the texture of the animal from that of the background. This variation of CTM is denoted as CTM_{-} while the original version is denoted as CTM_{+}. Fig. 8 demonstrates the improvement of the CTM_{-} algorithm on these images. But notice that it still failed to segment out the body of the crocodile from the background; in this case the camouflage is effective enough to fool even human eyes.

We also observe another limitation of CTM from the results in Fig. 8. As an example, for the Leopard image, the algorithm needs to use a relatively small ε to extract the image segment of the leopard from the background. Nevertheless, under the same ε, the background textures are oversegmented. At a fixed γ, the CTM algorithm searches for the best distortion parameter ε value to code

the feature vectors of the entire image, despite the fact that these textures may have different noise variances (e.g., foreground versus background).

A possible solution to this problem is to assign different ε values to different image regions in a *supervised* scenario. Given a set of training images that are segmented by a human subject, one can learn the distribution of ε of all the textures. Then, given a new image, one needs to infer the appropriate ε to use for different regions in a Bayesian fashion.

Such an extension may give more relevant segmentation results for several important applications, such as salient object detection. For instance, saliency is arguably a subjective notion, as people have their own preference of which region in an image is the most salient one. We have shown through extensive experiments in this paper that whether an image region can be segmented from its surroundings is closely related to the distortion allowed in the lossy coding. Therefore, it is possible to learn a compression-based saliency detector through a set of examples. The segmentation results will most likely resemble the results of the individual human subject who has provided the training examples.

5. Discussion and conclusion

In this paper, we have proposed that texture features of a natural image should be modeled as a mixture of possibly degenerate distributions. We have introduced a lossy compression-based clustering algorithm, which is particularly effective for segmenting degenerate Gaussian distributions. We have shown that the algorithm can be customized to successfully segment natural images by harnessing the natural low-dimensional structures that are present in raw texture features such as Gaussian windows.

In addition, the lossy compression-based approach allows us to introduce the distortion as a useful parameter so that we can obtain a hierarchy of segmentations of an image at multiple quantization scales. We have proposed a simple heuristic criterion to adaptively determine the distortion for each image if one wants to match the segmentation with that of humans.

In this paper, we have studied only unsupervised segmentation of natural images. However, the proposed framework can also be extended to supervised scenarios. We believe that it is of great importance to better understand how humans segment natural images from the lossy data compression perspective. Such an understanding would lead to new insights into a wide range of important problems in computer vision such as salient object detection and segmentation, perceptual organization, and image understanding and annotation. These are some of the challenging problems left open for future investigation.

Acknowledgments

This work is partially supported by NSF CAREER IIS-0347456, NSF CRS-EHS-0509151, NSF CCF-TF-0514955, ONR YIP N00014-05-1-0633, and ARO MURI W911NF-06-1-0076. The authors would like to thank Parvez Ahammad for his suggestions and literature references.

References

[1] D. Comanicu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 603–619.
[2] T. Cover, J. Thomas, Elements of Information Theory, Wiley Series in Telecommunications, 1991.
[3] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society 39 (B) (1977) 1–38.
[4] Y. Deng, B. Manjunath, H. Shin, Color image segmentation, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, vol. 2, 1999, pp. 23–25.
[5] D. Dowson, B. Landau, The Frechet distance between multivariate normal distributions, Journal Multivariate Analysis 12 (3) (1982) 450–455.
[6] P. Felzenszwalb, D. Huttenlocher, Efficient graph-based image segmentation, International Journal on Computer Vision 59 (2) (2004) 167–181.
[7] M. Figueiredo, A. Jain, Unsupervised learning of finite mixture models, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 1–16.
[8] J. Freixenet, X. Munoz, D. Raba, J. Marti, X. Cuff, Yet another survey on image segmentation, in: Proceedings of European Conference on Computer Vision, 2002, pp. 408–422.
[9] T. Gevers, A. Smeulders, Combining region splitting and edge detection through guided Delaunay image subdivision, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, 1997, pp. 1021–1026.
[10] Z. Ghahramani, G. Hinton, The EM Algorithm for Mixtures of Factor Analyzers, Tech. Rep. CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996.
[11] R. Haralick, L. Shapiro, Image segmentation techniques, Computer Vision, Graphics, and Image Processing 29 (1) (1985) 100–132.
[12] S. Kamvar, D. Klein, C. Manning, Interpreting and Extending Classical Agglomerative Clustering Methods using a Model-Based Approach, Tech. Rep. 2002-11, Department of Computer Science, Stanford University, 2002.
[13] T. Kanungo, B. Dom, W. Niblack, D. Steele, A fast algorithm for MDL-based multi-band image segmentation, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, 1994, pp. 609–616.
[14] E. Levina, P. Bickel, The earth mover's distance is the Mallows distance: some insights from statistics, in: Proceedings of IEEE International Conference on Computer Vision, vol. 2, 2001, pp. 251–256.
[15] L. Lucchese, S. Mitra, Color image segmentation: a state-of-the-art survey, in: Proceedings of the Indian National Science Academy, 2001.
[16] Y. Ma, H. Derksen, W. Hong, J. Wright, Segmentation of multivariate mixed data via lossy coding and compression, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (9) (2007) 1546–1562.
[17] M. Madiman, M. Harrison, I. Kontoyiannis, Minimum description length vs. maximum likelihood in lossy data compression, in: Proceedings of the 2004 IEEE International Symposium on Information Theory, 2004, p. 461.
[18] J. Malik, S. Belongie, T. Leung, J. Shi, Contour and texture analysis for image segmentation, International Journal on Computer Vision 43 (1) (2001) 7–27.
[19] B. Manjunath, R. Chellappa, Unsupervised texture segmentation using Markov random field models, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (5) (1991) 478–482.

[20] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proceedings of IEEE International Conference on Computer Vision, 2001, pp. 416–423.
[21] G. McLachlan, T. Krishnan, The EM Algorithm and Extensions, John Wiley & Sons, 1997.
[22] M. Meila, Comparing clusterings: an axiomatic view, in: Proceedings of the International Conference on Machine Learning, 2005, pp. 577–584.
[23] G. Mori, Guiding model search using segmentation, in: Proceedings of IEEE International Conference on Computer Vision, vol. 2, 2005, pp. 1417–1423.
[24] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, International Journal on Computer Vision 42 (3) (2001) 145–175.
[25] B. Olshausen, D. Field, Natural image statistics and efficient coding, Network: Computation in Neural Systems 7 (1996) 333–339.
[26] D. Panjwani, G. Healey, Markov random field models for unsupervised segmentation of textured color images, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (10) (1995) 939–954.
[27] C. Pantofaru, M. Hebert, A Comparison of Image Segmentation Algorithms, Tech. Rep. CMU-RI-TR-05-40, Carnegie Mellon University, 2005.
[28] Y. Rubner, J. Puzicha, C. Tomasi, J. Buhmann, Empirical evaluation of disimilarity measures for color and texture, Computer Vision and Image Understanding 84 (2001) 25–43.
[29] Y. Rubner, C. Tomasi, L. Guibas, The earth mover's distance as a metric for image retrieval, International Journal on Computer Vision 40 (2) (2000) 99–121.
[30] C. Schmid, Constructing models for content-based image retrieval, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, vol. 2, 2001, pp. 39–45.
[31] J. Shi, J. Malik, Normalized cuts and image segmentation, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, 1997, pp. 731–737.
[32] M. Tipping, C. Bishop, Probabilistic principal component analysis, Journal of Royal Statistical Society: Series B 61 (3) (1999) 611–622.
[33] A. Tremeau, N. Borel, A region growing and merging algorithm to color segmentation, Pattern Recognition 30 (7) (1997) 1191–1204.
[34] Z. Tu, S. Zhu, Image segmentation by data-driven Markov Chain Monte Carlo, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 657–673.
[35] N. Ueda, R. Nakan, Z. Ghahramani, SMEM algorithm for mixture models, Neural Computation 12 (2000) 2109–2128.
[36] M. Varma, A. Zisserman, Texture classification: are filter banks necessary?, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, vol. 2, 2003, pp. 691–698.
[37] J. Ward, Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association 58 (1963) 236–244.
[38] J. Wright, Y. Ma, Y. Tao, Z. Lin, H. Shum, Classification via Minimum Incremental Coding Length (MICL), Tech. Rep. UILU-ENG-07-2201, University of Illinois at Urbana-Champaign, 2007.
[39] S. Yu, Segmentation induced by scale invariance, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 444–451.
[40] S. Zhu, C. Guo, Y. Wang, Z. Xu, What are textons? International Journal on Computer Vision 62 (2005) 121–143.
[41] S. Zhu, A. Yuille, Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (9) (1996) 884–900.
[42] S. Zhu, Statistical modeling and conceptualization of visual patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (6) (2003) 691–712.