Multi-level Low-rank Approximation-based Spectral Clustering for image segmentation

Lijun Wang *,1, Ming Dong

Machine Vision Pattern Recognition Lab, Department of Computer Science, Wayne State University, Detroit, MI 48202, USA

ABSTRACT

Spectral clustering is a well-known graph-theoretic approach of finding natural groupings in a given dataset, and has been broadly used in image segmentation. Nowadays, High-Definition (HD) images are widely used in television broadcasting and movies. Segmenting these high resolution images presents a grand challenge to the current spectral clustering techniques. In this paper, we propose an efficient spectral method, Multi-level Low-rank Approximation-based Spectral Clustering (MLASC), to segment high resolution images. By integrating multi-level low-rank matrix approximations, i.e., the approximations to the affinity matrix and its subspace, as well as those for the Laplacian matrix and the Laplacian subspace, MLASC gains great computational and spacial efficiency. In addition, the proposed fast sampling strategy make it possible to select sufficient data samples in MLASC, leading to accurate approximations to the segmentation. From a theoretical perspective, we mathematically prove the correctness of MLASC, and provide detailed analysis on its computational complexity. Through experiments performed on both synthetic and real datasets, we demonstrate the superior performance of MLASC.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Computer vision is an intellectual frontier to explore the theories and techniques that make machines see like the human beings. Image segmentation is a fundamental problem in computer vision which partitions an image into meaningful parts based on the features like color, texture, shape and spatial locations (Pal and Pal, 1993). From a theoretical perspective, image segmentation attempts to determine what components of a dataset naturally belong together, and this is a problem well-known as clustering (Forsyth and Ponce, 2002). Among various clustering techniques for image segmentation, spectral method had emerged as a popular one (Shi and Malik, 2000; Bach and Jordan, 2003). This is mainly because it can group the pixels in the affinity domain, and consequently, capture non-convex geometries and solve a segmentation problem even when the meaningful regions cannot be easily separated by the perceived features.

Since the early 1990s, High-Definition (HD) images are widely used in television broadcasting and movies. Generally, an HD image is of the size 1920 × 1080. Images with higher resolution have also been popular today, e.g., on the Internet. To segment an image using spectral clustering, the image is usually modeled as a weighted graph, in which each pixel is a node, and each pair of pixels is connected by an edge. The partition of the graph (and thus the segmentation of the image) relies on the eigen-decomposition of the graph’s Laplacian matrix. Assuming an image has \( n \) pixels, its Laplacian matrix is of size \( n \times n \). This high spacial cost greatly limits the usage of spectral clustering on segmenting high resolution images.

To improve the efficiency of spectral clustering on large-scale datasets, many approximate schemes were proposed in the literature. Examples include using quantization and down-sampling to reduce the size of data (Yan et al., 2009), or incorporating low-rank matrix approximation to expedite and economize the computation (Fowlkes et al., 2004; Zhang et al., 2008; Wang et al., 2009). However, for extremely large data, all the aforementioned algorithms can only choose a small volume of data samples, usually determined by the available memory space. Another viable option to cluster large-scale data is to use a distributed or parallel strategy, such as distributed k-means (Jin et al., 2006) and parallel spectral clustering (Song et al., 2008). These approaches require a distributed computing infrastructure.

In this paper, we propose an efficient spectral method, Multi-level Low-rank Approximation-based Spectral Clustering (MLASC), to segment high resolution images. The main idea of the proposed method is, the eigenvectors of the graph’s Laplacian matrix can be approximated by extending the low-dimensional singular vectors to the high-dimensional ones from its representative subspace. Specifically, we first compute the low-rank approximation to the...
affinity matrix in a representative subspace, based on which the subspace of the graph’s Laplacian matrix and its top eigenvectors are further approximated. Finally, \( k \)-means is used for clustering in the computed embedding space.

MLASC incorporates multi-level low-rank approximations into spectral clustering, including the approximations to the affinity matrix and its subspace, as well as to the Laplacian matrix, its subspace and eigenvectors. Thus, the computational efficiency is greatly improved. When \( n \) is extremely large, existing low-rank methods are not able to sample sufficient amount data to achieve a good approximation, while MLASC can obtain enough samples through multi-level approximations, and thus maintain an accurate approximation. From a theoretical perspective, we mathematically prove the correctness of MLASC, and show it has a time complexity of \( \mathcal{O}(n) \). Through experiments performed on both synthetic and real datasets, we demonstrate the superior performance of MLASC.

The rest of the paper is organized as follows. We review related works in Section 2. The MLASC model formulation is presented in Section 3 along with a detailed algorithm. We report the results obtained on both synthetic and real-world datasets in Section 4 and conclude in Section 5.

2. Related work

In this section, we provide a brief review of image segmentation techniques, and introduce some essential background on spectral clustering and low-rank matrix approximation.

2.1. Image segmentation and spectral clustering

Image segmentation is typically used to locate objects and boundaries in a given image. General-purpose algorithms and techniques developed for image segmentation mainly include thresholding, edge detection, region splitting and merging, and clustering (Forsyth and Ponce, 2002; Xia et al., 2007; Jeon et al., 2006). In particular, clustering methods regard the image pixels as data points, and segment an image by grouping the pixels into different clusters.

Spectral clustering for image segmentation is based on graph theory. It formulates an image by a weighted graph and partitions it according to some cost functions. More specifically, a relaxation version to the graph partition problem is used to infer the pixel clusters from the eigenvectors of the graph’s Laplacian matrix (Fiedler, 1973; Chung, 1997; Ding et al., 2001; Shi and Malik, 2000; Cai et al., 2004). Traditional spectral clustering is not suitable for segmenting megapixel images due to its high spacial cost in the computation.

To overcome this difficulty, various methods are proposed in the literature to approximate spectral clustering. One approach is to use “coarsening and uncoarsening” methodology (Yan et al., 2009), in which data preprocessing techniques are employed to overcome computational and spacial bottlenecks, such as multi-scale density estimation (Mitra et al., 2002) and core-sets extraction (Badoiu et al., 2002). The preprocessor step is required to be fast and less memory consuming, moreover, it is expected not to negatively impact the clustering results. Yan et al. (2009) propose a fast approximate spectral clustering on the large-scale data by using \( k \)-means or random tree as the preprocessor. Once the cluster centers are obtained in the preprocessor step, they are regarded as the representatives to be clustered by a traditional spectral clustering. Finally, the memberships of the full data are recovered according to the relationship between data points and representatives. As shown in (Yan et al., 2009), the \( k \)-means or random tree preprocessor aims to minimize the effect of data reduction on the clustering accuracy in the spirit of rate-distortion theory.

Another approach to accommodate the fast growing data is to incorporate low-rank approximations. Since the solution of the spectral clustering is eventually to solve the eigenvalue decomposition problem in the graph’s Laplacian domain, various algorithms for approximating the eigenvalue decomposition can be adopted. For example, in (Fowlkes et al., 2004), a Nyström-based spectral clustering algorithm is proposed for image segmentation. The main idea is: first a set of data samples are randomly selected under a uniform distribution, and then, the affinities between samples are computed by a kernel. According to the Nyström theory (Baker, 1997; Delvins and Mohamed, 1985), the approximation to the Laplacian matrix is computed from the sample affinities, and the eigenvectors are extended from sample space to the full Laplacian domain.

To improve Nyström spectral clustering, some methods have been proposed in the literature by applying various quantization schemes. In (Zhang and Kwok, 2006), a block quantization strategy is used for fast spectral embedding by approximating the Laplacian kernel with a block-wise constant kernel matrix. Though the error analysis is provided on the quantization step, how the Nyström method affects the approximation quality in general remains unclear (Zhang et al., 2008). In (Zhang et al., 2008), \( k \)-means clustering is stated to find a local minimum of the quantization error, and thus, the preprocessor with the \( k \)-means step can improve the approximation accuracy.

2.2. Low-rank matrix approximation

Low-rank matrix approximations have recently gained popularity in computer vision, information retrieval and machine learning. A widely used low-rank approximation is SVD, which is known to be optimal in the sense of achieving the minimum reconstruction error with the Frobenius norm. However, the algorithms for computing the SVD generally operate through repeated matrix–vector multiplication, thereby requiring super-linear time and large working sets (Golub and Van, 1996). Moreover, the result of SVD is usually dense even if the original matrix is sparse, so it is prohibitive to be applied on large-scale datasets (Drineas et al., 2006a).

On the other hand, near-optimal low-rank approximations, a.k.a., exemplar-based approaches (Tong et al., 2008), can preserve the sparsity of the matrix while achieving great savings on running time and space. For example, CUR is a well-known exemplar-based approximation method (Drineas et al., 2006b), which selects the real columns and rows based on a judiciously-chosen data distribution. Even though the original data may be too large to be completely loaded in the memory, CUR allows one to keep a small randomly-chosen and rapidly-computable “sketch” in RAM with a few passes over the original data stored in the disk. This promising property makes it possible to analyze tremendous amounts of data by studying its “sketch”. More recently, CMD (Sun et al., 2008) and Colibri (Tong et al., 2008) are developed as two variants of CUR to improve its efficiency. Specifically, CMD can generate a unique column subspace by carefully removing duplicate columns. Further, Colibri (Tong et al., 2008) is proposed to solve the problem of redundant or overcomplete basis. That is, the data samples of the initial subspace may be linearly dependent or near-duplicate, which is usually seen in a tightly-connected subgraph where all nodes have similar neighbors. Consequently, with the redundant basis, the approximation is not efficient in terms of space. Thus, the Colibri method aims to generate a unique and linearly-independent column subspace. Both algorithms are shown to provide decomposition techniques equivalent to the CUR algorithm while requiring less time and space.
All the aforementioned approximation algorithms can be used to generate a low-rank approximation to an arbitrary matrix. Kernel-based methods have been popular in data mining and machine learning, which are widely used to extract the non-linear structure hidden inside the data, e.g., Support Vector Machines (SVM) and kernel Principal Component Analysis (KPCA). When processing large-scale data, the kernels used in these learning algorithms are expensive to compute and store. Hence, it is necessary to explore the techniques for improving the kernel machines. The well-known methods include Nyström-based approaches (Baker, 1997; Delves and Mohamed, 1985), gram matrix approximation (Drineas and Mahoney, 2005), and the column-sampling based approach (Drineas et al., 2006a).

3. Multi-level Low-rank Approximation-based Spectral Clustering

3.1. Model formulation
MLASC is formulated based on the theory in (Drineas et al., 2006a), that is, the top $k$ eigenvectors of the Laplacian matrix $L$ can be computed approximately by its subspace. Specifically, we first construct a representative subspace of the affinity matrix ($S$), based on which, the low-rank approximation $\tilde{S}$ is computed; then we compute the corresponding approximate subspace of $L$; and finally the approximation to its top $k$ eigenvectors. In the following, we discuss each step in details.

3.1.1. Compute the subspace and low-rank approximation of the affinity matrix
In our formulation, the affinity matrix is computed by a non-linear kernel, thus, we employ a well-known kernel matrix approximation method (Drineas and Mahoney, 2005) to generate $\tilde{S}$. Specifically, let $S(:,i)$ be the $i$th column of $S$, then the nearly-optimal probabilities $\{p_i\}_{i=1}^n$ (Drineas et al., 2006a) for sampling the columns of $S$ are defined as,

$$p_i = \beta \frac{\|S(:,i)\|_F^2}{\sum_{i=1}^n \|S(:,i)\|_F^2}, \quad \text{s.t.} \sum_{i=1}^n p_i = 1, \quad i \in [1,n].$$

(1)

where $\beta \leq 1$, and $\| \cdot \|_F$ is the square of the Frobenius norm. Note that when $\beta = 1$, they are optimal probabilities.

Under $\{p_i\}_{i=1}^n$, we select a set of columns (denote the index set as $I_c$) and get the subspace $C (C = S(:, I_c))$. By using the method proposed in (Drineas and Mahoney, 2005), we first scale each column of $C$ as,

$$C(:,i) = C(:,i) / \sqrt{c_i};$$

(2)

where $c$ is the number of columns. Let $W$ be the scaled intersection between the selected columns and the corresponding rows, $W(:,i) = C(I_c(:,i)) / \sqrt{c_i}$

(3)

$W$ be the best rank-$k$ approximation to $W$, and $W_k^{-1}$ be the Moore-Penrose generalized inverse of $W_k$, we can compute the approximation to the affinity matrix as,

$$\tilde{S} = C W_k^{-1} C^T.$$

(4)

Sampling issues: From Eq. (1), the sampling probabilities are computed with $S$, which is expensive when $S$ is large. This issue is common for image segmentation problems. For instance, $S$ is of the size 480,000 $\times$ 480,000 for a 600 $\times$ 800 image. Here, we propose to use an uniform distribution for nearly-optimal sampling, i.e., $p_i = 1/n$, and state,

Proposition 1. Define

$$p_i = \frac{1}{n}, \quad i \in [1,n],$$

(5)

then $\{p_i\}_{i=1}^n$ are nearly-optimal for sampling the columns of $S$ (see the appendix for proof).

Subspace volume: When $n$ is extremely large, the subspace $C_{n \times c}$ is restricted to have a very small volume due to the memory limitation. Generally, the approximation errors increase with the decrease of the subspace size. Hence, using a small subspace to approximate a large affinity matrix is not reliable. To solve this issue, we propose an iterative method to compute an approximate subspace $C$, based on which $S$ can be obtained.

The main idea is, with a small initial subspace, we iteratively compute a larger approximate subspace based on the uniform probability distribution Eq. (5), until a sufficient subspace is obtained. To efficiently compute the approximate subspace in each iteration, we apply LinearTimeCUR algorithm (Drineas et al., 2006b). Specifically, we first randomly select a small subspace with $c_1$ samples, i.e., $C_1(n \times c_1)$, then a larger subspace with $c_2 (> c_1)$ samples can be approximated as,

$$C_{2,n-c_1} = C_1 M_2 R_2^T.$$  

(6)

Here, $C_1$ is the scaled matrix of $C_1$ by Eq. (2). $R_2$ is the scaled selected row matrix

$$R_2 = C_1(I_{c_1} ;).$$

(7)

where $I_{c_1}$ is the index vector of the selected rows. To compute $M_2$, the middle matrix of the approximation, we first compute

$$C_1^T C_1 = \sum_{i=1}^n \sigma_i^2 (C_1) y_i^T y_i^T,$$

(8)

where $\sigma_i (C_1)$ is the $i$th top eigenvalue of $C_1$, and $y_i$ is the corresponding eigenvector. Then, $\Phi$ can be computed by the top $k$ eigenvalues and eigenvectors of $C_1$ as,

$$\Phi = \sum_{i=1}^k \frac{1}{\sigma_i (C_1)} y_i^T y_i^T.$$  

(9)

Next, we compute another variable $\Psi$ with

$$\Psi(r,.,:) = C_1(I_{c_1} ;) / \sqrt{p_{c_1}}, \quad i \in I_{c_1},$$

(10)

where $r$ is the number of selected rows, and $p_i$ is the probability of the $i^{th}$ row is selected, which is $1/n$ by using uniform sampling. Finally, we compute $M_2$ by

$$M_2 = \Phi \Psi^T.$$  

(11)

Similarly, the iterative rule to compute the $k$th approximate subspace is,

$$C_k = C_{k-1} M_k R_k.$$  

(12)

Since $M_k$ and $R_k$ are small matrices when compared with $C_k$, we can construct a large volume approximate subspace $\tilde{C}$ by using Eq. (12). The corresponding low-rank approximation to $S$ is

$$\tilde{S} = \tilde{C} W_k^{-1} \tilde{C}^T,$$

(13)

where $\tilde{C}$ is the result after scaling $\tilde{C}$ by Eq. (2).

3.1.2. Compute the subspace of $L$
Given the subspace of the affinity matrix, we next compute the corresponding subspace of $L$. First, we compute the approximation of the degrees as,

$$\tilde{d} = \tilde{S} \hat{1}.$$  

(14)
Then, the degree matrix $\tilde{D}$ is constructed as a diagonal matrix with each item of $\tilde{d}$ on its diagonal, and $\tilde{D}^{-\frac{1}{2}}$ with $\frac{1}{\sqrt{d_i}} (i \in [1, n])$ on its diagonal. We also compute the subspace of $\tilde{D}^{-\frac{1}{2}}$ as,

$$\tilde{D}^{-\frac{1}{2}} = \tilde{D}^{-\frac{1}{2}}(I_c, I_c),$$  \hfill (15)

where $I_c$ is the index set of the selected columns. Finally, the approximation to $L$ is

$$\tilde{L} = \tilde{D}^{-\frac{1}{2}}\tilde{S}\tilde{D}^{-\frac{1}{2}},$$  \hfill (16)

and the subspace of $\tilde{L}$ is

$$\tilde{L}_{sub} = \tilde{D}^{-\frac{1}{2}}C\tilde{D}^{-\frac{1}{2}} \approx \tilde{D}^{-\frac{1}{2}}C\tilde{D}^{-\frac{1}{2}} = \tilde{L}(; I_c).$$  \hfill (17)

We claim,

**Proposition 2.** If $C$ is a subspace of $S$, then $\tilde{L}_{sub}$ Eq. ((17)) is an approximate subspace of $L$. Further, if $C$ is an approximate subspace of $S$, then $\tilde{L}_{sub}$ is also an approximate subspace of $L$. (The proof is straightforward and omitted for abbreviation).

$\tilde{L}_{sub}$ needs to be selected based on nearly-optimal probabilities (refer to Eq. (1)) in the Laplacian domain so that the top eigenvectors can be computed. In our model, we generate $\tilde{L}_{sub}$ according to the subspace of $S$. We claim that,

**Proposition 3.** Given the optimal probabilities $\{p_i\}_{i=1}^n$ for sampling the columns of the affinity matrix $S$, where

$$p_i = \frac{\|S(:, i)\|^2}{\sum_{j=1}^{n} \|S(:, j)\|^2},$$  \hfill (18)

then, $\{p_i\}_{i=1}^n$ are nearly-optimal probabilities for sampling $\tilde{L}_{sub}$ in the Laplacian domain. Further, the nearly-optimal probabilities computed by Eq. (1) are nearly-optimal for sampling the columns of the Laplacian matrix (see the appendix for proof).

### 3.1.3. Compute the approximation to the top eigenvectors of $L$

It is shown in (Drineas et al., 2006a) that the top singular values and the corresponding singular vectors of a matrix could be approximated from its subspace matrix, selected under the nearly-optimal probabilities. With the probabilities obtained in Eqs. (5) and (18), we state that,

**Proposition 4.** In MLASC, the approximation to the top eigenvectors of $L$ can be computed from $\tilde{L}_{sub}$ (see the appendix for proof).

Given $\tilde{L}_{sub}$, which consists of those $c$ columns of $L$ selected based on the nearly-optimal probabilities, we first scale it by

$$\tilde{L}_{sub}^t(; i) = \tilde{L}_{sub}(; i)/\sqrt{p_i}, \; i \in [1, n],$$  \hfill (19)

where $p_i$ is the sampling probability used in approximating the affinity matrix. Then, we compute the SVD of $\tilde{L}_{sub}^t\tilde{L}_{sub}$ as

$$\tilde{L}_{sub}^t\tilde{L}_{sub} = V\Sigma V^t,$$  \hfill (20)

where $V$ has the singular vectors and $\Sigma$ the singular values. Finally, the top $k$ singular vectors of $L$ can be approximated as

$$U = \tilde{L}_{sub} \ast V(:, 1:k)\Sigma^{-\frac{1}{2}}.$$  \hfill (21)

### 3.2. Algorithm

Algorithm 1 shows the steps of MLASC.

**Algorithm 1. MLASC**

**INPUT:** the data matrix $A$, the maximum iteration $n_{\text{MaxIter}}$, and the cluster number $k$

**OUTPUT:** Cluster labels

**METHOD:**

1. Compute $C$ (or $\tilde{C}$) and $\tilde{S}$;
2. Compute the degree matrix $\tilde{L}$;
3. Compute the approximations to the top $k$ eigenvectors of $L$ by Eqs. (19)-(21);
4. Normalize the row vectors: $U(t,:) = U(t,:) / \|U(t,:)\|_2, \; t \in [1, n]$.

5. Regarding each row of $U$ as a point, perform clustering via $k$-means, and return the cluster labels.

### 3.3. Efficiency analysis

From Algorithm 1, the first step of MLASC is to compute the sampling probabilities, which use $O(1)$ time for an uniform sampling. Generating $\tilde{S}$ and $\tilde{C}$ needs $O(n)$ time (Drineas and Mahoney, 2005; Drineas et al., 2006b). Next, computing the degree matrix requests for $O(2cn + c^2)$ time, $L_{sub}^tL_{sub}$ for $O(n + nc + nc^2 + 2c^2)$, its SVD for $O(c^3)$, and $U_k$ for $O(k(c + c^2 + nc + n))$. Finally, $k$-means is used to cluster $U_{\tilde{k}}$ with each row as a point, which needs $O(kn)$ time, where $k$ is the number of iterations. Since $c \ll n$ and $k \ll c$, the total time of MLASC is $O(n^2 + nk)$. Thus, MLASC has a linear time complexity regarding $n$, very efficient to cluster large-scale datasets.

### 4. Experiments and results

In this section, we evaluate the performance of MLASC on both synthetic and real-world image datasets in terms of the clustering accuracy (or segmentation quality), the approximation error, and the running time. Note that the affinity matrix can be approximated using either Eq. (4) or Eq. (13). The difference is whether we start with the original subspace or the approximated one in the following, we denote the two different methods with MLASC-O and MLASC-A, respectively, and call them MLASC when no differentiation is needed. Particularly, in MLASC-A, for the tradeoff between the efficiency and accuracy, we empirically set the initial subspace size $c_1$, and in the iterations, we set $c_k = 2c_{k-1}$ ($\alpha > 1$).

In order to evaluate our methods, we compare it with three leading approximate-based spectral clustering algorithms, i.e., Nyström-based spectral clustering with random sampling (Fowlkes et al., 2004), InNyström with $k$-means as a preprocessor (Zhang et al., 2008), and KASP (Yan et al., 2009), a quantization-based fast spectral clustering method. In addition, the performance of the baseline clustering algorithm: normalized cuts (NCut) (Shi and Malik, 2000) on the synthetic and small-size real data is also reported. In our experiments, the subspace size $c$ in MLASC and Nyström is selected manually based on the observation of achieving reasonable segmentation results, which also serves as the number of representative points in the $k$-means step of InNyström and KASP. All the algorithms were implemented using MATLAB 7. The experiment was performed on a machine with a $3.0$ GHz Intel Xeon.
CPU, 3.0 GB RAM and the Windows XP operating system. In our experiment, the following evaluation metrics are used:

1. **Normalized mutual information (NMI)** (Strehl et al., 2002) is computed from the confusion matrix based on the true and predicted cluster labels, and ranges in \([0, 1]\). A high NMI value indicates that the predicted cluster label matches the ground truth well.

2. **Quantitative evaluation**: We evaluate color image segmentation results using \(Q\)-value (Borsotti et al., 1998), and a low value indicates good performance. In addition, three other performance metrics, i.e., Variation of Information (VI), Probabilistic Rand Index (PRI), and Segmentation Covering (Covering), are computed based on the ground-truth provided by the Berkeley benchmark datasets (Arbelaez et al., 2011). A high VI, high PRI, and low Covering value indicate good performance.

3. **Approximation error**: We use the sum-square-error (SSE) metric to evaluate the approximation accuracy,

\[
SSE = \frac{\| S - \tilde{S} \|_F}{\| S \|_F},
\]

where \(S\) is the affinity matrix, and \(\tilde{S}\) the approximation matrix. To evaluate the approximation error of eigenvectors, we compute the Euclidean distance between the projection matrices spanned by \(H_k\) (achieved by normalized cut) and \(U_k\) (obtained by MLASC),

\[
SSE = \frac{\| H_k U_k^T - U_k U_k^T \|_F}{\| H_k H_k^T \|_F}.
\]

A lower SSE value indicates higher approximation accuracy.

4. **Time**: We use the inline functions of MATLAB, tic and toc, to compute the actually running time of an algorithm.

### 4.1. Synthetic data

In this section, we illustrate the performance of MLASC on a non-linear synthetic dataset, shown in Fig. 1 (a). The data is constructed by two rings, each of which contains 1000 points. The affinity matrix is computed by \(S_{ij} = \exp(-|i-j|^2/2r^2) + \kappa\), where \(\sigma = 20\), \(\lambda = 10\) and \(\kappa = -10\). For MLASC-A, we set \(c_1 = 20\), and \(a = 5/4\).

First, we study the performance of MLASC with various values of \(c\). In Fig. 1 (b), we plot the average clustering accuracy against \(c\) for the six (approximate) spectral algorithms. Since the results obtained by NCut are independent to \(c\), we plot it as a flat line. Clearly, both our methods perform well. When \(c\) is greater than 50, they all get comparable clustering accuracy as NCut. Particularly, when \(c\) is over 300, MLASC-A achieves the highest clustering accuracy among the six. This indicates that, instead of using the original subspace (MLASC-O), we can also gain good clustering in the approximate subspace (MLASC-A). We see that INyström and KASP gain comparable results as MLASC, however, Nyström performs poorly.

Next, we compare the running time, shown in Fig. 1 (c). We observe MLASC-A is the fastest among all the approximate methods, followed by MLASC-O. The Nyström algorithm runs slowly when \(c\) is large, e.g., the running time is four times than that of MLASC.

**Fig. 1.** Comparison of clustering accuracy and efficiency with various \(c\): (a) shows the synthetic dataset, (b) shows the comparison of NMI, and (c) shows comparison of running time.
(a) Approximation error of the affinity matrix

(b) Approximation error of the eigenvectors

Fig. 2. Comparison of approximation errors of the affinity matrix and eigenvectors among four algorithms with various $c$.

Fig. 3. Segmentation results on small-size dataset (49 x 73) with $c = 100$.

<table>
<thead>
<tr>
<th>Picture</th>
<th>Evaluation metric</th>
<th>Methods</th>
<th>MLASC-O</th>
<th>Nystrom</th>
<th>INystrom</th>
<th>KASP</th>
<th>NCut</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Q-value</td>
<td>0.0043</td>
<td>0.0043</td>
<td>0.0051</td>
<td>0.0065</td>
<td><strong>0.0042</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td>0.5296</td>
<td>0.5007</td>
<td>0.2762</td>
<td><strong>0.5776</strong></td>
<td>0.5248</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td>0.7732</td>
<td>0.7621</td>
<td>0.6264</td>
<td><strong>0.7870</strong></td>
<td>0.7699</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td>2.2423</td>
<td>2.2876</td>
<td>3.4735</td>
<td><strong>1.9572</strong></td>
<td>2.2405</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>Q-value</td>
<td>0.0065</td>
<td>0.0062</td>
<td><strong>0.0043</strong></td>
<td>0.0075</td>
<td>0.0049</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td><strong>0.3680</strong></td>
<td>0.3674</td>
<td>0.2178</td>
<td>0.3645</td>
<td>0.3344</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td><strong>0.7542</strong></td>
<td>0.7509</td>
<td>0.6629</td>
<td>0.6906</td>
<td>0.7357</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td>3.4101</td>
<td>3.3717</td>
<td>4.1124</td>
<td><strong>3.2185</strong></td>
<td>3.7077</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Q-value</td>
<td>0.0068</td>
<td>0.0093</td>
<td><strong>0.0044</strong></td>
<td>0.0073</td>
<td>0.0066</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td><strong>0.3428</strong></td>
<td>0.3236</td>
<td>0.2596</td>
<td>0.3303</td>
<td>0.3403</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td>0.7706</td>
<td><strong>0.7710</strong></td>
<td>0.6767</td>
<td>0.7555</td>
<td>0.7690</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td>3.4362</td>
<td>3.4633</td>
<td>3.6380</td>
<td>3.4328</td>
<td><strong>3.4014</strong></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>Q-value</td>
<td>0.0050</td>
<td>0.0073</td>
<td><strong>0.0024</strong></td>
<td>0.0082</td>
<td>0.0051</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td>0.4554</td>
<td>0.3959</td>
<td>0.4055</td>
<td>0.4440</td>
<td><strong>0.4558</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td>0.7475</td>
<td>0.7177</td>
<td>0.6826</td>
<td>0.7044</td>
<td>0.7480</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td><strong>2.4581</strong></td>
<td>2.7885</td>
<td>2.5546</td>
<td>2.5493</td>
<td>2.4629</td>
<td></td>
</tr>
<tr>
<td>Average time (s)</td>
<td></td>
<td><strong>0.55</strong></td>
<td>1.91</td>
<td>2.46</td>
<td>4.59</td>
<td>7.18</td>
<td></td>
</tr>
</tbody>
</table>

The bold values show the best results among various methods.
when $c$ is over 300. Note that all approximate methods can run faster than NCut within a small subspace, i.e., $c < 400$, and thus achieve higher efficiency.

Finally, we plot the approximation error of the affinity matrix $S$ vs. $c$, shown in Fig. 2(a). Obviously, MLASC achieves lower approximation errors than other methods for all $c$. Notice that with the same $c$, MLASC-A gets a higher SSE than MLASC-O, indicating that the use of the approximate subspace will increase the approximation error of $S$. We also compare the approximation errors of the eigenvectors in Fig. 2(b). Clearly, due to the lower approximation error of $S$, MLASC and INyström can achieve more accurate embedding than Nyström for all $c$.

### 4.2. Real data

In this section, we evaluate MLASC on real-world image datasets. The images are mostly collected from the Berkeley segmentation Dataset and Benchmark (Martin et al., 2001) with the same size of $321 \times 481$ ($n = 154,401$ pixels). First, to compare the segmentation of MLASC with NCut, we construct a small-size real dataset by zooming out the benchmark images to the size of $49 \times 73$ ($n = 3577$ pixels). Second, sample images with original size are selected to form the medium-size dataset. On both datasets, we run MLASC-O. Finally, to evaluate our method on large-size (HD-quality) images, we construct two datasets. One is formed by zooming the benchmark images to the size of $1429 \times 2141$ (3 million pixels), which is done using the bi-cubic interpolation method provided by MATLAB. Since we use a uniform sampling when constructing the affinity subspace, i.e., samples are selected randomly with equal probabilities, our method will not suffer adverse consequences from this operation. The other is formed by real HD images with the size of $1200 \times 1600$ (about 2 million pixels).2 When running on a computer with 3.0 GB RAM, no algorithms except MLASC-A can have sufficient sample pixels to achieve reasonable segmentation on large-size datasets. So, only the results of MLASC-A are reported. The parameters of MLASC-A are set with $c_1 = 10$, and $x = 3/2$. Based on our visual observation, the number of segments in a testing image is generally between 4 and 6, and the average number, 5, is used in our experiment.

To compute the similarity between the pixels in an image, we use both color and coordinate information,

$$S_{ij} = e^{-\frac{|c_i - c_j|^2}{2\sigma_c^2}} e^{-\frac{|r_i - r_j|^2}{2\sigma_r^2}}.$$

*Fig. 4.* Segmentation results on medium-size dataset ($321 \times 481$) with $c = 100$.

<table>
<thead>
<tr>
<th>Picture</th>
<th>Evaluation metric</th>
<th>Methods</th>
<th>MLASC-O</th>
<th>Nyström</th>
<th>INyström</th>
<th>KASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elephant</td>
<td>Q-value</td>
<td>0.0026</td>
<td>0.0045</td>
<td>0.0026</td>
<td>0.0038</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td>0.4559</td>
<td>0.4275</td>
<td>0.2767</td>
<td>0.4465</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td>0.8102</td>
<td>0.7977</td>
<td>0.7436</td>
<td>0.7387</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td>2.3437</td>
<td>2.7179</td>
<td>3.4722</td>
<td>2.3356</td>
<td></td>
</tr>
<tr>
<td>Deer</td>
<td>Q-value</td>
<td>0.0056</td>
<td>0.0067</td>
<td>0.0037</td>
<td>0.0068</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td>0.5779</td>
<td>0.5596</td>
<td>0.5026</td>
<td>0.5692</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td>0.7818</td>
<td>0.7726</td>
<td>0.7389</td>
<td>0.7729</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td>1.9105</td>
<td>1.9850</td>
<td>2.3529</td>
<td>2.0330</td>
<td></td>
</tr>
<tr>
<td>Tiger</td>
<td>Q-value</td>
<td>0.0051</td>
<td>0.0091</td>
<td>0.0023</td>
<td>0.0101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td>0.3879</td>
<td>0.3772</td>
<td>0.4591</td>
<td>0.3766</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td>0.5638</td>
<td>0.5608</td>
<td>0.6054</td>
<td>0.5673</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td>2.7662</td>
<td>2.8727</td>
<td>2.3526</td>
<td>2.7855</td>
<td></td>
</tr>
<tr>
<td>Building</td>
<td>Q-value</td>
<td>0.0017</td>
<td>0.0104</td>
<td>0.0018</td>
<td>0.0047</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Covering</td>
<td>0.4559</td>
<td>0.3625</td>
<td>0.4462</td>
<td>0.4428</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRI</td>
<td>0.7477</td>
<td>0.7391</td>
<td>0.7356</td>
<td>0.7128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VI</td>
<td>1.9317</td>
<td>2.5098</td>
<td>2.0553</td>
<td>2.0510</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison of quantities on medium-size dataset ($c = 100$): Q-value, Covering, PRI, and VI.

The bold values show the best results among various methods.

<table>
<thead>
<tr>
<th>Name</th>
<th>MLASC</th>
<th>Nyström</th>
<th>INyström</th>
<th>KASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elephant</td>
<td>5.19</td>
<td>5.18</td>
<td>119.42</td>
<td>100.97</td>
</tr>
<tr>
<td>Deer</td>
<td>4.95</td>
<td>10.46</td>
<td>119.93</td>
<td>104.38</td>
</tr>
<tr>
<td>Tiger</td>
<td>3.69</td>
<td>11.64</td>
<td>117.41</td>
<td>99.80</td>
</tr>
<tr>
<td>Building</td>
<td>3.01</td>
<td>8.46</td>
<td>106.97</td>
<td>92.21</td>
</tr>
</tbody>
</table>

Table 3: Comparisons of the running time (s) on medium-size images.

The bold values show the best results among various methods.

2 http://www.hdwallpapers.net.
where $A_c(i,:)$ represents a 3-dimensional (RGB) color vector, and $A_p(i,:)\) a 2-dimensional coordinate vector. In our experiments, we empirically set $\sigma_c = 4$, and $\sigma_p = 0.1$ on all testing images. The sub-space size is set at $c = 100$ for the medium dataset and $c = 1000$ for the large datasets.

4.2.1. Experiments on small-size and medium-size datasets

First, we compare the performance of the five algorithms, i.e., MLASC-O, Nyström, INyström, KASP, and NCut, on small-size

![Fig. 5. Segmentation results on large-size dataset (1429 × 2141) with c = 1000.](image)

Table 4

<table>
<thead>
<tr>
<th>Name</th>
<th>Q-value</th>
<th>Covering</th>
<th>PRI</th>
<th>VI</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse</td>
<td>0.0097</td>
<td>0.4418</td>
<td>0.6975</td>
<td>2.5966</td>
<td>90.98</td>
</tr>
<tr>
<td>Stone</td>
<td>0.0066</td>
<td>0.4496</td>
<td>0.8498</td>
<td>2.3959</td>
<td>88.23</td>
</tr>
<tr>
<td>Zebra</td>
<td>0.0069</td>
<td>0.3248</td>
<td>0.6032</td>
<td>2.5530</td>
<td>83.84</td>
</tr>
</tbody>
</table>

![Fig. 6. Segmentation results on HD images (1200 × 1600) with c = 1000.](image)
images, shown in Fig. 3. For each line in this figure, the original color images are shown at the beginning, followed by the segmentation results obtained by the five algorithms. The segmentation is shown with the pseudo-color images, where each region is represented by a unique color. As observed, MLASC-O, KASP, and NCut can all achieve sound segmentation while the Nystrom method produces poor results. The segmentation quality is confirmed by the Q-value and benchmarks in Table 1. Regarding the running time, MLASC-O is the fastest among the five, also reported in Table 1.

Next, we evaluate our method on medium-size images with about 150 K pixels. Note that Ncut requires high spatial cost of storing the affinity matrix and the graph Laplacian matrix. Specifically, each matrix needs $(321 \times 481)^2 \times 4$ bytes, approximately 90 GB. Though the cost of storage can be reduced by using sparse matrices, 3G RAM is still far too small, and we did not evaluate Ncut on the medium-size images. In the following, we compare the performance of the four algorithms, i.e., MLASC-O, Nystrom, INystrom, and KASP and show the experimental results in Fig. 4. As seen, MLASC-O provides more reasonable segmentation results than other methods on all the testing images. We notice INystrom usually gives comparable segmentation results, however, sometimes it breaks one object into a few pieces. For example, on “Elephant”, the sky is split into three parts, and some areas of the elephant and the sky are assigned to the grass region. Similarly, the sky is broken on the “Building” image. We also observe that the segmentation results by Nystrom and KASP frequently mix the foreground and background objects. The quantity comparisons are reported in Table 2. MLASC-O achieves best results in most cases, which is consistent with our visual observation.

In Table 3, we provide the comparison of the running time. Clearly, MLASC-O is the fastest among the four algorithms, followed by Nystrom. INystrom and KASP require much more time, i.e., over 100 s are needed to segment one image, mainly due to the extra time required in the quantization step.

### 4.2.2. Experiments on large-size datasets

Finally, we evaluate the performance of MLASC-A on large-size images. Fig. 5 shows the segmentation results on benchmark images with 3 million pixels: in each row, the first one is the original image, the second is the mean-filled image, and the third is the segmentation. The “mean-filled” image is obtained by filling each segmented region with the mean color of the corresponding region in the original image. MLASC-A gains solid segmentation results for all the testing images: the foreground objects, like the horses, stones, and zebras are clearly separated from the background, and the segmentation quality is confirmed by quantitative evaluation metrics in Table 4. The running time is also reported. Notice that our method is very efficient, and can process a 3M image in 90 s.

We also evaluate MLASC-A on real HD images with about 2 million pixels. Our method presents good performance on all testing images, shown in Fig. 6. This is confirmed by the low Q-value, 0.009, in average. The running time is about 60 s per image. It is worth pointing out that the use of the approximation subspace in MLASC-A makes it possible to include a good number of sample pixels ($c = 1000$) in the computation, an indispensable step to segment a large-size image as a whole.

### 5. Conclusion

In this paper, we present Multi-level Low-rank Approximation-based Spectral Clustering (MLASC) for high-resolution image segmentation. Compared with the existing methods, MLASC gains high efficiency in both computational time and space. First, the approximation to the affinity matrix and its subspace makes it possible to sample sufficient pixels to accurately approximate the subspace of the Laplacian matrix and its eigenvectors, and thus achieve good segmentation. Second, with the proposed nearly-optimal sampling strategy, MLASC can quickly select data samples from a large-scale dataset. Empirically, we demonstrate that MLASC outperforms leading approximation-based spectral clustering in terms of accuracy, approximation error and running time.

### Appendix A. Proof of Proposition 1

**Proposition 1**. Since the values of $S$ represent the similarity between data points, the higher value indicates closer distance. Consequently, the diagonal elements in $S$, which represent the similarities between one data point and itself, take the highest value of $S_{ii}$, i.e., $S_{ii} \geq S_{ij}$, $\forall i, j \in [1, n]$. Thus,

$$p_i = \frac{1}{n} \sum_{j=1}^{n} \frac{S_{ij}}{n} \sum_{j'=1}^{n} \frac{S_{ij'}}{n} \geq \frac{\sum_{j=1}^{n} S_{ij}}{n^2} = \frac{1}{n^2} \sum_{j=1}^{n} ||S(:, i)||_F^2,$$

(A.1)

where $\beta = \frac{\sum_{j=1}^{n} S_{ij}}{n^2}$. Because $S_{ii} \geq S_{ij}$, $\forall i, j \in [1, n]$, we get $\beta \leq 1$. In addition, we have $\sum_{i=1}^{n} p_i = 1$. Hence, $\{p_i\}_{i=1}^{n}$ are nearly-optimal probabilities for sampling the columns of $S$. The proof is completed.

### Appendix B. Proof of Proposition 3

**Proof 2**. Let $\{p_i^{(1)}\}_{i=1}^{n}$ denote the optimal probabilities for sampling the columns of $L$, where

$$p_i^{(1)} = \frac{||L(:, i)||_F^2}{\sum_{j=1}^{n} ||L(:, j)||_F^2}. \quad (B.1)$$

Then, with Eq. (16), we get

$$p_i^{(1)} = \frac{\sum_{i=1}^{n} S_{ij}}{n \sum_{i=1}^{n} S_{ij}^2} \leq \frac{1}{d_{\max}} \sum_{j=1}^{n} \frac{S_{ij}^2}{d_{\max}} = d_{\max} \sum_{j=1}^{n} ||S(:, j)||_F^2,$$

where $d_i$ is the $i$th degree, $d_{\min}$ is the minimum of $d_s$, and $d_{\max}$ is the maximum of $d_s$. Let $\beta = \frac{d_{\min}}{d_{\max}}$ ($\beta \leq 1$), then

$$p_i^{(1)} = \frac{||S(:, i)||_F^2}{\sum_{j=1}^{n} ||S(:, j)||_F^2} \geq \beta p_i^{(1)}. \quad (B.2)$$

Hence, $\{p_i^{(1)}\}_{i=1}^{n}$, Eq. (18) are nearly-optimal probabilities for sampling the subspace in the Laplacian domain.

For the nearly optimal probabilities computed by Eqs. (1, 5), we get

$$\tilde{p}_i \geq \beta_1 \frac{||S(:, i)||_F^2}{\sum_{j=1}^{n} ||S(:, j)||_F^2} \geq \beta_1 \beta p_i^{(1)}. \quad (B.3)$$

Since $\beta_1, \beta \leq 1$, so $\{p_i^{(1)}\}_{i=1}^{n}$ are nearly-optimal for sampling in the Laplacian domain.

### Appendix C. Proof of Proposition 4

**Proof 3**. $S$ is computed as a symmetric semi-positive definite matrix by a non-linear kernel metric, so $L$ is a symmetric semi-positive definite matrix since given any vector $\tilde{y}$.
\[ y^T L y = y^T D^{-1/2} S D^{-1/2} y = y^T S y \geq 0, \]  

(C.1)

where \( \tilde{y} = y D^{-1/2} \) is a \( 1 \times n \) vector. As a result, the eigenvalue decomposition of \( I \) is also a singular value decomposition. According to Algorithm 1, the eigenvector matrix \( U \) is computed from \( \tilde{L}_{	ext{opt}} \), which is constructed under the nearly-optimal probabilities, so the top \( k \) singular vectors of \( I \) can be approximated by \( U \). □

References


