ADAPTIVE SUPERPIXEL SEGMENTATION AGGREGATING LOCAL CONTOUR AND TEXTURE FEATURES

Xiaolin Xiao, Yue-Jiao Gong, Yicong Zhou

Department of Computer and Information Science, University of Macau
shellyxiaolin@gmail.com, gongyuejiao@gmail.com, yicongzhou@umac.mo

ABSTRACT

Superpixel segmentation targets at grouping pixels in an image into atomic regions that align well with the natural object boundaries. In this paper, we propose a novel superpixel segmentation method based on an iterative and adaptive clustering algorithm that embraces color, contour, texture, and spatial features together. The algorithm adjusts the weights of different features automatically in a content-aware way, so as to fit the requirements of various image instances. More specifically, in each iteration, the weights in the aggregation function are adjusted according to the discriminabilities of features in the current working scenario. This way, the algorithm not only possesses improved robustness but also relieves the burden of setting the parameters manually. Experimental verification shows that the algorithm outperforms existing peer algorithms in terms of commonly used evaluation metrics, while using a low computational cost.

Index Terms— Superpixel, adaption, contour, texture

1. INTRODUCTION

Superpixel segmentation produces atomic regions of pixels (namely, the superpixels) that are consistent with human perception. Unlike the traditional rigid pixel representation of images, superpixels provide visually meaningful entities, which can further be utilized as inputs for mid- or high-level computer vision tasks. The prominent advantage of using superpixels instead of pixels is the reduction in computational cost for subsequent processing.

Existing superpixel algorithms can be generally classified into the following two categories. The first one is the graph-cut-based methods [1–10], which consider the image as a graph containing vertices and edges. Each vertex corresponds to a pixel in the image, while edges are defined among the adjacent vertices. Starting with an image as a graph, the image is then partitioned into a few disjoint sub-graphs (superpixels) by minimizing the cut on edges. The second category includes the gradient-based methods [11–18], where pixels are clustered along the direction that the gradients change most quickly in each iteration, and finally they are grouped into superpixels when the stopping criterion is achieved.

This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/016/2015/A1 and by the Research Committee at University of Macau under Grants MYRG2014-00003-FST and MYRG2016-00123-FST.

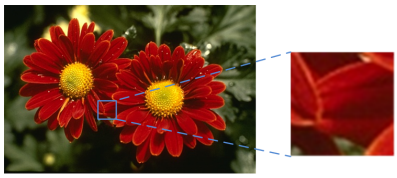

Fig. 1. Region that is hard to cluster merely based on color.

According to different requirements in diverse application scenarios, the design principles of existing superpixel algorithms vary, emphasizing on better boundary adherence [10, 17] or superpixel regularity [15], etc. To the best of our knowledge, none of the existing superpixel algorithms has consistently good performance in handling different types of images, since the parameters and operations in these algorithms are fixed for all input images. To solve this problem, we proposed an adaptive clustering-based superpixel segmentation algorithm (ACS).

Our motivation of designing ACS comes from the following observations: (1) the merely use of color difference is inadequate to produce locally meaningful and compact entities, especially in low contrast regions (see Fig. 1); (2) natural images vary significantly in their contents, so that different features (color, sptial, contour, texture, etc.) have different discriminabilities. For example, untextured images can be easily segmented by emphasizing the contour feature, while the textured images should rely more on the texture feature.

Compared with previous works, the novelties of ACS lie in: (1) perception consistent: we adopt a color difference measure based on a more perception consistent standard; (2) content adaptability: the aggregating weights of features are automatically adapted according to their discriminabilities on different images; and (3) simplicity and efficiency: it is simple and intuitive to add new features on existing superpixel algorithms, but it is a nontrivial work (always tedious and time-consuming) to set reasonable weights. ACS relieves the burden since the weights are automatically set.

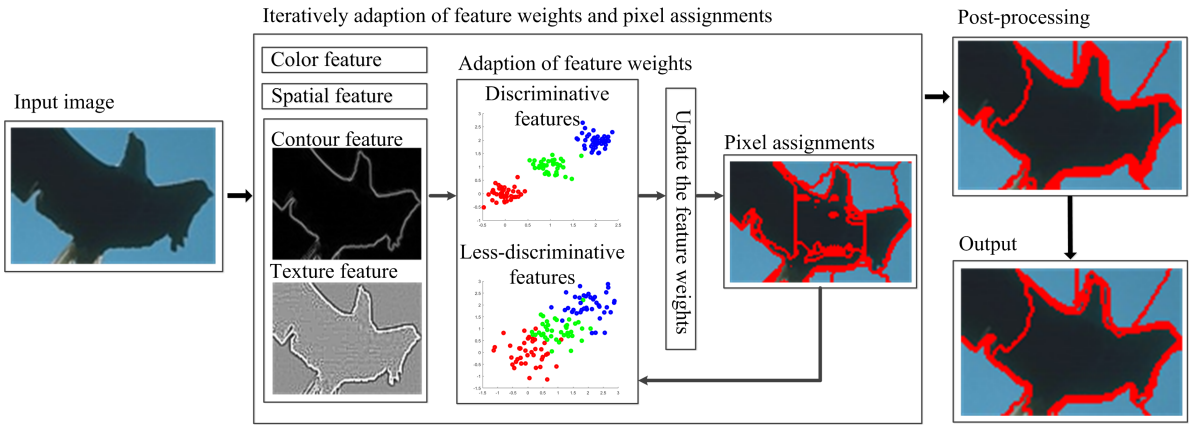

Fig. 2. Flow chart of the proposed ACS algorithm.

2. ADAPTIVE CLUSTERING-BASED SUPERPIXEL SEGMENTATION (ACS)

The framework of ACS is illustrated in Fig. 2. Generally speaking, ACS is based on a simplified linear clustering that incorporates color, spatial, contour and texture features into an adaptive distance measure. In each iteration, ACS can adjust the weights of different features regarding to their discriminabilities. Besides, we use the post-processing method in [17] to get the final segmentation to guarantee the connectivities within superpixels.

2.1. The features and distance measure

CIELCH is a color space specified by the International Commission on Illumination(CIE). Many start-of-the-art superpixel algorithms [16, 17] measure the color difference in CIELAB space. However, since the CIELAB color space is not completely uniform, especially in some saturated regions, we propose to represent each pixel in the CIELCH color space, where L, C, and H represent lightness, chroma, and hue, respectively. When converting the color representation from a CIELAB vector $[l, a, b]^T$ to a CIELCH vector $[l, c, h]^T$, the lightness stays the same, while c and h are the polar coordinates of a and b.

The corresponding color difference d_c is calculated using the CIE94 measure:

$$d_c = \sqrt{(\frac{\Delta l}{k_l s_l})^2 + (\frac{\Delta c}{k_c s_c})^2 + (\frac{\Delta h}{k_h s_h})^2}, \quad (1)$$

where Δl, Δc, and Δh represent the differences of l, c, and h between two pixels, k_l, s_l, k_c, and k_h are constants, while s_c and s_h can be computed regarding to the compared pixel values.

Spatial distance. The spatial distance d_s is calculated using the normalized Euclidean distance between pixel coordinates.

Image gradient measures the directional intensity changes in an image, and its magnitude is achieved by the square root of the sum of the squared directional intensity changes. In ACS, we represent the gradient magnitude by g, and calculate it in l domain since human eyes are very sensitive to lightness changes. A threshold is adopted to remove the side effect of fake contours. The gradient difference is denoted as d_g.

Weber local descriptor [19] is designed to extract the local salient patterns in an image. We exploit the differential excitation part of WLD which is denoted as u. The feature is computed as the ratio over two terms: the relative lightness differences of a current pixel against its neighbors and the lightness of the current pixel. Further, an $\arctan$ mapping is used to prevent u from changing sharply as the input changes. The texture difference is denoted as d_u.

Combining the above color, contour, texture and spatial(denoted by x and y, represent the spatial position of pixel) features into a seven dimension feature space, the pixels can be represented by $\boldsymbol{p} = [l, c, h, x, y, g, u]^T$, where the distance D between pixels is measured as:

$$D = \sqrt{w_c (d_c)^2 + w_s (d_s)^2 + w_g (d_g)^2 + w_u (d_u)^2}, \quad (2)$$

where w_c, w_s, w_g, and w_u are weights for d_c, d_s, d_g, and d_u, respectively.

2.2. Iteratively adapt feature weights

In this section, we will present a measure to compare the discriminabilities of different features [20] and then apply the measure to automatically adapt the feature weights at each iteration. The principle for feature weighting is to assign a larger weight to a feature that has a smaller sum of the within cluster distances, while assigning a smaller weight to a feature that has a larger sum of the within cluster distances.

Firstly, the weights of all features are equally initialized and used for pixel assignments. Then, in each iteration, we measure the discriminability of each feature according to the

Algorithm 1: Adaptive Clustering-based Superpixel Segmentation (ACS)

Input : Input image $\boldsymbol{I}$, Superpixel number k.
Output: Label matrix $\boldsymbol{L}$ of the input image .

Represent each pixel $\boldsymbol{p}$ by $[l, c, h, x, y, g, u]^T$,
Select k initial centers on the image grid,
Initialize $L(\boldsymbol{p}) = 0$ for each pixel,
Initialize distance $d(\boldsymbol{p}) = \infty$ recording the difference between each pixel and its nearest center,
Initialize w_c,w_s,w_g and w_u to 1,
repeat
  for each center $\boldsymbol{c}_i (i = 1, 2, ...k)$ **do**
    for each pixel $\boldsymbol{p}_q (q = 1, 2, ...n)$ located in the neighborhood of $\boldsymbol{c}_i$ **do**
      Calculate the distance D between $\boldsymbol{p}_q$ and $\boldsymbol{c}_i$,
      if $D < d(\boldsymbol{p}_q)$ **then**
        $d(\boldsymbol{p}_q) = D$,
        $L(\boldsymbol{p}_q) = i$,
      end
    end
  end
  Update the cluster centers to the mean vectors of each cluster,
  for each feature j **do**
    Calculate the within cluster variance SW_j using Eq. (3),
  end
  for each feature j **do**
    Calculate the weight of feature w_j using Eq. (4),
  end
until iteration is stopped;
Merge unconnected superpixels to their most similar neighbors.

sum of the within cluster distances SW_j for feature j using Eq. (3). Note that the larger SW_j, the smaller discriminability for feature j.

$$SW_j = \sum_{i=1}^{k} \sum_{q=1}^{n} \hat{u}_{\boldsymbol{p}_q, \boldsymbol{c}_i} d_j(\boldsymbol{p}_q \rightarrow \boldsymbol{c}_i), \tag{3}$$

where the subscript $j \in fset = \{c, s, g, u\}$, representing the feature set of color, spatial, contour, and texture; k is the number of superpixels; n stands for the number of pixels; $\hat{u}_{\boldsymbol{p}_q, \boldsymbol{c}_i}$ is a binary variable to indicate whether a pixel $\boldsymbol{p}_q (q = 1, 2, ...n)$ belongs to a cluster center $\boldsymbol{c}_i (i = 1, 2, ...k)$; and $d_j(\boldsymbol{p}_q \rightarrow \boldsymbol{c}_i)$ measures the distance of pixel $\boldsymbol{p}_q$ to center $\boldsymbol{c}_i$ on feature j.

The weight of feature j is then adjusted according to its relative feature discriminabilities over all features as:

$$w_j = \frac{1}{\sum_{t \in fset} \left[\frac{SW_j}{SW_t} \right]^{\frac{1}{\beta - 1}}}, \tag{4}$$

where β is a constant set to 9 as recommended in [20]. Then, the weights of different feature distances in the aggregating function are updated accordingly in next iteration. The stopping criterion is directly achieved by a fixed number of iteration T, in our experiment, $T = 10$. The implementation of ACS is summarized in Algorithm 1.

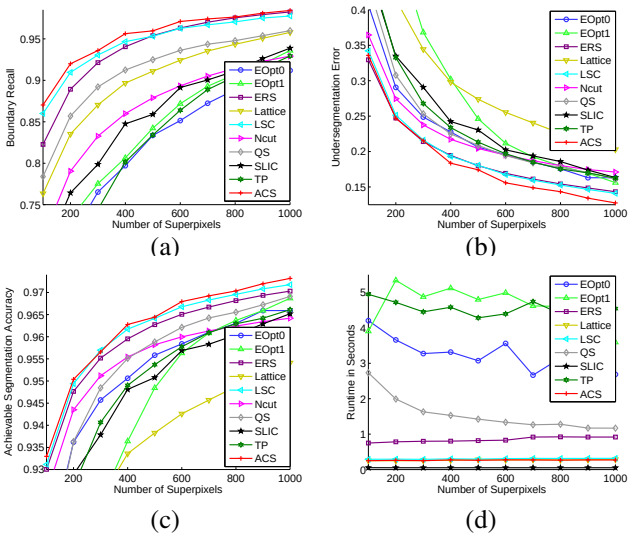

Fig. 3. Quantitative evaluation of different superpixel algorithms. (a) Boundary Recall. (b) Undersegmentation Error. (c) Achievable Segmentation Accuracy. (d) Runtime.

3. EXPERIMENTS

This section performs experiments to evaluate the proposed ACS algorithm. We use the Berkeley segmentation database [21], which consisting of 300 test images with human-labeled groundtruth segments. We compare ACS with nine state-of-the-art superpixel methods: Ncut [1], QS [14], Lattice [5], TP [15], EOpt0 and EOpt1 [9], ERS [10], SLIC [16], and LSC [17]. For ACS, the parameters w_c, w_s, w_g and w_u are initialized and adapted in each iteration. All algorithms are tested in the same computational environment.

3.1. Evaluation metrics

To better evaluate the proposed algorithm, we use the following metrics used in [16, 17]: **Boundary Recall (BR)** measures boundary adherence by the percentage of natural boundaries recovered by superpixel boundaries. **Undersegmentation Error (UE)** measures the "unsegmented" superpixels which overlaps with multiple natural objects. **Achievable Segmentation Accuracy (ASA)** measures the maximum segmentation performance that can be achieved using superpixels as atomic units. **Runtime** is used to evaluate algorithm efficiency.

3.2. Quantitative comparisons

Fig. 3 illustrates the quantitative comparison results of all superpixel algorithms in terms of different metrics. The number of superpixels k, in each test image, is set to 100, 200, ...1000, respectively. Considering the BR metric, as shown in Fig. 3(a), we can observe that ACS gains the best BR, while EOpt0 yields the worst boundary overlap. In Fig. 3(b), ACS

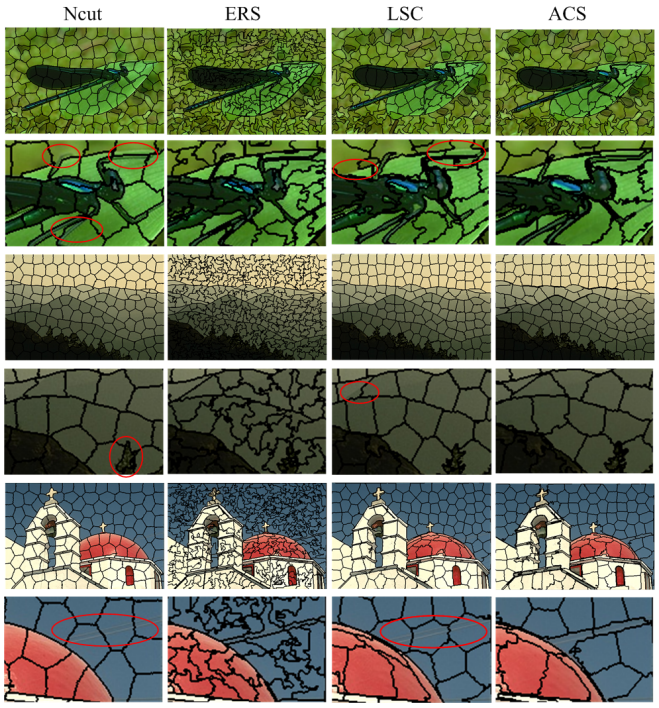

Fig. 4. Visualization of different superpixel algorithms when k=200.

gets the best UE when k is larger than 200, and it achieves more significant improvements as k increases. Lattice generates superpixels by adding vertical or horizontal paths on the image, thus, the superpixel boundaries are prone to winging across the object boundaries. These results verify that Lattice performs the worst in terms of UE. As shown in Fig. 3(c), ACS and LSC achieve comparable performance using the ASA metric, while the results of ACS are slightly better except when k=300. As for the runtime illustrated in Fig. 3(d), ACS is also competitive. It ranks as the third fastest superpixel algorithm, following SLIC and Lattice. Note that the runtime of Ncut is not plotted in Fig. 3(d) because the value is too large. In summary, ACS performs equally or better than the other state-of-the-art algorithms in all metrics.

3.3. Visual results

We compare the visual results of ACS, Ncut, ERS, and LSC when k= 200. Ncut is chosen because it is a representative graph-based superpixel algorithm that has wide applications, while ERS and LSC are the two best performed algorithms in quantitative comparisons except ACS. It can be seen from Fig. 4 that Ncut, LSC, and ACS generate superpixels with regular shapes and sizes. However, Ncut performs unsatisfactorily in capturing the small and dark contours. LSC is unable to capture boundaries when the color contrast is low, which shows that merely using the color-based difference measure is inadequate for superpixel segmentation. Besides, the superpixel boundaries of ERS wing across object boundaries and are not appealing for visualization. The red ellipses on Fig. 4 highlight the details where other algorithms are inadequate compared to ACS.

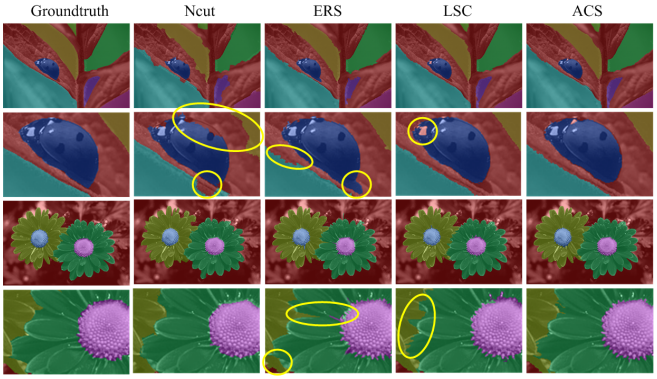

Fig. 5. Visualization of the maximum segmentation performance using different superpixel algorithms when k=400.

3.4. Investigation on the pre-processing performance

Superpixels provide visually meaningful entities at lower computational cost for subsequent processing than pixels. Thus, they are widely used in applications such as image segmentation and image understanding. The baseline of exploiting superpixels lies in that the performance of subsequent processing should not decrease too much using superpixels instead of pixels. We apply an ideal classifier that assign each superpixel to the groundtruth segment that the superpixel overlaps the most, and visualize the ideal segmentation results in Fig. 5. We compare ACS with Ncut, ERS and LSC when k=400. The results verify the advantage of using ACS as a pre-processing step for other computer vision tasks. The yellow ellipses on Fig. 4 highlight the details where other algorithms are inadequate compared to ACS.

4. CONCLUSION

In this paper, we presented a content-aware algorithm for superpixel segmentation. The algorithm is based on the adaptive combination of low-level features including color, contour, texture and spatial position. The color distance is formulated by using a measure that is consistent with human perceived pixel difference. The contour and texture features are derived from image gradient and WLD in lightness domain, respectively. More importantly, we adjust weights of different features in an iterative and adaptive way according to image contents, resulting in a more accurate and reasonable distance measure to discriminate pixels in natural images. Experimental results have demonstrated the advantages of our algorithm over other state-of-the-art superpixel algorithms.

5. REFERENCES

[1] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[2] Xiaofeng Ren and Jitendra Malik, "Learning a classification model for segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.* IEEE, 2003, pp. 10–17.

[3] Xuming He, Richard S Zemel, and Debajyoti Ray, "Learning and incorporating top-down cues in image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, pp. 338–351. Springer, 2006.

[4] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Ben Ward, and Philip HS Torr, "Videotrace: rapid interactive scene modelling from video," *ACM Trans. Graphics*, vol. 26, no. 3, pp. 86, 2007.

[5] Andrew P Moore, JD Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones, "Superpixel lattices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.* IEEE, 2008, pp. 1–8.

[6] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, 2001.

[7] Yuri Boykov and Vladimir Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, 2004.

[8] Vladimir Kolmogorov and Ramin Zabin, "What energy functions can be minimized via graph cuts?," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 147–159, 2004.

[9] Olga Veksler, Yuri Boykov, and Paria Mehrani, "Superpixels and supervoxels in an energy optimization framework," in *Proc. Eur. Conf. Comput. Vis.*, pp. 211–224. Springer, 2010.

[10] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa, "Entropy rate superpixel segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.* IEEE, 2011, pp. 2097–2104.

[11] Yizong Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, 1995.

[12] Dorin Comaniciu and Peter Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[13] Yaser Ajmal Sheikh, Erum Arif Khan, and Takeo Kanade, "Mode-seeking by medoidshifts," in *Proc. IEEE Int. Conf. Comput. Vis.* IEEE, 2007, pp. 1–8.

[14] Andrea Vedaldi and Stefano Soatto, "Quick shift and kernel methods for mode seeking," in *Proc. Eur. Conf. Comput. Vis.*, pp. 705–718. Springer, 2008.

[15] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2290–2297, 2009.

[16] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, 2012.

[17] Zhengqin Li and Jiansheng Chen, "Superpixel segmentation using linear spectral clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.* IEEE, 2015, pp. 1356–1363.

[18] Yue-Jiao Gong, Yicong Zhou, and Xinglin Zhang, "A superpixel segmentation algorithm based on differential evolution," in *Multimedia and Expo (ICME), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[19] Jie Chen, Shiguang Shan, Chu He, Guoying Zhao, Matti Pietikäinen, Xilin Chen, and Wen Gao, "Wld: A robust local image descriptor," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1705–1720, 2010.

[20] Joshua Zhexue Huang, Michael K Ng, Hongqiang Rong, and Zichen Li, "Automated variable weighting in k-means type clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 657–668, 2005.

[21] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Comput. Vis.* IEEE, 2001, vol. 2, pp. 416–423.