International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Segmentation of Natural Images by Texture and Boundary Compression

Hossein Mobahi · Shankar R. Rao · Allen Y. Yang · Shankar S. Sastry · Yi Ma

the date of receipt and acceptance should be inserted later

Abstract We present a novel algorithm for segmentation of natural images that harnesses the principle of minimum description length (MDL). Our method is based on observations that a homogeneously textured region of a natural image can be well modeled by a Gaussian distribution and the region boundary can be effectively coded by an adaptive chain code. The optimal segmentation of an image is the one that gives the shortest coding length for encoding all textures and boundaries in the image, and is obtained via an agglomerative clustering process applied to a hierarchy of decreasing window sizes as multi-scale texture features. The optimal segmentation also provides an accurate estimate of the overall coding length and hence the true entropy of the image. We test our algorithm on the publicly available Berkeley Segmentation Dataset. It achieves state-of-the-art segmentation results compared to other existing methods.

Research was supported in part by NSF IIS 07-03756, ONR N00014-09-1-0230, ARO MURI W911NF-06-1-0076, and ARL MAST-CTA W911NF-08-2-0004. Hossein Mobahi was supported by Computational Science & Engineering (CSE) Ph.D fellowship of University of Illinois at Urbana Champaign. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, U.S. Government, or the CSE program. The U.S. Government is authorized to reproduce and distribute for Government purposes notwithstanding any copyright notation hereon.

H. Mobahi and Y. Ma
Coordinated Science Lab, University of Illinois, Urbana, IL 61801, USA.
E-mail: {hmobahi2,yima}@illinois.edu

S. Rao
HRL Laboratories, LLC, Malibu, CA 90265, USA.
E-mail: srrao@hrl.com

A. Yang and S. Sastry
Cory Hall, Department of EECS, University of California, Berkeley, CA 94720, USA.
E-mail: {yang,sastry}@eecs.berkeley.edu

Y. Ma
Visual Computing Group, Microsoft Research Asia, Beijing, China.

1 Introduction

The task of partitioning a natural image into regions with homogeneous texture, commonly referred to as *image segmentation*, is widely accepted as a crucial function for high-level image understanding, significantly reducing the complexity of content analysis of images. Image segmentation and its higher-level applications are largely designed to emulate functionalities of human visual perception (e.g., in object recognition and scene understanding). Dominant criteria for measuring segmentation performance are based on qualitative and quantitative comparisons with human segmentation results. In the literature, investigators have explored several important models and principles that can lead to good image segmentation:

1. Different texture regions of a natural image admit a mixture model. For example, Multiscale Normalized Cuts (MNC) by Cour et al. (2005), F&H by Felzenszwalb & Huttenlocher (2004), Normalized Tree Partitioning by Wang et al. (2008), and Multi-Layer Spectral Segmentation by Kim et al. (2010) formulate the segmentation as a graph-cut problem, while Mean Shift (MS) by Comanicu & Meer (2002) seeks a partition of a color image based on different modes within the estimated empirical distribution.
2. Region contours/edges convey important information about the saliency of the objects in the image and their shapes (see Elder & Zucker (1996); Gevers & Smeulders (1997); Arbelaez (2006); Zhu et al. (2007); Ren et al. (2008)). Several recent methods have been proposed to combine the cues of homogeneous color and texture with the cue of contours in the segmentation process, including Malik et al. (2001); Tu & Zhu (2002); Kim et al. (2005).
3. The properties of local features (including texture and edges) usually do not share the same level of homogeneity at the same spatial scale. Thus, salient image regions can only be extracted from a hierarchy of image features under multiple scales (see Yu (2005); Ren et al. (2005); Yang et al. (2008); Donoser et al. (2009)).

Despite much work in this area, good image segmentation remains elusive to obtain for practitioners, mainly for the following two reasons: First, there is little consensus on what criteria should be used to evaluate the quality of image segmentations. It is difficult to strike a good balance between objective measures that depend solely on the intrinsic statistics of imagery data and subjective measures that try to empirically mimic human perception. Second, in the search for objective measures, there has been a lack of consensus on good models for a unified representation of image segments including both their textures and contours.

Recently, an *objective* metric based on the notion of lossy *minimum description length* (MDL) has been proposed for evaluating clustering of general mixed data (Ma et al. (2007)). The basic idea is that, given a potentially mixed data set, the "optimal segmentation" is the one that, over all possible segmentations, minimizes the coding length of the data, subject to a given quantization error. For data drawn from a mixture of Gaussians, the optimal segmentation can often be found efficiently using an agglomerative clustering approach. The MDL principle and the new clustering method have later been applied to the segmentation of natural images, known as *compression-based texture merging* (CTM) (Yang et al. (2008)). This approach has proven to be highly effective for imitating human segmentation of natural images. Preliminary success of this approach leads to the following important question: *To what extent is segmentation obtained by image compression consistent with human perception?*

The CTM algorithm also has its drawbacks. In particular, although the CTM method utilizes the idea of data compression, it does not exactly seek to compress the image *per se*. First, it "compresses" feature vectors or windows extracted around all pixels by grouping them into clusters as a mixture of Gaussian models. As a result, the final coding length is highly *redundant* due to overlapping between windows of adjacent pixels, and has no direct relation to the true entropy of the image. Second, the segmentation result encodes the membership of pixels using a Huffman code, which does not take into account the smoothness of boundaries nor the spatial relationship of adjacent pixels that are more likely belong to one texture region. Thus, CTM does not give a good estimate of the true entropy of the image, and it cannot be used to justify a strong connection between image segmentation and image compression.

1.1 Contributions

In this paper, we contend that, much better segmentation results can be obtained if we more closely adhere to the principle of image compression, by correctly counting only the necessary bits needed to encode a natural image for both the texture and boundaries. The proposed algorithm precisely estimates the coding length needed to encode the texture of each region based on the rate distortion of its probabilistic distribution and the number of *non-overlapping* interior windows. In order to adapt to different scales and shapes of texture regions in an image, a hierarchy of multiple window sizes is incorporated in the segmentation process. The algorithm further encodes the boundary information of each homogeneous texture region by carefully counting the number of bits needed to encode the boundary with an adaptive chain code.

Based on the MDL principle, the optimal segmentation of an image is defined as the one that minimizes its total coding length, in this case a close approximation to the true entropy of the image. With any fixed quantization, the final coding length gives a purely objective measure for how good the segmentation is in terms of the level of image compression. Finally, we propose a simple yet effective regression method to adaptively select a proper quantization level for each individual image to achieve the optimal segmentation result.

We conduct extensive experiments to compare the results with human segmentation using the Berkeley Segmentation Dataset (BSD) (Martin et al. (2001)). Although our method is conceptually simple and the measure used is purely objective, the segmentation results match extremely well with those by humans, exceeding or competing with the best segmentation algorithms.

2 Coding Length Functions for Texture and Boundary

2.1 Construction of Texture Features

We first discuss how to construct texture vectors that represent homogeneous textures in image segments. In order to capture the variation of a local *texton*, one can directly apply a $w \times w$ cut-off window around a pixel across the three color channels, and stack the color values inside the window in a vector form as in Yang et al. (2008). [1]

[1] Another popular approach for constructing texture vectors is to use multivariate responses of a fixed 2-D texture filter bank. A previous study by Varma & Zisserman (2003) has argued

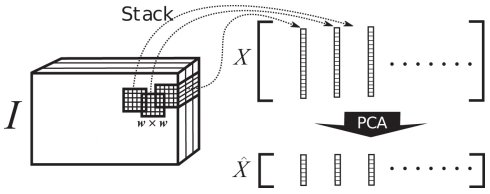


Fig. 1 Texture features are constructed by stacking the $w \times w$ windows around all pixels of a color image I into a data matrix X and then projected to a low-dimensional space via principal component analysis (PCA).

Figure 1 illustrates the process of constructing texture features. Let the w-neighborhood $\mathcal{W}_w(p)$ be the set of all pixels in a $w \times w$ window across three color channels (e.g., RGB or $L^*a^*b^*$) centered at pixel p. Define the set of features X by taking the w-neighborhood around each pixel in I, and then stacking the window as a column vector:

$$X \doteq \{\boldsymbol{x}_p \in \mathbb{R}^{3w^2} : \boldsymbol{x}_p = \mathcal{W}_w(p)^S \text{ for } p \in I\}. \tag{1}$$

For ease of computation, we further reduce the dimensionality of these features by projecting the set of all features X onto their first D principal components. We denote the set of features with reduced dimensionality as $\hat{X}$. We have observed that for many natural images, the first eight principal components of X contain over 99% of the energy. In this paper, we choose to assign $D = 8$.

Over the years, there have been many proposed methods to model the representation of image textures in natural images. One model that has been shown to be successful in encoding textures both empirically and theoretically is the Gaussian *Mesh Markov Model* (MMM) (Levina & Bickel, 2006). Particularly in texture synthesis, the Gaussian MMM can provide consistent estimates of the joint distribution of the pixels in a window, which then can be used to fill in missing texture patches via a simple nonparametric scheme (Efros & Leung, 1999).

However, to determine the optimal compression rate for samples from a distribution, one must know the rate-distortion function of that distribution (Yang et al., 2008). Unfortunately, to our knowledge, the rate-distortion function for MMMs is not known in closed form and difficult to estimate empirically. Over all distributions with the same variance, it is known that the Gaussian distribution has the highest rate-distortion, and is in this sense the worst case distribution for compression. Thus by using the rate-distortion for a Gaussian distribution, we obtain an *upper bound* for the true coding length of the MMM.

2.2 Texture Encoding

To describe encoding texture vectors, we first consider a single region R with N pixels. Based on Yang et al. (2008), for a fixed quantization error ε, the expected number of bits needed to code the set of N feature windows $\hat{X}$ up to distortion ε^2 is given by:

$$L_\varepsilon(\hat{X}) \doteq \underbrace{\tfrac{D}{2}\log_2\det(I + \tfrac{D}{\varepsilon^2}\Sigma)}_{\text{codebook}} + \underbrace{\tfrac{N}{2}\log_2\det(I + \tfrac{D}{\varepsilon^2}\Sigma)}_{\text{data}} + \underbrace{\tfrac{D}{2}\log_2(1 + \tfrac{\|\boldsymbol{\mu}\|^2}{\varepsilon^2})}_{\text{mean}}, \tag{2}$$

that the difference in segmentation results between the two approaches is small, and yet it is more expensive to compute 2-D filter bank responses.

where $\boldsymbol{\mu}$ and Σ are the mean and covariance of the vectors in $\hat{X}$. Equation (2) is the sum of three coding-lengths for the D Gaussian principal vectors as the codebook, the N vectors w.r.t. that codebook, and the mean of the Gaussian distribution.

The coding length function (2) is uniquely determined by the mean and covariance $(\boldsymbol{\mu}, \Sigma)$. To estimate them empirically, we need to exclude the windows that cross the boundary of R (as shown in Figure 2(a)). Such windows contain textures from the adjacent regions, which cannot be well modeled by a single Gaussian as the interior windows. Hence, the empirical mean $\hat{\boldsymbol{\mu}}_w$ and covariance $\hat{\Sigma}_w$ of R are only estimated from the *interior* of R:

$$\mathcal{I}_w(R) \doteq \{p \in R : \forall q \in \mathcal{W}_w(p), q \in R\}. \tag{3}$$

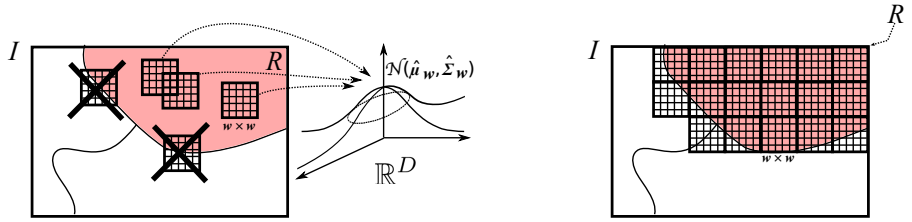


Fig. 2 (a) Only windows from the interior of a region are used to compute the empirical mean $\hat{\boldsymbol{\mu}}_w$ and covariance $\hat{\Sigma}_w$. (b) Only nonoverlapping windows that can tile R as a grid are encoded.

Furthermore, in (2), encoding all texture vectors in $\hat{X}$ to represent region R is highly redundant because the N windows *overlap* with each other. Thus, to obtain an efficient code of R that closely approximates its true entropy, we only need to code the *nonoverlapping* windows that can tile R as a grid, as in Figure 2 (b).

Ideally, if R is a rectangular region of size $mw \times nw$, where m and n are positive integers, then clearly we can tile R with exactly $mn = \frac{N}{w^2}$ windows. So for coding the region R, (2) becomes:

$$L_{w,\varepsilon}(R) \doteq (\tfrac{D}{2} + \tfrac{N}{2w^2}) \log_2 \det(I + \tfrac{D}{\varepsilon^2} \hat{\Sigma}_w) + \tfrac{D}{2} \log_2(1 + \tfrac{\|\hat{\boldsymbol{\mu}}_w\|^2}{\varepsilon^2}). \tag{4}$$

Real regions in natural images normally do not have such nice rectangular shapes. However, (4) remains a good approximation to the actual coding length of a region R with relatively smooth boundaries.[2]

2.3 Boundary Encoding

To code windows from multiple regions in an image, one must know to which region each window belongs, so that each window can be decoded w.r.t. the correct codebook. For generic samples from multiple classes, one can estimate the distribution of

[2] For a large region with a sufficiently smooth boundary, the number of boundary-crossing windows is significantly smaller than the number of those in the interior. For boundary-crossing windows, their average coding length is roughly proportional to the number of pixels inside the region if the Gaussian distribution is sufficiently isotropic.

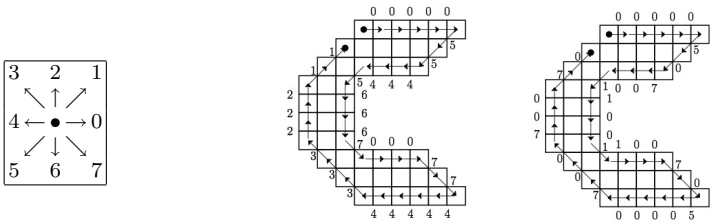


Fig. 3 Left: The Freeman chain code of an edge orientation along 8 possible directions. **Middle:** Representation of the boundary of a region in an image w.r.t. the Freeman chain code. **Right:** Representation w.r.t the difference chain code.

each class label and then code the membership of the samples using a scheme that is asymptotically optimal for that class distribution (such as the Huffman code used in Yang et al. (2008)). Such coding schemes are highly inefficient for natural image segmentation, as they do not leverage the spatial correlation of pixels in the same region. In fact, for our application, pixels from the same region form a connected component. Thus, the most efficient way of coding group membership for regions in images is to code the *boundary* of the region containing the pixels.

A well-known scheme for representing boundaries of image regions is the *Freeman chain code*. In this coding scheme, the orientation of an edge element is quantized along 8 discrete directions, shown in Figure 3. Let $\{o_t\}_{t=1}^{T}$ denote the orientations of the T boundary pixels of R. Since each chain code can be encoded using three bits, the coding length of the boundary of R is

$$B(R) = 3 \sum_{i=0}^{7} \#(o_t = i). \tag{5}$$

The coding length $B(R)$ can be further improved by using an adaptive Huffman code that leverages the prior distribution of the chain codes. Though the distribution of the chain codes is essentially uniform in most images, for regions with smooth boundaries, we expect that the orientations of consecutive edges are similar, and so consecutive chain codes will not differ by much. Given an initial orientation (expressed in chain code) o_t, the *difference chain code* of the following orientation o_{t+1} is $\Delta o_t \doteq \mod (o_t - o_{t+1}, 8)$. Figure 3 compares the original Freeman chain code with the difference chain code for representing the boundary of a region. Notice for this region, the difference encoding uses only half of the possible codes, with most being zeroes, while the Freeman encoding uses all eight chain codes. Given the prior distribution $P[\Delta o]$ of difference chain codes, $B(R)$ can be encoded more efficiently using a lossless Huffman coding scheme:

$$B(R) = -\sum_{i=0}^{7} \#(\Delta o_t = i) \log_2(P[\Delta o = i]). \tag{6}$$

For natural images, we estimate $P[\Delta o]$ using images from the BSD that were manually segmented by humans. We compare our distribution with the one estimated by Liu & Zalik (2005), which used 1000 images of curves, contour patterns, and shapes obtained from the web. As the results in Table 1 show, the regions of natural images tend to have more smooth boundaries when segmented by humans.

Table 1 The prior probability of the difference chain codes estimated from the BSD and by Liu & Zalik (2005).

Difference Code	0	1	2	3	4	5	6	7
Angle change	0°	45°	90°	135°	180°	−135°	−90°	−45°
Probability (BSD)	0.585	0.190	0.020	0.000	0.002	0.003	0.031	0.169
Probability (Liu-Zalik)	0.453	0.244	0.022	0.006	0.003	0.006	0.022	0.244

3 Image Segmentation Algorithm

In this section, we discuss how to use the coding length functions to construct a better compression-based image segmentation algorithm. We first describe a basic approach. Then we propose a hierarchical scheme to deal with small regions using multi-scale texture windows. Finally, we investigate a simple yet effective regression scheme to adaptively choose a proper distortion parameter ε based on a set of manually labeled segmentation examples.

3.1 Minimization of the Total Coding Length Function

Suppose an image I can be segmented into non-overlapping regions $\mathcal{R} = \{R_1, \ldots, R_k\},\ \cup_{i=1}^{k} R_i = I$. The total coding length of the image I is

$$L^S_{w,\varepsilon}(\mathcal{R}) \doteq \sum_{i=1}^{k} L_{w,\varepsilon}(R_i) + \tfrac{1}{2} B(R_i). \tag{7}$$

Here, the boundary term is scaled by a half because we only need to represent the boundary between any two regions once. The optimal segmentation of I is the one that minimizes (7). Finding this optimal segmentation is, in general, a combinatorial task, but we can often do so using an *agglomerative* approximation.

To initialize the optimization process, one can assume each image pixel (and its windowed texture vector) belongs to an individual group of its own. However, this presents a problem that the maximal size of the texture window can only be one without intersecting with other adjacent regions (i.e., other neighboring pixels). In our implementation, similar to Yang et al. (2008), we utilize an oversegmentation step to initialize the optimization by *superpixels*. A superpixel is a small region in the image that does not contain strong edges in its interior. Superpixels provide a coarser quantization of an image than the underlying pixels, while respecting strong edges between the adjacent homogeneous regions. There are several methods that can be used to obtain a superpixel initialization, including those of Mori et al. (2004), Felzenszwalb & Huttenlocher (2004), and Ren et al. (2005). We have found that Mori et al. (2004)[3] works well for our purposes.

Given an oversegmentation of the image, at each iteration, we find the pair of regions R_i and R_j that will maximally decrease (7) if merged:

$$(R_i^*, R_j^*) = \underset{R_i, R_j \in \mathcal{R}}{\operatorname{argmax}} \ \Delta L_{w,\varepsilon}(R_i, R_j), \quad \text{where}$$

[3] We use the publicly available code for this method available at `http://www.cs.sfu.ca/~mori/research/superpixels/` with parameter `N_sp` = 200.

$$\begin{aligned}\Delta L_{w,\varepsilon}(R_i, R_j) &\doteq L^S_{w,\varepsilon}(\mathcal{R}) - L^S_{w,\varepsilon}((\mathcal{R}\backslash\{R_i, R_j\}) \cup \{R_i \cup R_j\}) \\ &= L_{w,\varepsilon}(R_i) + L_{w,\varepsilon}(R_j) - L_{w,\varepsilon}(R_i \cup R_j) \\ &\quad + \tfrac{1}{2}(B(R_i) + B(R_j) - B(R_i \cup R_j)). \end{aligned} \tag{8}$$

$\Delta L_{w,\varepsilon}(R_i, R_j)$ essentially captures the difference in the lossy coding lengths of the texture regions R_i and R_j and their boundaries before and after the merging. If $\Delta L(R^*_i, R^*_j) > 0$, we merge R^*_i and R^*_j into one region, and repeat the process until the coding length $L^S_{w,\varepsilon}(\mathcal{R})$ can not be further reduced.

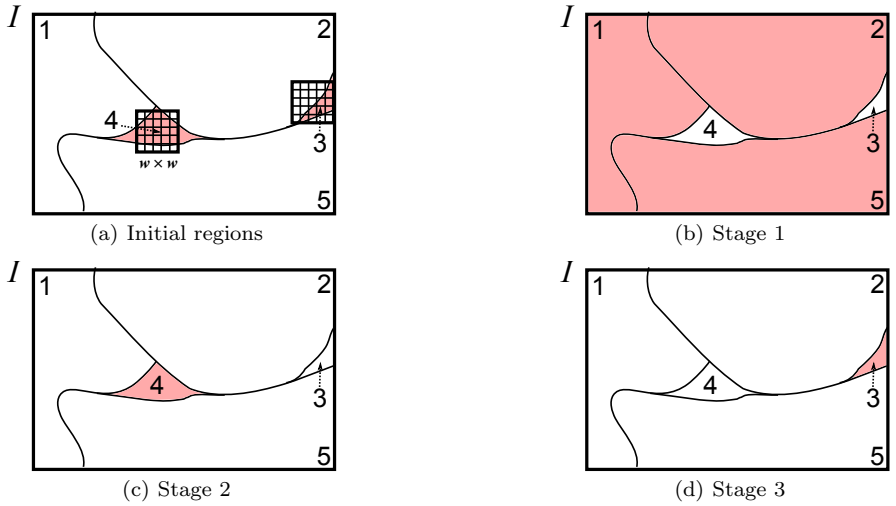
To model the spatial locality of textures, we further construct a *region adjacency graph* (RAG): $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each vertex $v_i \in \mathcal{V}$ corresponds to region $R_i \in \mathcal{R}$, and an edge $e_{ij} \in \mathcal{E}$ indicates that regions R_i and R_j are adjacent in the image. To perform image segmentation, we simply apply a constrained version of the above agglomerative procedure – only merging regions that are adjacent in the image. The proposed region-merging method has been widely used by other image segmentation algorithms (Haralick & Shapiro (1985); Tremeau & Borel (1997); Deng & Manjunath (2001)).

In terms of the computational complexity, one can show that the agglomerative clustering process that iteratively minimizes (7) is a polynomial time algorithm. More specifically, let w be the window size, n be the image size, and k be the number of initial superpixel segments. One can show that the computational complexity of agglomerative clustering is bounded by $O(kw^6 + n^2w^2)$. Also note that the complexity bound has ignored the cost to sort and maintain the ordering of the coding length difference (8), as the algorithm can use a heap structure to efficiently implement the sorting and re-sorting algorithms (Kurita (1995)).

3.2 Hierarchical Implementation

The above region-merging scheme is based on the assumption of a fixed texture window size, and clearly cannot effectively deal with regions or superpixels that are very small. In such cases, the majority of the texture windows will intersect with the boundary of the regions. We say that a region R is *degenerate* w.r.t. window size w if $\mathcal{I}_w(R) = \emptyset$. For such regions, the w-neighborhoods of all pixels will contain pixels from other regions, and so $\hat{\boldsymbol{\mu}}$ and $\hat{\Sigma}$ cannot be reliably estimated. These regions are degenerate precisely because of the window size; for any w-degenerate region R, there is $1 \leq w' < w$ such that $\mathcal{I}_{w'}(R) \neq \emptyset$. We say that R is *marginally nondegenerate* w.r.t. window size w if $\mathcal{I}_w(R) \neq \emptyset$ and $\mathcal{I}_{w+2}(R) = \emptyset$. To deal with these degenerate regions, we propose to use a hierarchy of window sizes. Starting from the largest window size, we recursively apply the above scheme with ever smaller window sizes till all degenerate regions have been merged with their adjacent ones. In this paper, we start from 7×7 and reduce to $5 \times 5, 3 \times 3$, and 1×1. Please refer to Figure 4 for an example of our hierarchical scheme.

Notice that at a fixed window size, the region-merging process is similar to the CTM approach proposed in Yang et al. (2008). Nevertheless, the new coding length function and the hierarchical implementation give much more accurate approximation to the true image entropy and hence lead to much better segmentation results. We summarize the overall algorithm for image segmentation in Algorithm 1, which we refer to as *Texture and Boundary Encoding-based Segmentation* (TBES).



(a) Initial regions (b) Stage 1

(c) Stage 2 (d) Stage 3

Fig. 4 An example of our scheme for hierarchical image segmentation (a) Initial set of regions. Note that regions 3 and 4 are degenerate w.r.t. the window size w. (b) In the first stage, only nondegenerate regions 1, 2, and 5 are considered for merging. (c) In the next stage, w is reduced, causing region 4 to be marginally nondegenerate. We consider merging region 4 with it's nondegenerate neighbors. (d) In the last stage, w is reduced enough so that region 3 becomes nondegenerate. These stages are repeated until the overall coding length can no longer be reduced.

3.3 Choosing the Distortion Level

Algorithm 1 requires a single parameter, the distortion level ε, that determines the coarseness of the segmentation. The optimality of ε is measured by the segmentation that best matches with human perception. As shown in Figure 5, since natural images have different scales of resolution, no single choice of ε is optimal for all images. In this section, we propose a solution to adaptively select a proper distortion parameter such that the segmentation result better approximates human perception. The method assumes that a set of training images $\mathcal{I} = \{I_1, \cdots, I_K\}$ have been manually segmented by human users as the ground truth set $\mathcal{S}_g = \{\mathcal{R}_g(I_1), \cdots, \mathcal{R}_g(I_K)\}$.

To objectively quantify how well a given segmentation matches with human perception, we first need a measure for the discrepancy between two segmentations $\mathcal{R}_1$ and $\mathcal{R}_2$, denoted as $d(\mathcal{R}_1, \mathcal{R}_2)$. Intuitively, the discrepancy measure should be small when $\mathcal{R}_1$ and $\mathcal{R}_2$ are similar in some specific sense.[4] Given a measure d, the best ε for I_i, denoted by ε_i^*, can be obtained by:

$$\varepsilon_i^* = \arg\min_{\varepsilon} d(\mathcal{R}_\varepsilon(I_i), \mathcal{R}_g(I_i)), \quad \text{for each } I_i \in \mathcal{I}. \tag{9}$$

An example of the relationship between ε and a discrepancy measure d is shown in Figure 6.

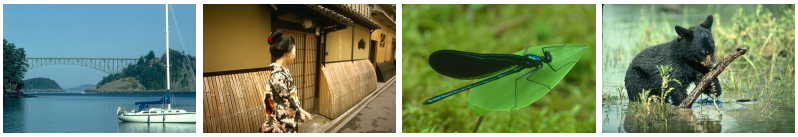
As ground truth segmentations are not available for non-training images, we shall use the training images $\mathcal{S}_g = \{\mathcal{R}_g(I_i)\}$ to infer ε for a test image. A classical technique

[4] We will discuss several discrepancy measures in Section 4.2, such as the probabilistic Rand index (PRI) and variation of information (VOI).

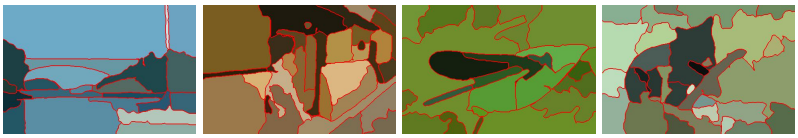
Algorithm 1 Texture and Boundary Encoding-based Segmentation (TBES)

Given image I, distortion ε, max window size w_M, superpixels $\mathcal{R} = \{R_1, \ldots, R_k\}$,

for $w = 1 : 2 : w_M$ **do**
  Construct $\hat{X}_w$ by stacking the $w \times w$ windows around each $p \in I$ as column vectors and applying PCA.
Construct RAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \simeq \mathcal{R}$ and $e_{ij} \in \mathcal{E}$ only if R_i and R_j are adjacent in I.
$w = w_M$
repeat
  if $w = w_M$ **then**
    Find R_i and R_j such that $e_{ij} \in \mathcal{E}$, $\mathcal{I}_w(R_i) \neq \emptyset$, $\mathcal{I}_w(R_j) \neq \emptyset$, and $\Delta L_{w,\varepsilon}(R_i, R_j)$ is maximal.
  else
    Find R_i and R_j such that $e_{ij} \in \mathcal{E}$, $\mathcal{I}_w(R_i) \neq \emptyset$, $\mathcal{I}_w(R_j) \neq \emptyset$, $\mathcal{I}_{w+2}(R_i) = \emptyset$ or $\mathcal{I}_{w+2}(R_j) = \emptyset$ and $\Delta L_{w,\varepsilon}(R_i, R_j)$ is maximal.
  if $\Delta L_{w,\varepsilon}(R_i, R_j) > 0$ **then**
    $\mathcal{R} := (\mathcal{R} \setminus \{R_i, R_j\}) \cup \{R_i \cup R_j\}$.
    Update $\mathcal{G}$ based on the newly merged region.
    $w = w_M$
  else if $w \neq 1$ **then**
    $w = w - 2$
until $\mathcal{I}_{w_M}(R) \neq \emptyset, \quad \forall R \in \mathcal{R} \quad$ **and** $\quad \Delta L_{w_M,\varepsilon}(R_i, R_j) \leq 0, \quad \forall R_i, R_j \in \mathcal{R}$
Output: The set of regions $\mathcal{R}$.



(a) Original images.



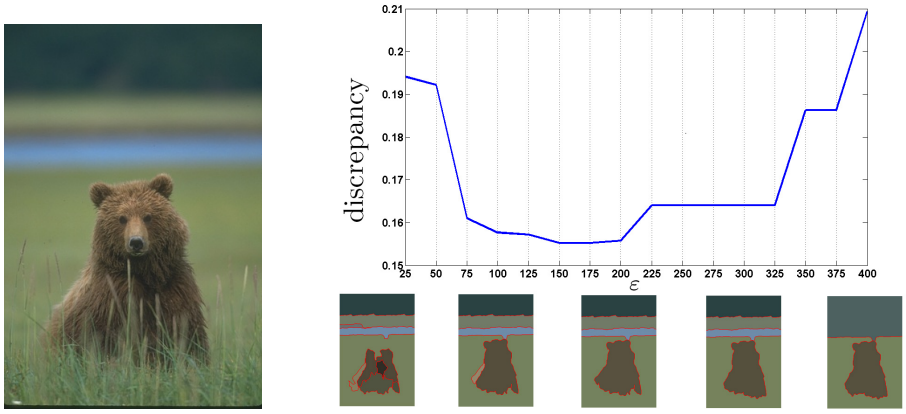
(b) Segmentation results with distortion ($\varepsilon = 25$)

(c) Segmentation results with distortion ($\varepsilon = 400$)

Fig. 5 A comparison of segmentation results w.r.t. different distortion levels. The low distortion generates better segmentations for the left two images, while the high distortion generates better results for the right two images.

for estimating a continuous parameter, such as ε, from training data is *linear regression* (Duda et al. (2001)). The method requires a pair $(\varepsilon_i, \boldsymbol{f}_i)$ per training image I_i, where ε_i is the "optimal" distortion for image I_i and $\boldsymbol{f}_i$ is a set of features extracted from I_i. Then the regression parameters $\boldsymbol{w}$ can be estimated by solving the following objective function:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_i (\boldsymbol{w}^T \boldsymbol{f}_i - \varepsilon_i^*)^2. \tag{10}$$



(a) Input Image (b) Distortion vs Discrepancy of Segmentation

Fig. 6 The effect of distortion ε on the discrepancy $d(\mathcal{R}_\varepsilon(I_i), \mathcal{R}_g(I_i))$ on an example image. The discrepancy shown in the plot is the probability that an arbitrary pair of pixels do not have consistent labels in $\mathcal{R}_\varepsilon(I_i)$ and $\mathcal{R}_g(I_i)$, namely, PRI^C (please refer to Section 4.2).

The distortion level ε w.r.t. a new test image I with its feature vector $\boldsymbol{f}$ is given by $\varepsilon(\boldsymbol{f}) \doteq \boldsymbol{w}^{*T}\boldsymbol{f}$.

The features $\boldsymbol{f}_i$ in (10) should be chosen to effectively model the statistics of the image, so that the relationship between ε and $\boldsymbol{f}_i$ is well approximated by the linear function $\varepsilon_i \approx \boldsymbol{w}^T\boldsymbol{f}_i$. A simple idea to define $\boldsymbol{f}_i$ could consider how contrastive the regions in I_i are. Intuitively, when the textures in I_i are similar, such as in camouflage images, stronger sensitivity to contrast in patterns is required. Since computing the standard deviation of pixel intensities gives a measure of pattern contrast, we resize each I_i with multiple scales, and define the features $\boldsymbol{f}_i$ as the standard deviations of the pixel intensities at the multiple image resolutions.

Another issue in linear regression is that the classical model (10) is insufficient to accurately predict the distortion level for Algorithm 1. In particular, the discrepancy measure d is only used to determine the optimal ε^* for a training image. Segmentation results for other choices of ε are not used in the regression. However, it is possible to better estimate the distortion ε by taking into account the segmentation results around a neighborhood of the optimal distortion ε^* in the training set.

For agglomerative image segmentation, the discrepancy measures that we use in this paper exhibit a simple behavior. Specifically, as ε deviates from ε^* in either direction, the discrepancy between the segmentation and the ground truth almost increases monotonically. This is because as ε deviates from ε^*, it leads to over-segmentation or under-segmentation, both of which have larger discrepancies from the ground truth (see Figure 6). Motivated by this observation, we approximate the discrepancy function d by a convex quadratic form:

$$d(\mathcal{R}_\varepsilon(I_i), \mathcal{R}_g(I_i)) \approx a_i\varepsilon^2 + b_i\varepsilon + c_i, \quad \text{where } a_i > 0. \tag{11}$$

The parameters (a_i, b_i, c_i) are then estimated by least squares fitting w.r.t. the pairs (d_i, ε). The latter is attained by sampling the function $d(S_\varepsilon(I_i), S_g(I_i))$ at different ε's.

Once we substitute (11) in (9) in combination with the linear model $\varepsilon = \boldsymbol{w}^T \boldsymbol{f}_i$, the objective function to recover the linear regression parameter $\boldsymbol{w}^*$ is given by

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_i a_i (\boldsymbol{w}^T f_i)^2 + b_i (\boldsymbol{w}^T f_i) + c_i. \tag{12}$$

Since $a_i > 0$ for all training images I_k, (12) is an unconstrained convex program. Thus it has a closed-form solution:

$$\boldsymbol{w}^* = -\frac{1}{2} (\sum_i a_i \boldsymbol{f}_i \boldsymbol{f}_i^T)^{-1} (\sum_i b_i \boldsymbol{f}_i). \tag{13}$$

Once $\boldsymbol{w}^*$ is learned from the training data, the optimal distortion of the test image I with its feature vector $\boldsymbol{f}$ is predicted by $\varepsilon(\boldsymbol{f}) = \boldsymbol{w}^{*T} \boldsymbol{f}$. We caution that, based on $\boldsymbol{w}^*$, the prediction of the distortion parameter $\varepsilon(\boldsymbol{f}_i)$ for each training image I_i may not necessarily be the same as ε_i^* selected from the ground truth $S_g(I_i)$. Nevertheless, the proposed solution ensures that the linear model minimizes the average discrepancy over the training data.

4 Experiments

In this section, we conduct extensive evaluation to validate the performance of the TBES algorithm. The experiment is based on the publicly available Berkeley Segmentation Dataset (BSD) (Martin et al. (2001)). BSD is comprised of 300 natural images, which covers a variety of natural scene categories, such as portraits, animals, landscape, and beaches. The database is partitioned into a training set of 200 images and a testing set of 100 images. It also provides ground-truth segmentation results of all the images obtained by several human subjects. On average, five segmentation maps are available per image. Multiple ground truth allows us to investigate how human subjects agree with each other. The average run time of our implementation in MATLAB for segmenting an image in BSD is 164.59 s for feature construction, 0.49 s for vicinity map construction, and 412.51 s for segmentation. The numbers were obtained on a 3Ghz Intel processor with 1GB of RAM.

The implementation of the TBES algorithm and the benchmark scripts are available online at: `http://perception.csl.illinois.edu/coding/image_segmentation/`.

4.1 Color Spaces and Compressibility

The optimal coding length of textured regions of an image depends in part on the color space. We seek to determine the color space in which natural images are most compressible based on the proposed lossy compression scheme (7). It has been noted in the literature that the *Lab* color space (also known as $L^*a^*b^*$) better approximates the perceptually uniform color metric (Jain, 1989). This has motivated some of the previous works (Yang et al., 2008; Rao et al., 2009) to utilize such representation in methods for natural image segmentation. In order to check the validity of this assumption, particularly for our segmentation scheme by compressing texture, we perform a study on five color spaces that have been widely used in the literature, namely, Lab, YUV, RGB, XYZ, and HSV.

We use the manually segmented training images in the Berkeley dataset to rank the compressibility of the 5 color spaces. Given a color space, for any image and corresponding segmentation, the number of bits required to encode texture information is computed by (2), with features constructed as in Section 2.1. The average coding length of an image is computed as the one over all ground-truth segmentation maps for that image. Finally, the average coding length of the dataset is computed over the entire images in the dataset.

We note that the volume of the pixel distribution (and thus the coding length) can change if the pixel values are rescaled. This means one color space can look more compressible by merely producing numbers in a smaller range, say $[0, 1]$ as opposed to another which is in range $[0, 255]$. In order to achieve a fair comparison, we normalize the feature vectors by scale factor c, which is constant across features from the same color space:

$$c = 1/\sqrt{\bar{\lambda}_{\max}} \tag{14}$$

where $\bar{\lambda}_{\max}$ is the average of the maximum eigenvalues of the feature covariance matrix over all regions and all images in the dataset.

The average (normalized) coding lengths of five representative color spaces are shown in Figure 7. Among all the 5 color spaces examined, *Lab* has the shortest coding length. Therefore, in the rest of our experiments, input images are first converted to the *Lab* color space.

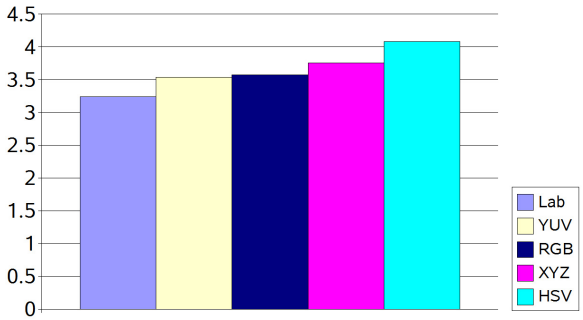


Fig. 7 Average coding length of an image in five representative color spaces.

4.2 Experimental Setup

To quantitatively evaluate the performance of our method, we use four metrics for comparing pairs of image segmentation: the *probabilistic Rand index* (PRI) (Rand (1971)), *variation of information* (VOI) (Meila (2005)), *boundary displacement error* (BDE) (Freixenet et al. (2002)), and the *global F-measure* (Arbelaez (2006)):

1. The *probabilistic Rand index* (PRI) is a classical metric that measures the probability that an arbitrary pair of samples have consistent labels in the two partitions. The PRI metric is in the range $[0, 1]$, with higher values indicating greater similarity between two partitions. When used to adaptively choose ε as described in Section 3.3, we use $\mathrm{PRI}^C \doteq (1 - \mathrm{PRI})$.

2. The *variation of information* (VOI) measures the sum of information loss and information gain between the two clusterings, and thus it roughly measures the extent to which one clustering can explain the other. The VOI metric is nonnegative, with lower values indicating greater similarity.
3. The *boundary displacement error* (BDE) 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.
4. The *global F-measure* (GFM) is the harmonic mean of precision and recall, a pair of complimentary metrics for measuring the accuracy of the boundaries in an image segmentation given the ground truth boundaries. Precision measures the fraction of true boundary pixels in the test segmentation. Recall measures the fraction of ground-truth boundary pixels in the test segmentation. When used to adaptively choose ε, we use $\mathrm{GFM}^C \doteq (1 - \mathrm{GFM})$.

In cases where we have multiple ground-truth segmentations, to compute the PRI, VOI, or BDE measure for a test segmentation, we simply average the results of the metric between the test segmentation and each ground-truth segmentation. To compute the GFM measure from multiple ground-truth segmentations, we apply the same techniques used in Arbelaez et al. (2009), which roughly aggregate the boundary precision and recall over all ground-truth images as an ensemble. With multiple ground-truth segmentations for an image, we can also estimate the human performance w.r.t. these metrics by treating each ground-truth segmentation as a test segmentation and computing the metrics w.r.t. the other ground-truth segmentations.

The adaptive ε scheme in our method relies on the feature vector $\boldsymbol{f}$ used in (12) as follows. The image I is converted to grayscale and its size is rescaled by a set of specific factors. The standard deviation of pixel intensity of each rescaled image constitutes a component of the feature vector. Empirically, we have observed that using four scale factors , i.e., $\boldsymbol{f} \in \mathbb{R}^4$, produces good segmentation results for our algorithm on the BSD database.

The parameters (a_k, b_k, c_k) in the quadratic form in (11) are estimated as follows. We sample $25 \leq \varepsilon \leq 400$ uniformly, in steps of 25 and compute the corresponding $d(S_\varepsilon(I_k), S_g(I_k))$ for each sample. This gives a set $\{(d_{k,n}, \varepsilon_{k,n})\}_{n=1}^{16}$ for an image I_k. We use this set to estimate (a_k, b_k, c_k) by least squares method.

4.3 Results

We quantitatively compare the performance of our method TBES with seven *publicly available* image segmentation methods, namely, *Mean-Shift* (MS) by Comanicu & Meer (2002), *Markov Chain Monte Carlo* (MCMC) by Tu & Zhu (2002), *F&H* by Felzenszwalb & Huttenlocher (2004), *Multiscale NCut* (MNC) by Cour et al. (2005), *Compression-based Texture Merging* (CTM) by Yang et al. (2008), *Ultrametric Contour Maps* (UCM) by Arbelaez et al. (2009), and *Saliency Driven Total Variation* (SDTV) by Donoser et al. (2009), respectively. The user-defined parameters of these methods have been tuned by the training subset of each dataset to achieve the best performance w.r.t. each segmentation index. Then, the performance of each method is evaluated based on the test subset.

Table 2 shows the segmentation accuracy of TBES compared to the human ground truth and the other seven algorithms.[5] In addition to the evaluation of the algorithms, multiple ground truth segmentations in BSD allow us to estimate the human performance w.r.t. these metrics. This was achieved by treating each ground-truth segmentation as a test segmentation and computing the metrics w.r.t. the other ground-truth segmentations. To qualitatively inspect the segmentation, Figure 8 illustrates some representative results.

(a) Animals (b) Buildings (c) Landscape (d) People (e) Water

Fig. 8 Representative segmentation results (in color) of the TBES algorithm on various image categories from BSD. For each image pair, the top is the original input image, and the bottom is the segmentation result where each texture region is rendered by its mean color. The distortion ε was chosen adaptively to optimize PRI.

Among all the algorithms in Table 2, TBES achieves the best performance w.r.t. PRI and VOI. It is also worth noting that there seems to be a large gap in terms of VOI between all the algorithm indices and the human index (e.g., 1.705 for TBES versus 1.163 for human). With respect to BDE and GFM, UCM achieves the best performance, which is mainly due to the fact that UCM was designed to construct texture regions from the hierarchies of (strong) image contours and edges. In this category, TBES still achieves the second best performance, largely exceeding the indices posted by the rest of the algorithms in the literature.

[5] The quantitative performance of several existing algorithms was also evaluated in a recent work (Arbelaez et al. (2009)), which was published roughly at the same time as this paper. The reported results therein generally agree with our findings.

Table 2 Comparison on the BSD using the PRI, VOI, BDE, and GFM indices. For PRI and GFM, higher values indicate better segmentation; for VOI and BDE, lower values indicate better segmentation.

Method	PRI	VOI	BDE	GFM
Human	0.868	1.163	7.983	0.787
TBES	**0.807**	**1.705**	12.681	0.647
MS	0.772	2.004	13.976	0.600
MCMC	0.768	2.261	13.897	0.467
F&H	0.770	2.188	14.057	0.579
MNC	0.742	2.651	13.461	0.590
CTM	0.755	1.897	14.066	0.595
UCM	0.796	1.715	**10.954**	**0.706**
SDTV	0.801	1.790	15.513	0.593

Table 3 A comparison of the efficacy of the individual components of the TBES algorithm. The first row shows the performance of TBES, and each following row corresponds to disabling one component of TBES. $\text{TBES}_{(\varepsilon)}$, $\text{TBSE}_{(w)}$, $\text{TBSE}_{(h)}$, and $\text{TBSE}_{(b)}$ correspond to disabling the code for adaptive choice of epsilon, discounting overlapping windows, hierarchical window sizes, and boundary coding, respectively. For $\text{TBES}_{(\varepsilon)}$, a fixed $\varepsilon = 150$ is chosen. For $\text{TBES}_{(h)}$, a fixed window size $w = 7$ is chosen. The best performance values are highlighted in bold face.

Method	Omitted Component	PRI	VOI	BDE	GFM
TBES	None	**0.807**	**1.705**	**12.681**	**0.647**
$\text{TBES}_{(\varepsilon)}$	Adaptive ε	0.793	1.792	15.020	0.545
$\text{TBES}_{(w)}$	Nonoverlapping Windows	0.790	1.788	13.972	0.597
$\text{TBES}_{(h)}$	Hierarchical Resolutions	0.794	1.743	13.335	0.613
$\text{TBES}_{(b)}$	Boundary Code	0.796	1.775	13.659	0.638

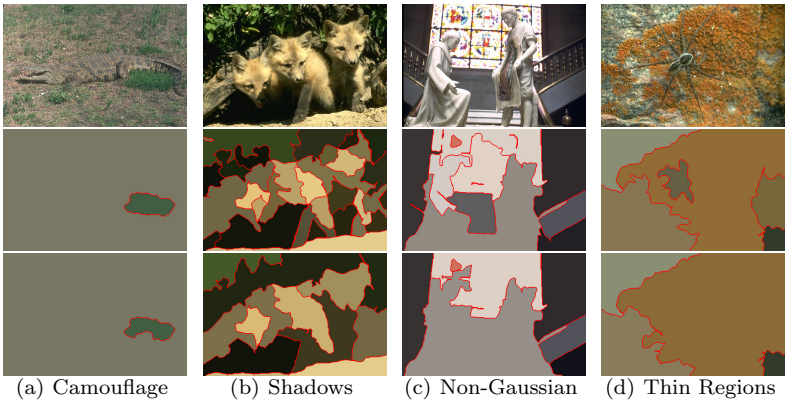
Note that in Table 2, TBES consistently outperforms CTM, on which the fundamental lossy-coding framework of TBES is based. To clarify the contribution of each new TBES component, we further provide an analysis of the efficacy of the components of TBES in a "leave-one-out" comparison. In Table 3, the performance of TBES with certain functions individually disabled is shown. The variations of the code include disabling adaptive choice of epsilon, discounting overlapping windows, hierarchical window sizes, and boundary coding, respectively.

Clearly, as TBES retains the best performance over all four segmentation metrics in Table 3, disabling any segmentation criterion degrades its performance. Since $\text{TBES}_{(\varepsilon)}$ gives the overall worst performance in Table 3, one can conclude that adaptively choosing the distortion level ε is the singular most important heuristic in TBES, which justifies our argument in Section 3.3 that since natural images represent different scene categories with different scales of resolution, no single choice of ε is optimal for all images. Furthermore, it is interesting to observe that all the variations of TBES in Table 3 still achieve better segmentation metrics than the original CTM algorithm in Table 2.

Finally, we briefly discuss a few images on which our method fails to achieve a good segmentation. The examples are shown in Figure 9. The main causes for visually inferior segmentation are camouflage, shadows, non-Gaussian textures, and thin regions:

1. It is easy to see that the texture of animal camouflages is deliberately chosen to be similar to the background texture. The algorithm falls behind humans in this situation, arguably, because human vision can *recognize* the holistic shape and texture of the animals based on experiences.

2. As shades of the same texture may appear very different in images, TBES may break up the regions into more or less the same level of shade.
3. Some patterns in natural images do not follow the Gaussian texture assumption. Examples include geometric patterns such as lines or curves.
4. Thin regions, such as spider's legs, are problematic for TBES for two reasons. First, it has trouble to properly form low-level superpixels used as the initialization of our method. Second, large enough windows which can better capture the statistics of the texture can barely fit into such thin regions. Consequently, texture estimation at these regions is ill-conditioned and unstable.



(a) Camouflage (b) Shadows (c) Non-Gaussian (d) Thin Regions

Fig. 9 Examples from BSD (in color) where TBES algorithm failed obtaining a reasonable segmentation. **Top:** Original input images. **Middle:** Segmentation w.r.t PRI. **Bottom:** Segmentation w.r.t VOI.

To realize whether these problems are unique to our method or are more universal, we have investigated similar problematic cases with the other methods reported here (Comanicu & Meer (2002); Tu & Zhu (2002); Felzenszwalb & Huttenlocher (2004); Cour et al. (2005); Yang et al. (2008); Arbelaez et al. (2009); Donoser et al. (2009)). None of the methods were able to handle camouflage very well. Shadows are challenging for these methods as well. However, we observe that UCM performs relatively better in this case. For geometric patterns, CTM seems to be slightly better than others, but still is an over-segmentation. In the category of thin regions, all algorithms performed very poorly, but mean-shift is better by, for example, roughly picking up some of the spider's legs. It is further worth pointing out an interesting observation about PRI versus VOI that the former prefers over-segmentation and the latter prefers under-segmentation (as shown in Figure 9).

5 Conclusion

We have proposed a novel method for natural image segmentation. The algorithm uses a principled information-theoretic approach to combine cues of image texture and boundaries. In particular, the texture and boundary information of each texture region is encoded using a Gaussian distribution and adaptive chain code, respectively. The

partitioning of an image is achieved by an agglomerative clustering process applied to a hierarchy of decreasing window sizes. Based on the MDL principle, the optimal segmentation of the image is defined as the one that minimizes its total coding length. As the lossy coding length function also depends on a distortion parameter that determines the coarseness of the segmentation, we have further proposed an efficient linear regression method to learn the optimal distortion parameter from a set of training images when provided by the user. Our experiments have validated that the new algorithm outperforms other existing methods in terms of region-based segmentation indices (i.e., PRI and VOI), and is among the top solutions in terms of contour-based segmentation indices (i.e., BDE and GFM). To aid peer evaluation, the implementation of the algorithm and the benchmark scripts have been made available on our website.

References

Arbelaez, P. 2006. Boundary extraction in natural images using ultrametric contour maps. In Workshop on Perceptual Organization in Computer Vision.

Arbelaez, P., M. Maire, C. Fowlkes, & J. Malik 2009. From contours to regions: an empirical evaluation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Comanicu, D., & P. Meer 2002. Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24:603–619.

Cour, T., F. Benezit, & J. Shi 2005. Spectral segmentation with multiscale graph decomposition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Deng, Y., & B. Manjunath 2001. Unsupervised segmentation of color-texture regions in images and video. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(8):800–810.

Donoser, M., M. Urschler, M. Hirzer, & H. Bischof 2009. Saliency driven total variation segmentation. in Proceedings of the International Conference on Computer Vision.

Duda, R., P. Hart, & D. Stork 2001. Pattern Classification. Wiley, 2 edition.

Efros, A., & T. Leung 1999. Texture synthesis by non-parametric sampling. In Proceedings of the International Conference on Computer Vision.

Elder, J., & S. Zucker 1996. Computing contour closures. In Proceedings of the European Conference on Computer Vision.

Felzenszwalb, P., & D. Huttenlocher 2004. Efficient graph-based image segmentation. International Journal of Computer Vision, 59(2):167–181.

Freixenet, J., X. Munoz, D. Raba, J. Marti, & X. Cuff 2002. Yet another survey on image segmentation. in Proceedings of European Conference on Computer Vision, 408–422.

Gevers, T., & A. Smeulders 1997. Combining region splitting and edge detection through guided Delaunay image subdivision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Haralick, R., & L. Shapiro 1985. Image segmentation techniques. Computer Vision, Graphics, and Image Processing, 29(1):100–132.

Jain, A. 1989. Fundamentals of Digital Image Processing. Prentice Hall.

Kim, J., J. Fisher, A. Yezzi, M. Cetin, & A. Willsky 2005. A nonparametric statistical method for image segmentation using information theory and curve evolution. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(10):1486–1502.

Kim, T., K. Lee, & S. Lee 2010. Learning full pairwise affinities for spectral segmentation. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Kurita, T. 1995. An efficient clustering algorithm for region merging. IEICE Transactions of Information and Systems, E78-D(12):1546–1551.

Levina, E., & P. Bickel 2006. Texture synthesis and non-parametric resampling of random fields. Annals of Statistics, 34(4):1751–1773.

Liu, Y., & B. Zalik 2005. Efficient chain code with Huffman coding. Pattern Recognition, 38(4):553–557.

Ma, Y., H. Derksen, W. Hong, & J. Wright 2007. Segmentation of multivariate mixed data via lossy coding and compression. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(9):1546–1562.

Malik, J., S. Belongie, T. Leung, & J. Shi 2001. Contour and texture analysis for image segmentation. International Journal of Computer Vision, 43(1):7–27.

Martin, D., C. Fowlkes, D. Tal, & J. Malik 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In Proceedings of the International Conference on Computer Vision.

Meila, M. 2005. Comparing clusterings: an axiomatic view. In Proceedings of the International Conference on Machine Learning.

Mori, G., X. Ren, A. Efros, & J. Malik 2004. Recovering human body configurations: combining segmentation and recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Rand, W. 1971. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association, 66(336):846–850.

Rao, S., H. Mobahi, A. Yang, S. Sastry, & Y. Ma 2009. Natural image segmentation with adaptive texture and boundary encoding. In Proceedings of the Asian Conference on Computer Vision, volume 1, pages 135–146.

Ren, X., C. Fowlkes, & J. Malik 2005. Scale-invariant contour completion using condition random fields. In Proceedings of the International Conference on Computer Vision.

Ren, X., C. Fowlkes, & J. Malik 2008. Learning probabilistic models for contour completion in natural images. International Journal of Computer Vision, 77:47–63.

Tremeau, A., & N. Borel 1997. A region growing and merging algorithm to color segmentation. Pattern Recognition 30(7):1191–1204.

Tu, Z., & S. Zhu 2002. Image segmentation by data-driven Markov Chain Monte Carlo. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5):657–673.

Varma, M., & A. Zisserman 2003. Texture classification: are filter banks necessary? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Wang, J. Y. Jia, X. Hua, C. Zhang, & L. Quan 2008. Normalized tree partitioning for image segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Yang, A., J. Wright, Y. Ma, & S. Sastry 2008. Unsupervised segmentation of natural images via lossy data compression. Computer Vision and Image Understanding, 110(2):212–225.

Yu, S. 2005. Segmentation induced by scale invariance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Zhu, Q., G. Song, & J. Shi 2007. Untangling cycles for contour grouping. In Proceedings of the International Conference on Computer Vision.