Image Segmentation by Unsupervised Sparse Clustering

Byoung-Ki Jeon, Yun-Beom Jung, and Ki-Sang Hong
Electrical and Computer Engineering Division, POSTECH
San 31 Hyoja-Dong, Pohang, 790-784, Korea
{standard,jyb,hongks}@postech.ac.kr

Abstract

In this paper, we present a novel solution of image segmentation based on positiveness by regarding the segmentation as one of the graph-theoretic clustering problems. On the contrary to spectral clustering methods using eigenvectors, the proposed method tries to find an additive combination of positive components from an originally positive data-driven matrix. By using the positiveness constraint, we obtain sparsely clustered results which are closely related to human perception and thus we call this method sparse clustering. The proposed method adopts a binary tree structure and solves a model selection problem by automatically determining the number of clusters using intra- and inter-cluster measures. We tested our method with various kinds of data such as points, gray-scale, color, and texture images. Experimental results show that the proposed method provides very successful and encouraging segmentations.

1. Introduction

Image segmentation is used to distinguish objects from their background or to partition an image into related regions. Among various approaches to image segmentation, the proposed method in this study belongs to a *graph theoretic clustering method*. In a graph-theoretic framework, we consider that the original data, such as pixel values in the given image, constitute a weighted undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Each node in $\mathbf{V}$ is a feature vector calculated from each original datum, and an edge in $\mathbf{E}$ is linked between every pair of nodes. The edge weight $w(i, j)$ is a function of the distance between nodes i and j in feature space and it is mainly defined as larger as the distance gets smaller. If a matrix is constructed where each element is the edge weight between a pair of nodes, then it will be a square symmetric matrix representing adjacencies between graph nodes - here, we call it the *affinity matrix* - and having the same number of rows (or columns) as the number of nodes in $\mathbf{V}$.

According to the *spectral theorem* (also called *principal axis theorem*) which is a theoretical basis of spectral clustering, a real symmetric matrix, such as the affinity matrix, can be factored into $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T = \lambda_1\mathbf{q}_1\mathbf{q}_1^T + \lambda_2\mathbf{q}_2\mathbf{q}_2^T + \cdots + \lambda_n\mathbf{q}_n\mathbf{q}_n^T$ with the orthonormal eigenvectors $\mathbf{q}_i$'s in $\mathbf{Q}$ and the eigenvalues λ_i's in $\mathbf{\Lambda}$ [1]. Since the affinity matrix actually represents *similarities* between nodes, it cannot have any negative entries. For the matrix with only positive elements, we can observe that its largest eigenvector has only positive entries but the next eigenvectors can have negative entries because of the orthogonality constraints between eigenvectors. In other words, spectral clustering can lead to a subtractive combination of negative components from the originally positive affinity matrix and this can cause complex cancellations - they prevent us from obtaining intuitive meanings from the factorized components.

Several previous works have been proposed for factorization using positiveness or non-negativeness constraints [2] [3] [4]. There are roughly two reasons why they are interested in positiveness. The first one is that the positiveness makes intuitive interpretation possible. If we apply one of the spectral clustering methods to originally positive data such as pixel values or the affinity matrix, an interpretation of the eigenvectors with negative entries is often impossible. Further, these negative entries can cause a subtractive combination with complex cancellations. The second reason is that the positiveness property can be a key to the ways how the human brain perceives objects - there is a psychological and physiological evidence for parts-based representations in brain [3]. Since the positiveness allows only additive, not subtractive, combinations, it can lead to parts-based representations and eventually provide *sparse codings* where many entries of each factor are zero [5] [6]. The sparse coding represents an observation vector using only a few sparse basis vectors and is different from *distributed coding*, such as *principal component analysis (PCA)*, which represents an observation using most of the basis vectors which are holistic.

By incorporating preceding properties into the clustering framework, we propose a novel clustering method which factorizes the originally positive affinity matrix under the positiveness constraint using the *positive tensor factorization (PTF)* method [4]. The proposed method provides sparsely clustered results based on positiveness and therefore we call it *sparse clustering*. Since the image segmentation problem can be regarded as inherently hierarchical - human beings first segment an image into several large regions and then successively segment those regions in details, the proposed method adopts a *hierarchical structure* based on the *binary tree*. In this structure, we use both *intra- and inter-cluster measures* to determine whether the current cluster should be factorized or not, and solve the *model selection* problem by automatically determining the number of clusters based on these two measures. If we only use a single measure, then the hierarchical structure may cause over- or under-segmentation.

2. Positiveness-based Factorization

To factorize the affinity matrix, we use the PTF method which factorizes the tensor of order d, $\mathbf{T}_{i_1,\cdots,i_d}$ with positive entries into N positive components, $\mathbf{C}^{(j)}_{i_j,n}$, as follows:

$$\mathbf{T}_{i_1,\cdots,i_d} = \sum_{n=1}^{N} \mathbf{C}^{(1)}_{i_1,n}\mathbf{C}^{(2)}_{i_2,n}\cdots\mathbf{C}^{(d)}_{i_d,n}, \tag{1}$$

such that the *reconstruction error*,

$$RE = \sum_{i_1,\cdots,i_d}\left(\mathbf{T}_{i_1,\cdots,i_d} - \sum_{n=1}^{N} \mathbf{C}^{(1)}_{i_1,n}\mathbf{C}^{(2)}_{i_2,n}\cdots\mathbf{C}^{(d)}_{i_d,n}\right)^2 \tag{2}$$

is minimized. The update rule is obtained by taking derivatives of Eqn. (2) with respect to each $\mathbf{C}^{(j)}_{i_j,n}$, finding the point where the derivative vanishes, and defining a positive scaling factor based on the information around that point. For detail explanation, please refer to [4].

2.1. Features and Affinity Matrix

In this paper, we try to solve the image segmentation problem in a graph-theoretic framework where the given image data constitutes a weighted undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. To segment an image means to partition the graph $\mathbf{G}$ and this is performed by factorizing an affinity matrix which measures similarities between all of the graph nodes in $\mathbf{V}$. Therefore, the affinity matrix is a square symmetric matrix with the same number of rows (or columns) as the number of nodes and defined using not only feature similarities but also spatial adjacencies as follows:

$$A_{ij} = e^{-\frac{||\mathbf{F}(\mathbf{X}_i)-\mathbf{F}(\mathbf{X}_j)||^2}{\sigma_F^2}} \times \begin{cases} e^{-\frac{||\mathbf{X}_i-\mathbf{X}_j||^2}{\sigma_X^2}} & \text{if } ||\mathbf{X}_i - \mathbf{X}_j|| < r \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathbf{X}_i$ is the spatial position vector of node i and $\mathbf{F}(\mathbf{X}_i)$ is the feature vector at $\mathbf{X}_i$ which has different types according to the input data.

2.2. Factorization of Affinity Matrix

Based on the PTF algorithm, considering how the affinity matrix can be factorized we can get the segmentation information from the factorized components. The affinity matrix $\mathbf{A}$ can be regarded as a tensor of order 2 ($d = 2$) and it needs to be factorized into two positive components ($N = 2$) since the proposed method adopts the hierarchical binary tree structure to recursively bi-segment images. If y and x are indices denoting rows and columns of the given image, an element of the affinity matrix $\mathbf{A}$ can be written with the tensor notation as Eqn. (1):

$$\begin{aligned}\mathbf{A}_{(y,x),(y,x)} = \mathbf{A}_{I,J} = \mathbf{A}_{J,I} &= \sum_{n=1}^{2} \mathbf{C}^{(1)}_{I,n}\mathbf{C}^{(2)}_{J,n} && (4)\\ &= \mathbf{C}^{(1)}_{I,1}\mathbf{C}^{(2)}_{J,1} + \mathbf{C}^{(1)}_{I,2}\mathbf{C}^{(2)}_{J,2} \\ &= {\mathbf{C}^{(1)}_{I,1}}^2 + {\mathbf{C}^{(1)}_{I,2}}^2, && (5)\end{aligned}$$

where $\mathbf{C}^{(1)}_{I,n} = \mathbf{C}^{(2)}_{J,n}$ due to the symmetry of the affinity matrix and thus we need to update only one matrix, $\mathbf{C}^{(1)}$. It has P rows and N columns where P is the number of pixels (or graph nodes) and N is the number of positive components.

In our binary tree structure, the matrix $\mathbf{C}^{(1)}$ has two columns ($N = 2$) and they can be allocated to the left and right nodes of the binary tree. From $\mathbf{C}^{(1)}$, we can obtain the segmentation information as follows. $\mathbf{C}^{(1)}$ has P rows where each row corresponds to each pixel position and has two non-negative values. After convergence, one of them is probable to be a somewhat large positive value and the other to be a negligible positive value or zero. Therefore, if the value of the first column is larger than or equal to that of the second column, then the corresponding pixel is assigned to the left node, and vice versa - this means bi-partitioning of the graph or bi-segmentation of the image. This procedure is a kind of *discretization* of continuous factorized components and we call this the *comparison-based method*. The discretization problem is also discussed in the Ncut paper [7], which bi-partitions the graph by selecting the thresholding point such that the resulting partition of the eigenvector has the best normalized cut value. Like this, we can bi-partition the cluster by thresholding a factorized component which corresponds to one of two columns of $\mathbf{C}^{(1)}$, say, the first column. The threshold value is determined by finding the splitting point such that the resulting partition has the smallest inter-cluster score, and we call this the *thresholding-based method*. In our implementation, the two proposed methods show very similar discretized results. However, the comparison-based method may provide unstable segmentation boundaries if the compared values are slightly different. Therefore, for the experimental results in Section 4, we adopt the thresholding-based discretization method.

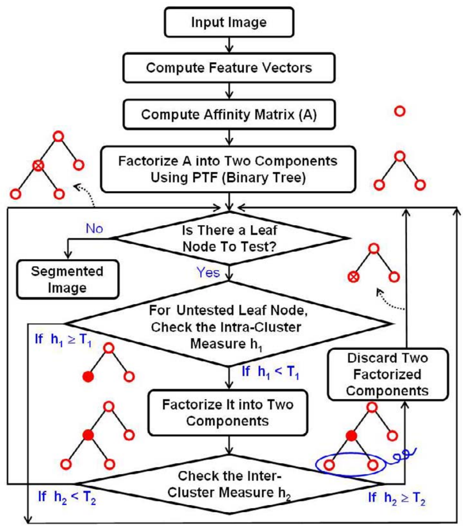
Input Image
Compute Feature Vectors
Compute Affinity Matrix (A)
Factorize A into Two Components Using PTF (Binary Tree)
No
Is There a Leaf Node To Test?
Segmented Image
Yes
For Untested Leaf Node, Check the Intra-Cluster Measure h1
If h1 ≥ T1
If h1 < T1
Discard Two Factorized Components
Factorize It into Two Components
Check the Inter-Cluster Measure h2
If h2 < T2
If h2 ≥ T2

Figure 1. Overall flow of unsupervised sparse clustering.

3. Unsupervised Sparse Clustering

In this section, we explain the main procedure and structure of the proposed method, including intra- and inter-cluster measures. Since our method clusters points based on PTF which provides sparse codings and determines the number of clusters based on the hierarchical structure without user interaction, we call it *unsupervised sparse clustering*.

3.1. Overall Flow and Hierarchical Structure

Figure 1 shows the overall flow of the proposed method. We first compute feature vectors according to the input data types as explained in Section 2.1. Next, we compute the affinity matrix from them and factorize it into two positive components using PTF. Since we use the binary tree, which is one hierarchical structure, we need to bi-factorize the affinity matrix. For each untested leaf node that is one of the factorized components, we compute the intra-cluster measure h_1, and if it is larger than or equal to a threshold T_1, then we go to the next untested leaf node. If it is smaller than T_1, we factorize that node into two positive components, and compute the inter-cluster measure h_2 between them. If it is larger than or equal to another threshold T_2, then we remove these factorized components and go to the next leaf node, and if not, we accept the components as new left and right leaf nodes and also go to the next leaf node. This procedure is iterated until there is no leaf node to test, and eventually leads to hierarchically segmented results.

3.2. Intra- and Inter-Cluster Measures

The proposed method automatically determines the number of clusters by incorporating two measures into the binary tree structure. The first one is the *intra-cluster measure* which determines if the current cluster should be factorized or not, and the second one is the *inter-cluster measure* which determines if the factorization by the intra-cluster measure is acceptable or not by checking separability between the two factorized components.

Spectral graph theory provides a measure of tightness of clusters called the *Cheeger constant* [8]. If there is a partition of $(\mathcal{I}, \bar{\mathcal{I}})$ for the current cluster C_i, the Cheeger constant $\phi(C_i)$ is defined as

$$\phi(C_i) = \min_{\mathcal{I}} \frac{\sum_{j\in\mathcal{I}, k\in\bar{\mathcal{I}}} A_{jk}^{(i)}}{\min\{\mathrm{vol}(\mathcal{I}), \mathrm{vol}(\bar{\mathcal{I}})\}}, \tag{6}$$

where $A_{jk}^{(i)}$ is an affinity value between nodes j and k in the cluster C_i. If the weight of the edges across the partition is small and each of the partitions has a moderately large volume, then the Cheeger constant is small - this means the cluster C_i is not tight enough and can be factorized more. However, to compute $\phi(C_i)$ is to find the optimal partition of $(\mathcal{I}, \bar{\mathcal{I}})$, and this is a *NP-hard* problem. Therefore, we actually use the *eigengap* which is closely related to the Cheeger constant [9] [10]. The eigengap of $\delta(C_i)$ for the cluster C_i is defined as $1 - \lambda_2/\lambda_1$, where λ_1 and λ_2 are the largest and the second largest eigenvalues of the affinity matrix for the pixels in C_i. The eigengap provides bounds of the Cheeger constant by the inequality

$$\frac{\phi^2}{2} \le \delta \le 2\phi, \tag{7}$$

and therefore we define the intra-cluster measure of the cluster C_i as the eigengap which is related to the lower-bound of the Cheeger constant ($\phi \ge \delta/2$) as follows

$$Intra(C_i) = h_1(C_i) = \delta(C_i). \tag{8}$$

If the intra-cluster measure h_1 is smaller than the threshold T_1, then we regard the cluster as not tight enough and factorize it more.

In case we only consider the intra-cluster information, the proposed method may provide over- or under-segmented results - if the threshold T_1 is selected to be large for factorizing a cluster which should be factorized, the algorithm may also factorize another cluster which should not be factorized and eventually may over-estimate the number of clusters, and vice versa. To prevent this, we adopt another measure called the inter-cluster measure h_2 to check the separability between the two sub-clusters previously factorized by the intra-cluster measure. It is defined as

$$Inter(C_j, C_k) = h_2(C_j, C_k) = \sum_{j\in C_j}\sum_{k\in C_k} \frac{A_{jk}^2}{d_j d_k}, \tag{9}$$

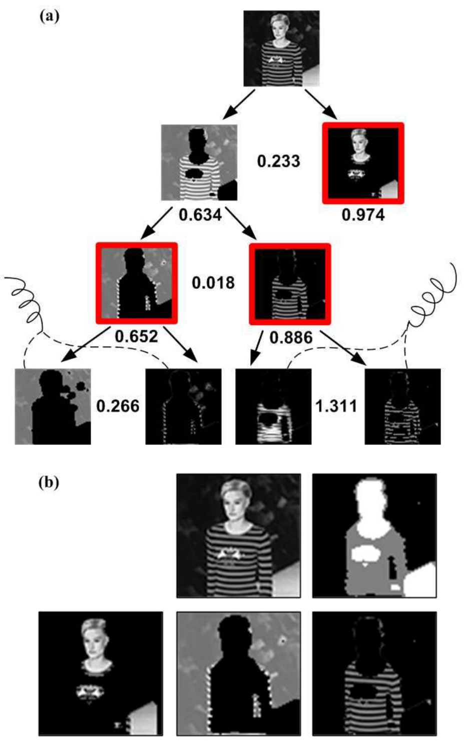

Figure 2. Segmentation result of a real texture image ('fashion show', 69×69). (a) Segmentation procedure by the binary tree ($T_1 = 0.9, T_2 = 0.25$). (b) The original and a labeled-region image followed by three segmented regions.

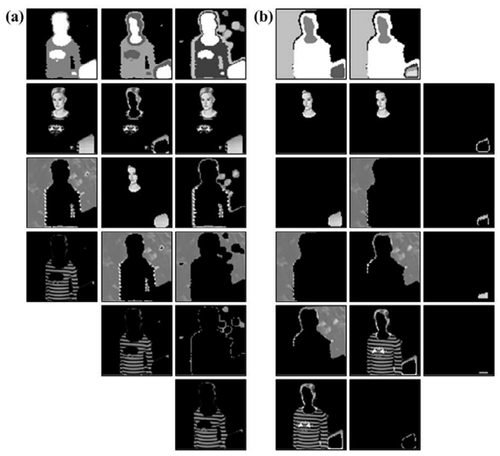

Figure 3. Comparison of the segmentation results. The result of each experiment using specific parameters is represented in each column where the segmented regions come after the labeled-region image shown in the first row. (a) Three results of the proposed method. (1st: $T_1 = 0.9, T_2 = 0.25$, 2nd: $T_1 = 0.98, T_2 = 0.25$, 3rd: $T_1 = 0.9, T_2 = 0.4$). (b) Two results of the normalized cut method (1st: $T_{ncut} = 0.235$, 2nd: $T_{ncut} = 0.25$).

where d_j denotes $\sum_{j,n \in C_j} A_{jn}$ and measures how strongly the node $j \in C_j$ is connected to the rest of C_j, and $\sum_{k \in C_k} A_{jk}$ measures how strongly the node j is connected to the nodes in the other cluster [10]. If the inter-cluster measure h_2 is smaller than the threshold T_2, then all nodes are more connected to nodes in the same cluster than to nodes in the other cluster, and thus the factorization by the intra-cluster measure is acceptable - if not, the two factorized sub-clusters are unacceptable and should be removed. The proposed inter-cluster measure has a similar form as the *min-max cut* [11]. This means that our results can show a tendency of the min-max cut method, although we do not directly minimize Eqn. (9) for partitioning.

4. Experimental results

In this section, we evaluate the proposed method by applying it to various kinds of data. Before presenting the results, we explain how to compute the affinity matrix by automatically selecting parameters such as the sigma for feature similarities and the neighborhood size and the sigma for spatial adjacencies. Since these parameters determine the shape of the affinity matrix, they significantly affect the performance of the proposed method. Most of the previous spectral clustering methods manually select these parameters depending on the input data and this makes it difficult to reproduce their algorithms. In this paper, we present rules for the automatic determination of parameters and experimentally show that they perform well. To determine σ_F in Eqn. (3), we first normalize the norms of the feature vector differences, $||\mathbf{F}(\mathbf{X}_i) - \mathbf{F}(\mathbf{X}_j)||$, and make a histogram of them. Then, we determine the σ_F as the value where the cumulative value of the histogram is about 20% of total histogram values - this procedure resembles the *noise estimator* described by Canny [12]. Further, we assert that the neighborhood size r and the sigma σ_X for the spatial adjacencies in Eqn. (3) should be set large so that the feature similarities between data with fairly large distances are sufficiently expressed in the affinity matrix. In our experiments, if the number of pixels is N_p, then we determine r as about $0.8\sqrt{N_p}$ and σ_X as about $0.6r$. By following these rules, we need not be bothered with the settings of many parameters and can obtain good and stable results.

Figure 2 shows the segmentation result of a real texture image, and from it we can understand how the proposed method performs unsupervised sparse clustering. For the texture image such as Fig. 2 we first compute feature

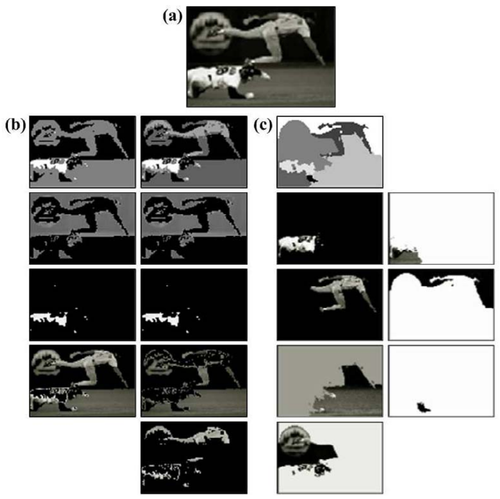

Figure 4. **Segmentation results of a real gray-scale image. In (b) and (c), the three results of the proposed and the Ncut method are shown in columns. In each result, a labeled-region image is followed by segmented regions. (a) The original image ('baseball', 96×66). (b) Two results of the proposed method (1st: $T_1 = 0.8, T_2 = 0.1$, 2nd: $T_1 = 0.9, T_2 = 0.1$). (c) The result of the normalized cut method ($T_{ncut} = 0.04$).**

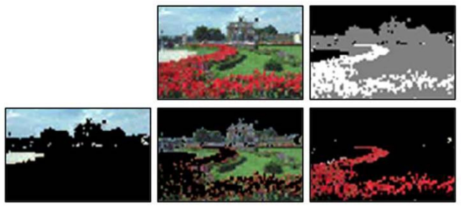

Figure 5. **Segmentation result of a real color image ('flower garden', 82×53). The original and a labeled-region image are represented with three segmented regions ($T_1 = 0.98, T_2 = 0.3$).**

vectors using squared magnitude responses of Gabor filters with different scales and orientations, where the window size is 7×7 and the numbers of scales and orientations are 5 (3, 4, 5, 6, 7) and 6 ($0°, 30°, 60°, 90°, 120°, 150°$), respectively [13]. Next, we compute the affinity matrix and perform a hierarchical segmentation based on PTF. In Fig. 2(a), a binary tree structure is shown and each node corresponds to each set of clustered pixels. In this figure, there are two kinds of numerals - the first one is represented under each node and means its intra-cluster score, $h_1(C_i)$, and the second one is represented between two nodes and means

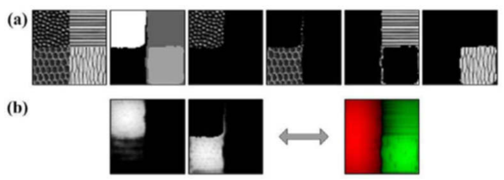

Figure 6. **Results of a texture image ('texture mosaic', 64×64). (a) The original and a labeled-region image followed by four segmented regions ($T_1 = 0.9, T_2 = 0.2$). (b) PTF results of the proposed method vs. the eigenvector of Ncut method.**

the inter-cluster score between them, $h_2(C_j, C_k)$. If $h_1(C_i)$ is larger than $T_1(= 0.9)$, the cluster C_i is not factorized any more, and if $h_2(C_j, C_k)$ is larger than $T_2(= 0.25)$, the factorized clusters C_j and C_k are removed. In Fig. 2(b), the original and a labeled-region image are shown in the first row, and three region images where each has one of the segmented regions are shown in the second row. The labeled-region image is obtained by assigning the same gray-level to the pixels in the same segmented region, and therefore it has the same number of gray-levels as that of the segmented regions.

Since the evaluation of image segmentation results is very subjective, it is difficult to define the only solution and thus the ability of an algorithm to provide multiple solutions is also important. In Fig. 3, we provide several segmentation results by changing thresholds for the intra- and inter-cluster measures, and compare them with those of the Ncut method [7]. For this, we implemented the Ncut method using Gabor filters as well as *DOOG (Difference of Offset Gaussians) filters* and the results using the Gabor filters are shown in Fig. 3(b). This is because the Ncut results using Gabor and DOOG filters are almost the same and the comparison with our method should be performed under the same conditions. The Ncut method also performs a recursive bi-partitioning which stops if the normalized cut value is larger than the threshold T_{ncut}. The results of the proposed method are shown in Fig. 3(a), where the first column is from Fig. 2(b) and the second and third column are obtained by altering the values of T_1 and T_2 from those of the first one, respectively. In the second column, by increasing T_1 for the intra-cluster measure, we factorize more the first region of the first column into a homogenous region with inner parts of the face and the stage and a textured boundary region enclosing them. In the third column, by increasing T_2 for the inter-cluster measure, we can obtain results more sensitive to soft textures in the background. The results of the Ncut method are shown in Fig. 3(b). In its first column, the Ncut method succeeds in separating the face and the stage into two regions, but fails in maintaining the background in a single region. Furthermore, it does not separate a mark in the breast from the clothes although they are different from a viewpoint of textures. In the second and the third column, the Ncut method increases T_{ncut} for separating the mark, but it only causes unwanted over-segmentation - this can be prevented by the intra-cluster measure of the proposed method. Therefore, we assert that the proposed method can provide superior segmentation results by interactively using both the intra-cluster and the inter-cluster measures.

Figure 4 shows the segmentation results of a real gray-scale image comparing our method with the Ncut method. The Ncut result is excerpted from the original paper [7], where the parameters of $\sigma_F = 0.1$, $\sigma_X = 4.0$, and $r = 5$ are used for computing the affinity matrix. As shown in Fig. 4(c), the Ncut method successfully extracts two players, but falsely segments background regions despite their clear boundaries. On the contrary, the proposed method successfully segments the background regions as well as two players as shown in the first column of Fig. 4(b). In the second column, we increase the threshold T_1 for the intra-cluster measure and obtain a more factorized player where the resulting boundaries are meaningful.

Figure 5 and 6 show that the proposed method also performs well in a real color and a synthetic texture image. In Fig. 6(a), we apply our method to the image of Brodatz-like texture patterns widely used for texture segmentation and obtain a satisfactory result. Figure 6(b) represents the two factorized components by the PTF - two columns of $\mathbf{C}^{(1)}$ and the second smallest eigenvector by the Ncut method. They are computed at the root node of the binary tree where all the pixels of the original image are utilized. As explained in Section 2.2, the proposed and the Ncut method can perform segmentation by discretizing the components and the eigenvector, respectively. A careful observation of Fig. 6(b) makes us realize the differences between the proposed method and the Ncut method. In the components factorized by the PTF, most of entries are zero and therefore the proposed method based on them is called as a *sparse* method. On the contrary, almost all of the entries of the eigenvector have non-zero values and some of them are even negative - they are indicated in red, which corresponds to the left half of the eigenvector image. The negative entries cause complex cancellations by subtractive combinations and lacks of *intuition* and *physical meaning*. Here, we assert that our method based on *positiveness* proceeds to the right direction for finding combinational components from positive data.

5. Conclusion

In this paper, we present a novel solution of the image segmentation problem by proposing a sparse clustering method based on PTF that tries to find additive combinations of positive components from the affinity matrix with originally positive entries. Further, by adopting a hierarchical structure of the binary tree with intra- and inter-cluster measures, we develop an unsupervised segmentation method which automatically determines the number of clusters. Our method provides interesting and challenging results which show that the *positiveness- and perception-based factorization* can have better performance than the methods of optimizing graph cuts in an image segmentation area. We can further prove the superiority of our method by applying it to various research areas of computer vision and image processing.

References

[1] G. Strang, *Linear Algebra and Its Applications*, Harcourt Brace Javanovich, Inc., Orlando, Florida, 1988.

[2] P. Paatero and U. Tapper, “Positive Matrix Factorization - A Nonnegative Factor Model with Optimal Utilization of Error-estimates of Data Values,” *Environmetrics*, **5**, pp. 111-126, 1994.

[3] D. D. Lee and H. S. Seung, “Learning the Parts of Objects by Non-negative Matrix Factorization,” *Nature*, **401**, pp. 788-791, 1999.

[4] M. Welling and M. Weber, “Positive Tensor Factorization,” *Pattern Recognition Letters*, **22**(12), pp. 1255-1261, 2001.

[5] B. A. Olshausen and D. J. Field, “Emergence of Simple-cell Receptive-field Properties by Learning a Sparse Code for Natural Images,” *Nature*, **381**, pp. 607-609, 1996.

[6] C. Chennubhotla and A. Jepson, “Sparse PCA - Extracting Multi-scale Structure from Data,” *Proc. of IEEE International Conference on Computer Vision*, pp. 641-647, 2001.

[7] J. Shi and J. Malik, “Normalized Cuts and Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(8), pp. 888-905, 2000.

[8] F. Chung, *Spectral Graph Theory*, American Mathematical Society, 1997.

[9] R. Kannan, S. Vempala, and A. Vetta, “On Clusterings - Good, Bad and Spectral,” *Proc. of 41st Symposium on the Foundation of Computer Science*, pp. 367-377, 2000.

[10] A. Ng, M. Jordan, and Y. Weiss, “On Spectral Clustering: Analysis and an Algorithm,” *Proc. of Advances in Neural Information Processing Systems*, pp. 849-856, 2001.

[11] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. “A Min-max Cut Algorithm for Graph Partitioning and Data Clustering,” *Proc. of IEEE International Conference on Data Mining*, pp. 107-114, 2001.

[12] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **8**(6), pp. 679-698, 1986.

[13] A. K. Jain and F. Farrokhnia, “Unsupervised Texture Segmentation Using Gabor Filters,” *Pattern Recognition*, **24**(12), pp. 1167-1186, 1991.