Fast, Accurate Spectral Clustering Using Locally Linear Landmarks

Max Vladymyrov
Google, Inc.
Email: mxv@google.com

Miguel Á. Carreira-Perpiñán
EECS, UC Merced
Email: mcarreira-perpinan@ucmerced.edu

***Abstract*—For problems of image or video segmentation, where clusters have a complex structure, a leading method is spectral clustering. It works by encoding the similarity between pairs of points into an affinity matrix and applying k-means in its low-order eigenspace, where the clustering structure is enhanced. When the number of points is large, an approximation is necessary to limit the runtime even if the affinity matrix is sparse. This is commonly done with the Nyström formula, where one solves an eigenproblem using affinities between a subset of the data points (landmarks) and then estimates the eigenvectors over the entire data by interpolation. In practice, this can still require many landmarks to achieve reasonably accurate solutions, and applies only for explicitly defined affinity kernels. In this paper we propose two ideas: the Locally Linear Landmarks technique, where one solves a reduced spectral problem over landmarks that involves the entire, original affinity matrix; and a fast, good initialization for k-means. We show both approximation error and runtime are considerably reduced, even though fewer landmarks are used. We apply it to spectral clustering and to several variants of it that involve complex affinities: constrained clustering, affinity aggregation, neighborhood graphs based on tree ensembles, and video segmentation.**

Spectral clustering (SC; [28], [36]) is a popular method that has been used for many problems in machine learning and computer vision, such as image or motion segmentation [28], [12], [25]. It uses graph structure that can discover complex clusters in data better than model-based algorithms such as Gaussian mixture. The use of pairwise affinities allows for construction of different similarity measures, ranging from simple Gaussian affinities based on the pixel location and color and/or texture, to using edge and contour information or approaches involving information at multiple scales and semisupervised learning [28], [18], [22], [16]. The clustering structure in the original space is amplified in the projections of the data on the eigenspace of the graph, so that a simple clustering algorithm (e.g. k-means) can help locating the clusters efficiently.

One disadvantage of SC is its computational cost. With large number of points N, computing and storing a full $N \times N$ affinity matrix is impractical. Using a sparse matrix reduces the complexity, but the runtime can still be high for large N, because of the need to solve a sparse large eigenproblem. In addition, for high-dimensional input data, achieving good clustering solutions requires the use of a neighborhood graph because of the concentration of measure. In this case, one seeks an approximate solution. The most common way to do this is based on subsampling. Here, one selects a small subset of points (landmarks) from the input, solves the eigenproblem for them, and approximates the full eigenproblem with an out-of-sample mapping, most commonly the Nyström formula [37], [3]. By using sufficiently many landmarks, the approximation error can be reduced to an acceptable level while keeping the runtime lower than that of the full problem.

One problem with the subsampling approach is that *the solution for the landmarks is obtained using only the affinities between the landmarks themselves, while the rest of the points are used only in the out-of-sample formula*. Hence, for few landmarks the eigenproblem solution will have a large error, and *so will the extension to all the points*. One can use more landmarks, but at the cost of larger runtime. This is also problematic if a user wants to use a small bandwidth in the affinity kernel, because in order for two landmarks to interact they have to be within reach of each other. Additionally, the Nyström method *requires a continuous kernel function that generates the affinity matrix*, which is not always available. Many of the popular variants of SC, such as proximity graphs [7], constrained spectral clustering (CSC; [20]) and affinity aggregation for spectral clustering (AASC; [15]) are using custom affinity matrices that are not generated from a continuous kernel.

A different way to avoid this problem while still using a small number of landmarks is to use the connection between pairs of points, not just between the landmarks. This was proposed by Vladymyrov and Carreira-Perpiñán [35] in the context of manifold learning. Their locally linear landmarks (LLL) method solves a reduced spectral problem for landmarks, where the "reduced affinities" involve the entire original affinity matrix, rather than just the affinities between the landmarks. This is useful for problems with big and sparse affinity matrices, for which subsampling methods would need a lot of landmarks to get the structure right.

In this paper we analyze the application of LLL algorithm for different variations of spectral clustering. We show that the algorithm is particularly beneficial for problems of image and motion segmentation, with speed and accuracy benefits that sometime larger than LLL has for manifold learning, which is the original application of the algorithm. We also show that LLL can be used for effective approximation of custom non-Gaussian affinity matrices. Such affinities arise in order to improve the quality of the Gaussian affinities [7], add

Part of this work was done while M.V. was a PhD student at UC Merced.

additional must- and cannot-link constraints [20] or aggregate different features into one unified affinity [15]. For each of those affinities we derive a custom fast and accurate algorithm that benefits from the structure of the approximation used by LLL. For these custom-based affinities, it is difficult to apply Nyström and other approximations. Based on our knowledge, we are the first one to propose to scale up affinity aggregation and constrained SC algorithms.

We analyze the assumptions of LLL for the context of SC. Different from Vladymyrov and Carreira-Perpiñán [35] who concentrated on the manifold learning, the structure of the data in clustering is very different. While in manifold learning it is ok for the clustering information to be smoothed out in order to reveal the manifold structure, in clustering it is important to emphasize the discontinues between points from different clusters. In section IV and the experiments we validate the assumptions imposed in LLL for SC.

Finally, we also provide a simple but very effective way to accelerate the k-means step by using a good initialization based on the landmarks' solution. Our approach is robust, parameter-free and fast, essentially reducing the cost of the algorithm to a single k-means iteration over the full dataset.

In the experiments we show that our approach gives faster yet accurate solution with respect to the both exact SC and its variations and compares favorably to the Nyström extension. It is able to efficiently solve a motion segmentation problem using spatio-temporal affinities with $N = 787\,200$ variables. It took the algorithm under 10 minutes to get meaningful solution, while for Nyström we were not able to get good results at all.

I. The Spectral Clustering Problem

There exist several formulations of SC, depending on what kind of affinities are used, type of normalization applied to the graph Laplacian, etc [36]. Here we use the most common one, which approximates the normalized cut criterion [28]. Given a (sparse) affinity matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ defined on data points $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$ of $D \times N$:

1) Find K dimensional spectral embedding:

$$\min_{\mathbf{X}} \operatorname{tr}\left(\mathbf{X}\mathbf{L}\mathbf{X}^T\right), \text{ s.t. } \mathbf{X}\mathbf{D}\mathbf{X}^T = \mathbf{I},\ \mathbf{X}\mathbf{D}\mathbf{1} = \mathbf{0}, \qquad (1)$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is a graph Laplacian defined for a degree matrix $\mathbf{D} = \operatorname{diag}\left(\sum_{m=1}^{N} w_{nm}\right)$ and $\mathbf{X} \in \mathbb{R}^{K \times N}$ is a data projection.

2) Obtain the final clustering by running k-means on the normalized projections $\hat{\mathbf{X}}_{ij} = \mathbf{X}_{ij} / (\sum_j \mathbf{X}_{ij}^2)^{1/2}$.

The solution to the spectral embedding problem (1) is given by $\mathbf{X} = \mathbf{U}_K^T \mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{U}_K = (\mathbf{u}_1, \dots, \mathbf{u}_K)$ are $2, \dots, K+1$ trailing eigenvectors of the $N \times N$ matrix $\mathbf{C} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}}$. This eigendecomposition is a bottleneck and scales as $\mathcal{O}(N^3)$. In case of sparse affinities and small number of clusters K, one can possibly use iterative methods, such as power method or Krylov subspace methods [13], however the complexity still can be expensive.

Our goal is to approximate the solution of this spectral problem, and thus to achieve a fast approximation to final clustering result. As we will show, we can also speed up step 2 with no error.

II. Related Work

The most widespread approach to approximate the solution of a spectral problem, in particular spectral clustering, is to solve the problem for a subset of points (landmarks) and then to estimate the solution for the remaining points. For example, the Nyström formula [37], [3] has been widely applied to approximate the solution of SC [4], [12]. The formula nonlinearly interpolates the landmarks using the kernel that was used to construct the affinities. The projection $\mathbf{x}$ of a new point $\mathbf{y}$ can be found using the interpolation $\mathbf{x}_k = \frac{1}{\lambda_k} \sum_{n=1}^{L} \mathbf{X}_{nk} K(\mathbf{y}, \mathbf{Y}_i)$, where an out-of-sample kernel K is defined as $K(\mathbf{a}, \mathbf{b}) = \widetilde{K}(\mathbf{a}, \mathbf{b}) / \sqrt{\mathbf{D_a} \mathbf{D_b}}$, with $\widetilde{K}(\mathbf{a}, \mathbf{b})$ as a data-dependent kernel (e.g. Gaussian) and $\mathbf{D_a} = \sum_{n=1}^{L} \widetilde{K}(\mathbf{a}, \mathbf{y}_n)$, $\mathbf{D_b} = \sum_{n=1}^{L} \widetilde{K}(\mathbf{y}_n, \mathbf{b})$.

Nyström formula gives fast results, with cost being proportional to $\mathcal{O}(KL^2)$ to compute the embedding of landmarks and $\mathcal{O}(NLK)$ to compute the projection of N out-of-sample points. However, the quality of the method is not the best, since landmarks' projection only uses a small subsample of the affinity matrix, basically ignoring the rest of the available information. In addition, the algorithm relies on the existence of the data-dependent kernel $K(\mathbf{a}, \mathbf{b})$ which is not always available.

K-means-based approximate spectral clustering (KASP; [38]) replaces the original data with a subsample, while maintaining the mapping between the original data and the subsample points. Then, the SC is run on the subsample and the final clustering assignment to the original points comes from the cluster assignment of the corresponding subsample point. For the subsampling algorithm Yan et al [38] chose k-means and Random Projection trees. Similar to Nyström, the subsample matrix does not use the structure of the original data.

Laplacian eigenmaps latent variable model (LELVM; [6]) approximates SC based on the idea of adding new point to the subset embedding and solving the spectral problem for that new point. The final formula takes the form of a Nadaraya-Watson estimator jointly constructed on the feature vectors and embedding projections.

Landmark Spectral Clustering (LSC; [8]) uses the Anchor-Graphs [19], where a smaller affinity matrix is built between landmarks and points that approximate a full $N \times N$ affinity. This approach, however, does not solve the problem (1), in the sense that it does not approximate a *given* spectral problem and thus the clustering defined by it.

Multi-grid spectral clustering methods [9], [17] approximate the eigendecomposition specifically for the problem of image segmentation. The eigenspectrum of the Laplacian is defined using a random chain matrix expressed as Markov chain propagation model with a certain probability distribution, which can be fine scaled using a kernel matrix. Multi-scale approximation [10] was also proposed in the context of image

segmentation and is restricted to data defined on an image grid. It approximates a dense affinity matrix as a weighted sum of affinities connected at different scales, which allows faster computation. Maire and Yu [21] combine both multi-scale and multi-grid techniques together. These algorithms are specialized for data having a form of an image grid and may not apply to unstructured, high-dimensional data. In comparison to the proposed LLL-based algorithm, they do not benefit from savings when having to solve multiple SC problems on the same dataset but with different affinities (as in affinity aggregation or model selection over the affinity parameters). In addition, these methods are far more complex to implement than Nyström or LLL, and require several user parameters that are nontrivial to set (e.g. scale splits and their weights in multi-split SC or initial kernel size, coarse schedule for multi-grid methods).

III. Accelerating the Spectral Clustering Using Locally Linear Landmarks

LLL approximates (1) by constructing matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ defined for a set of landmarks, but using the structure of the whole data. For $L \ll N$ landmarks $\widetilde{\mathbf{Y}} = (\widetilde{\mathbf{y}}_1, \dots \widetilde{\mathbf{y}}_L)$ from $\mathbf{Y}$, LLL approximates each point as a linear combination of K_Z nearest landmarks $\mathbf{Y} \approx \widetilde{\mathbf{Y}}\mathbf{Z}$, where a reconstruction matrix $\mathbf{Z} \in \mathbb{R}^{L \times N}$ is a column matrix of the nearest K_Z landmarks of each point (K_Z nonzero entries per column). Thus, this approximation preserves the locality of the data and the matrix $\mathbf{Z}$ corresponds to the proximity of the data points to the nearby landmarks. We can solve for $\mathbf{Z}$ using $\min_{\mathbf{Z}} \|\mathbf{Y} - \widetilde{\mathbf{Y}}\mathbf{Z}\|^2$, s.t. $\mathbf{1}^T\mathbf{Z} = \mathbf{1}^T$, where the constraint comes from the translational invariance of $\mathbf{Z}$. The solution to this optimization problem is found for each point separately by first solving the linear system $\sum_{k=1}^{L}(\mathbf{y}_n - \widetilde{\mathbf{y}}_i)(\mathbf{y}_n - \widetilde{\mathbf{y}}_j)z_{nk} = \mathbf{1}$ and then rescaling the weights so they sum to one.

Further, LLL assumes that this local reconstruction matrix $\mathbf{Z}$ also approximates the points in the projection space:

$$\mathbf{X} \approx \widetilde{\mathbf{X}}\mathbf{Z}. \tag{2}$$

Now, the spectral embedding problem (1) takes a form

$$\min_{\widetilde{\mathbf{X}}} \operatorname{tr}\left(\widetilde{\mathbf{X}}\widetilde{\mathbf{A}}\widetilde{\mathbf{X}}^T\right) \text{ s.t. } \widetilde{\mathbf{X}}\widetilde{\mathbf{B}}\widetilde{\mathbf{X}}^T = \mathbf{I}, \tag{3}$$

with $L \times L$ *reduced affinity* matrices

$$\widetilde{\mathbf{A}} = \mathbf{Z}\mathbf{L}\mathbf{Z}^T, \qquad \widetilde{\mathbf{B}} = \mathbf{Z}\mathbf{D}\mathbf{Z}^T. \tag{4}$$

This problem has the same form as the original problem (1), but defined on a smaller-size reduced affinities $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$. LLL basically solves a smaller embedding problem for the landmarks using smaller reduced affinities that incorporate the structure of the original affinity. After the landmarks are projected, the rest of the points can be found using reconstruction (2). Notice, that $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ do not have the same form as the original matrices $\mathbf{L}$ and $\mathbf{D}$. For example, $\mathbf{A}$ may or may not have a form of graph Laplacian, $\mathbf{B}$ may not be diagonal – those matrices just approximate $\mathbf{L}$ and $\mathbf{D}$.

The cost of LLL is based on four parts: computing $\mathbf{Z}$ is $\mathcal{O}(NDK_Z^2)$; $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ are $\mathcal{O}(cK_ZN)$, where c depends on the sparsity of $\mathbf{W}$; K eigenvectors takes $\mathcal{O}(KL^2)$; and the final projection (2) is $\mathcal{O}(LdN)$. The total cost is $\mathcal{O}\left((K_Zc + Ld + DK_Z^2)N + KL^2)\right)$ which is linear in N.

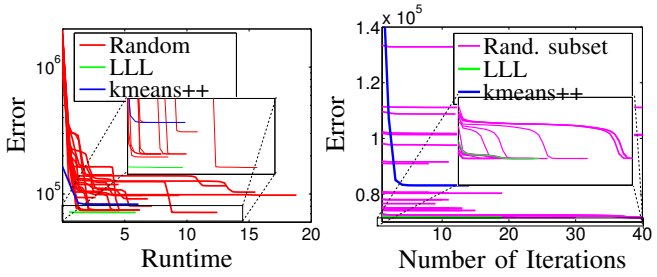

Fig. 1. *Left:* 20 runs of k-means initialized at random (red), with LLL (green) and with k-means++ (blue). *Right:* k-means initialized with LLL landmarks and 50 other subsets of L points.

As one can see from the form of the approximation (3) LLL requires feature vectors, but the majority of (spectral) clustering applications do use features.

a) Accelerating the k-means clustering step: LLL approximation gives us the K-dimensional projections $\mathbf{X}$ of the original data $\mathbf{Y}$ (using (2)). Next step is to run k-means on $\mathbf{X}$ to obtain the final K clusters. This problem is non-convex and, to avoid local optima, a common strategy is to use multiple restarts with different initializations and pick the result with the lowest error. We can speed up the k-means step by capitalizing on the fact that the landmarks themselves provide a reasonable clustering, and there are fewer landmarks than points. Instead of running k-means for the whole dataset m times, we run it just on the landmarks, pick the best solution and feed its centroids as an initialization to k-means on the entire data. Thus, k-means is run just once on the full problem, but with an initialization that is much better than random. The cost is $\mathcal{O}(mLK^2 + NK^2)$ and when $L \ll N$ the speed-up over the $\mathcal{O}(mNK^2)$ of the naive k-means step is proportional to m. Also, we can afford to run many restarts to increase the chance of finding a good local optimum.

The idea of seeding for initialization of k-means is not new [11], [30]. However, our approximation makes sense in the context of LLL, since instead of the random sample of points, our sample comes from the landmark location. Those landmarks are acting as the reference points to the rest of the data for the local linear reconstruction (2). Therefore, the landmarks have more chance to be inside the cluster than on its boundary.

Fig. 1 on the left shows the decrease of k-means objective function for the clustering $N = 787\,200$ points to 6 classes using LLL (see motion segmentation example below). Red curves correspond to randomly initialized restarts of full k-means and a green curve is a single full k-mean initialized from the best of $m = 20$ different restarts of k-means on LLL landmarks. We also compare to the kmeans++ [2], which picks centroids greedily one at a time so they are widely separated. Overall it took 0.5 seconds to run k-means using LLL landmarks and 5.3 seconds for a single run of a full k-means. In comparison, 20 restarts of full k-means took 91.3 seconds. On the inset we can see that only one out of 20 runs

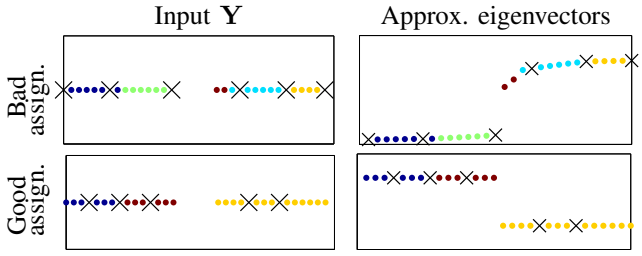

Fig. 2. *Left column:* dataset $\mathbf{Y}$ with 5 landmarks selected far from the cluster boundary (top row) and closer (bottom row). *Right column:* final eigenvectors form LLL. The color of the points are the same if they share two nearest landmark assignment.

of k-means converged to that solution. This means that using smaller number of restarts m we could end up only with more expensive, but also worse solution that using LLL landmarks. In the right plot of fig. 1 we show k-means error using LLL landmarks versus k-means using 50 different subsets of L random points from $\mathbf{Y}$. While LLL landmarks do not give the best result among random subsets (see the inset), the error is lower then most of the random subsets.

IV. Is Local Linearity a Good Assumption for Clustering?

Comparing to the original motivation of Vladymyrov and Carreira-Perpiñán [35] that propose LLL to be used for manifold learning, the desired properties of SC in the projection space are different. For clustering, we do not need to preserve the manifold structure, but rather separate the points into clusters. Let us consider the following ideal clustering situation with SC. The eigenvectors are piecewise constant with same values for points in each cluster. A clustering solution is also a piecewise constant function. The approximate solution (from LLL) for the landmarks' eigenvectors is piecewise constant as well, corresponding to the same clusters. The LLL out-of-sample formula (2) is a linear combination of the nearest landmarks' eigenvector value. Then we have two cases: (1) if the nearest landmarks to the test point are in the same cluster, their eigenvectors values ("cluster labels") are equal and (2) will predict that same eigenvector value, which is correct. This will happen with the test points in the "interior" of each cluster, i.e. "nearly everywhere" if the cluster boundaries are narrow. (2) if the nearest landmarks to the test point are not all in the same cluster, their eigenvector values ("cluster labels") are not all equal and the formula will predict an "average" of those values, giving more weight to the landmarks that are closer to the test point. The resulting predicted eigenvector value will be smoothed out, since it averages different labels. This will happen with test points in the "boundary" between different clusters. How narrow are the cluster boundaries depends on the number of landmarks. The more landmarks, the narrower the boundaries, since this is the region where points have nearest neighboring landmarks from different clusters.

In fig. 2 we show the example of two cases above. The dataset on the left consists of points along a line separated into two clusters. Exact SC solves this problem given sufficiently narrow bandwidth of the kernel, such that the points on the side of one cluster won't be affected by the points on the side of the other. For LLL, there is one more constraint. If the test point is reconstructed by landmarks from different clusters (such as dark red points in the top plots) the target eigenvector will be smoothed out. Good assignment (as the one on the bottom), gives much better approximation.

We expect the a piecewise linear model of LLL to match perfectly the piecewise constant clustering model except at discontinuities (cluster boundaries). These points are difficult for SC in the first place, i.e. their exact eigenvector value will be less "pure" than at points in the interior of a cluster. LLL might have troubles with small clusters, because they will get fewer landmarks. However, SC tends not to produce small clusters, since it approximates the Normalized Cut objective function [28], which discourages this.

V. Using More Complex Affinities

One of the important benefits of LLL is that it can use any affinity matrix and with any explicit or implicit kernel that generates it. Comparing to that, Nyström method requires well defined data-independent kernel function that generates the affinities [4], and, to our knowledge, was applied only to the Gaussian kernel [32], [12]. Landmark Spectral Clustering [8] is also not applicable because their approximation constructs their own affinity matrix. Finally, the vector quantization techniques (such as the algorithm of Yan et al [38]) also cannot be easily applied for this problem, since the affinities are built between the subset of points, while in this section we consider specifically the affinities that involve the whole dataset.

Apart from expensive eigendecomposition that can be alleviated with LLL, the custom affinities involve large overhead in their computation, which can become a bottleneck by itself. For each of those cases we investigate how the structure of LLL can help the computation of the affinities and propose custom algorithm that can reduce the cost of both the computation of the affinities and the eigendecomposition. Our analysis comes naturally given the structure of LLL and does not involve additional approximations. Based on our knowledge, the algorithms below have been never applied on large matrices before. Slightly abusing the notation, in this section we are going to call $\overline{\mathbf{W}}$ the modified affinities of currently described algorithm.

A. Constrained Spectral Clustering

In constrained spectral clustering (CSC) user additionally provides *must* and *cannot* link constraints between some of the data points. Lu and Carreira-Perpiñán [20] propose the algorithm for including this information inside the affinity matrix. Original affinity information is assumed to be interpreted as a covariance matrix of a zero-mean Gaussian process, which acts as a prior on the labels. The pairwise constraints serve as noisy observations. The modified affinities are then computed using the Bayes' rule as the posterior probability of the labels given the pairwise constraints. The final expression for the modified affinities:

$$\overline{\mathbf{W}} = (\mathbf{W}^{-1} + \mathbf{M})^{-1} = \mathbf{W} - \mathbf{W}(\mathbf{I} + \mathbf{M}\mathbf{W})^{-1}\mathbf{M}\mathbf{W}, \quad (5)$$

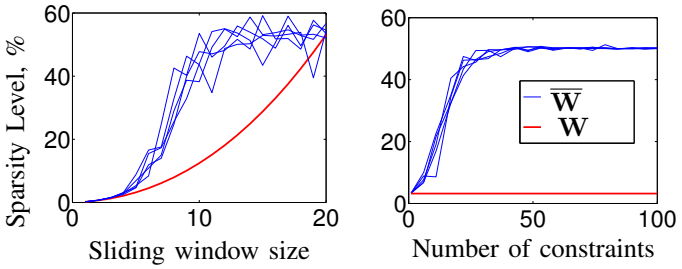

Fig. 3. Sparsity of the original $\mathbf{W}$ and constrained $\overline{\mathbf{W}}$ affinity matrices for 64×64 cameraman image. *Left:* as a sliding window increases with 10 random constraints (must and cannot), *right:* as a number of constraint increase with the sliding window radius equals to 5. Experiments were repeated 5 times with different random sets of constraints.

where $\mathbf{M}$ contains the constraint information:

$$\mathbf{M}_{ij} = \begin{cases} \frac{m_i}{\epsilon_m^2} + \frac{c_i}{\epsilon_c^2} & \text{if } i = j \\ -\frac{1}{\epsilon_m^2} & (i,j) \in \mathcal{M} \\ \frac{1}{\epsilon_c^2} & (i,j) \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}.$$

Here m_i and c_i are the number of must- and cannot-links respectively for point i, ϵ_m and ϵ_c are the importance of the constraints with respect to the affinity (with $\epsilon \to 0$ implying harder constraints).

The modified affinity $\overline{\mathbf{W}}$ is able to capture constrains well with two drawbacks: (1) computing the inverse in (5) scales as $\mathcal{O}(N^3)$ and (2) $\overline{\mathbf{W}}$ is much denser than $\mathbf{W}$.

The first problem can be solved efficiently by reorganizing the points, such that the constrained ones appear first: $\mathbf{M} = \left(\begin{smallmatrix} \mathbf{M}_{1:2R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right)$, where R is the number of constraints and $\mathbf{M}_{1:2R}$ has a size of at most $2R \times 2R$. Then, $\mathbf{I} + \mathbf{MW} = \left(\begin{smallmatrix} \mathbf{M}_{1:2R}\mathbf{W}_{1:2R}+\mathbf{I}_{1:2R} & \mathbf{M}_{1:2R}\mathbf{W}_{2R+1:N} \\ \mathbf{0} & \mathbf{I}_{2R+1:N} \end{smallmatrix}\right)$, where we split $\mathbf{W} = \left(\mathbf{W}_{1:2R}\mathbf{W}_{2R+1:N}\right)^T$. Now, for the inverse: $\left(\begin{smallmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)^{-1} = \left(\begin{smallmatrix} \mathbf{A}^{-1} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$. Thus, the only matrix to be inverted is a small $2R \times 2R$ matrix $\mathbf{A} = \mathbf{M}_{1:2R}\mathbf{W}_{1:2R} + \mathbf{I}_{1:2R}$, which is always invertible due to the added identity.

The second problem makes things complicated. Adding many constraints to the original affinity can make it much denser than the original one, as a result of which not only the subsequent eigendecomposition is much more complicated to do, but even storing the matrix can be problematic. In fig. 3 we show how the sparsity level of $\overline{\mathbf{W}}$ changes with respect to two different variables: the sparsity of $\mathbf{W}$ and the number of constrains R. For both cases the constrained affinity matrix becomes dense very fast.

With LLL we can a huge benefit of computing the constrained affinity $\overline{\mathbf{W}}$ without ever evaluating the original affinity $\mathbf{W}$, thus removing both of the problems above. Using (5) we can expand the reduces affinities (4) as

$$\begin{aligned}\mathbf{Z}\overline{\mathbf{L}}\mathbf{Z}^T = \mathbf{Z}\overline{\mathbf{D}}\mathbf{Z}^T - \mathbf{Z}\overline{\mathbf{W}}\mathbf{Z}^T = \mathbf{Z}\,\mathrm{diag}\left(\mathbf{W1}\right)\mathbf{Z}^T \\ - \mathbf{Z}\,\mathrm{diag}\left(\mathbf{WQW1}\right))\mathbf{Z}^T - \mathbf{ZWZ}^T + \mathbf{ZWQWZ}^T,\end{aligned}$$

where $\mathbf{Q} = (\mathbf{I} + \mathbf{MW})^{-1}\mathbf{M}$ is $N \times N$ matrix, but only with $N \times L$ nonzero rows (due to the sparsity of $\mathbf{M}$). Therefore, instead of computing $\overline{\mathbf{L}}$ and then computing the reduced affinity, we can precompute $\mathbf{ZW}$ in $\mathcal{O}(cNL)$, where c is the constant that depends on the sparsity of $\mathbf{W}$, and solve for diagonal matrix in the second term by multiplying the elements from right to left. Overall, the complexity of computing the affinity is $\mathcal{O}(cNL)$ and it avoids direct computation and storage of dense matrix $\overline{\mathbf{W}}$.

B. Affinity Aggregation for Spectral Clustering

In many situations using just a single affinity matrix built on top of Euclidean distance features does not give satisfactory performance. A practitioner has to try many different features with many different parameters in order to get good performance. Huang et al [15] propose an affinity aggregation spectral clustering (AASC) that combines multiple affinities in a single SC framework. For K affinity matrices $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(K)}$ and the weighting coefficients $\mathbf{v} = (v_1, \ldots, v_k)$ the AASC minimizes the following:

$$\min_{\mathbf{X},\mathbf{v}} \mathbf{X}\overline{\mathbf{L}}\mathbf{X}^T, \text{s.t. } \mathbf{X}\overline{\mathbf{D}}\mathbf{X}^T = \mathbf{I}, \mathbf{v}^T\mathbf{1} = 1, \qquad (6)$$

where $\overline{\mathbf{L}} = \overline{\mathbf{D}} - \overline{\mathbf{W}}$ and $\overline{\mathbf{W}}$ is an aggregated affinity matrix $\overline{\mathbf{W}} = \sum_{k=1}^K v_k^2\mathbf{W}^{(k)}$. To solve (6) the algorithm alternates between minimizing over $\mathbf{v}$ and $\mathbf{X}$. With respect to $\mathbf{v}$, the problem is a 1D root-finding problem that can be solved with a few Newton iterations. With $\mathbf{v}$ fixed, the objective function is a standard SC objective with affinity $\overline{\mathbf{W}}$. The algorithm works well in combining affinity matrices resulting from different features, however it is too expensive for large problems, requiring an eigendecomposition of $N \times N$ matrix for every iteration. In addition, the aggregate affinity matrix is denser than the input affinities.

Similarly to the case above, we can use the structure of LLL to solve the problems with dense affinity. The reduced affinity (4) takes the form:

$$\begin{aligned}\widetilde{\mathbf{L}} &= \mathbf{Z}\overline{\mathbf{L}}\mathbf{Z}^T = \mathbf{Z}\overline{\mathbf{D}}\mathbf{Z}^T - \mathbf{Z}\overline{\mathbf{W}}\mathbf{Z}^T \\ &= \mathbf{Z}\left(\textstyle\sum_{k=1}^K v_k^2\mathbf{W}^{(k)}\mathbf{1}\right)\mathbf{Z}^T - \mathbf{Z}\left(\textstyle\sum_{k=1}^K v_k^2\mathbf{W}^{(k)}\right)\mathbf{Z}^T \\ &= \textstyle\sum_{k=1}^K v_k^2\mathbf{Z}\mathbf{D}^{(k)}\mathbf{Z}^T - \sum_{k=1}^K v_k^2\mathbf{Z}\mathbf{W}^{(k)}\mathbf{Z}^T.\end{aligned}$$

We can benefit from precomputing $\mathbf{Z}$ and reduced affinity matrices $\mathbf{ZW}^{(k)}\mathbf{Z}^T$ just once in the beginning. Then, each iteration would just involve recomputing a weighted sum and eigendecomposition of small $L \times L$ matrix.

VI. Experiments

Here we are going to provide an extensive experimental analysis on benefits of LLL for spectral clustering problem. We are going to evaluate the algorithm for a classical settings of image segmentation with Gaussian affinity matrix. Then we will show how LLL can be useful for the problem of model selection of hyper-parameters. After that we are going to describe three different experiments with custom non-Gaussian affinity matrices. In the end, we are going to show the scalability of our approximation by applying it to the motion segmentation problem with $N = 787\,200$ data points.

All the algorithms in this paper use an affinity matrix as an input. Since we cannot explicitly compute the full affinity matrix, we can approximate it with a sparse $N \times N$ matrix

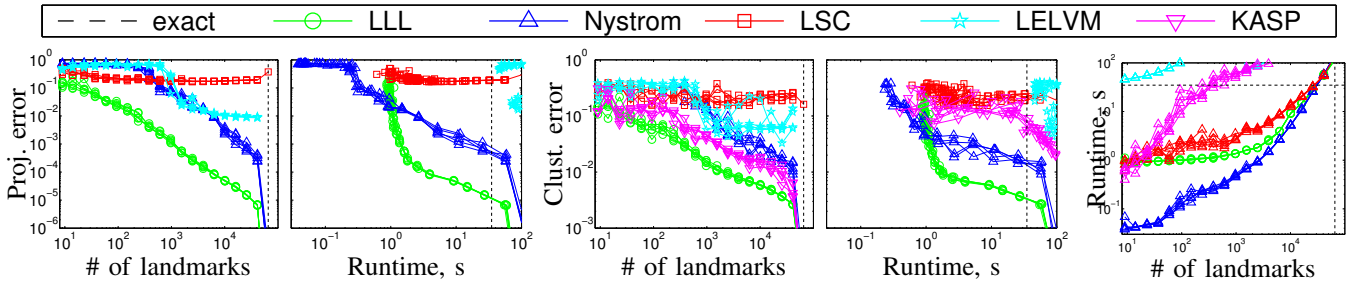

Fig. 4. Image segmentation of 256×256 `cameraman`. Projection and clustering errors with respect to the number of landmarks and the runtime. Dashed vertical black line represents the results of exact spectral clustering.

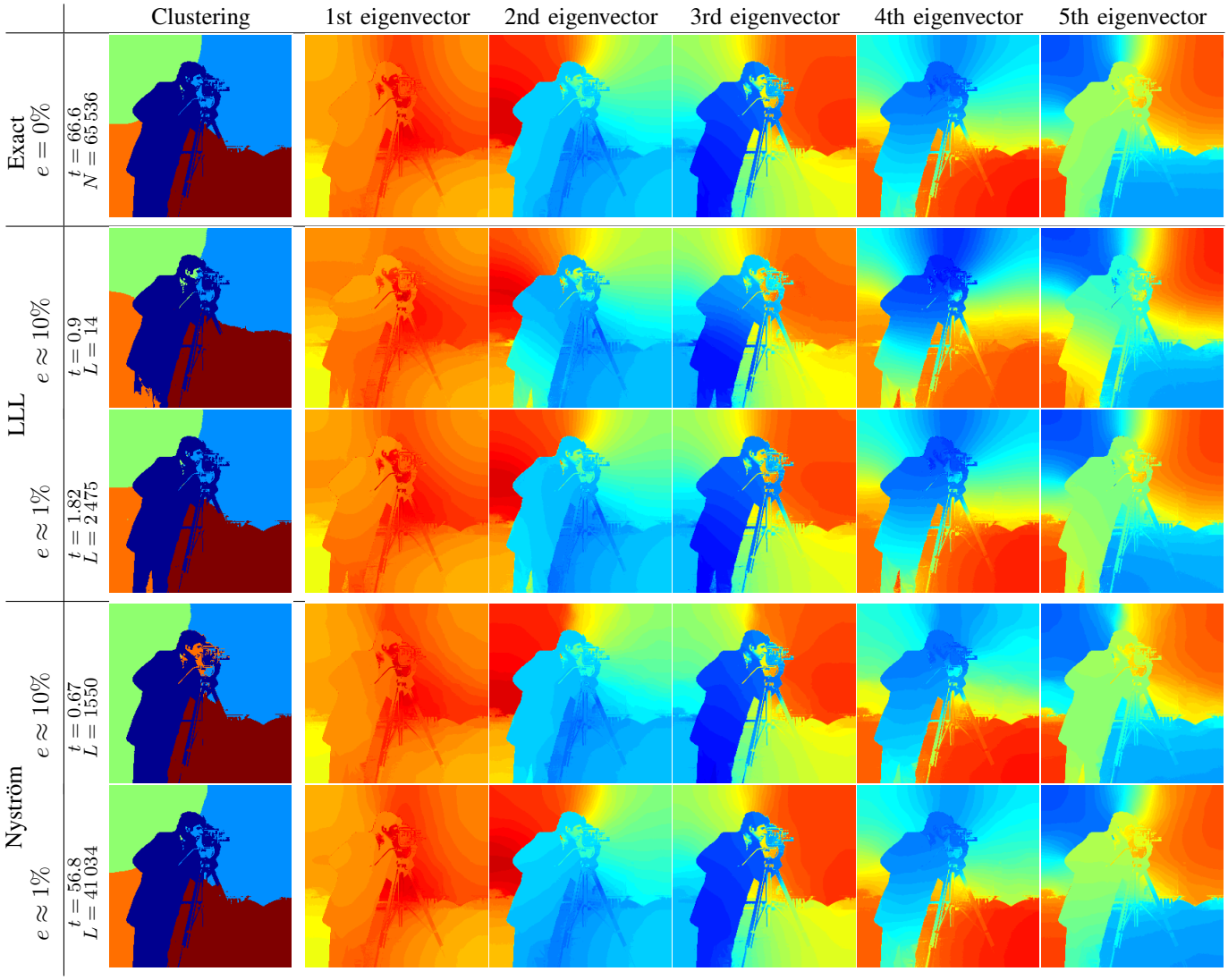

Fig. 5. Clustering results and leading eigenvectors for exact spectral clustering, LLL and Nyström. For each row, left columns give the clustering error e, the runtime t and the number of landmarks L needed to produce the results.

computed e.g. with a neighborhood graph. Constructing such a graph requires finding nearest neighbors for every point, which is costly. However, for some cases this matrix can be approximated using a domain knowledge. For example, for image segmentation the features can represent color, intensity or texture information, but also a spatial ordering (such as the coordinates of the pixels). Thus, we can compute the approximate nearest neighbors for each point using a *sliding window* approach, where for each pixel we define its neighbors as points that fall in a square window with a side r centered at a given datapoint. We can get additional savings by using similar approach to find nearest landmarks for LLL. Because the number of closest landmarks $K_{\mathbf{Z}}$ is quite low in practice (we used $K_Z = K + 1$, as in Vladymyrov and Carreira-

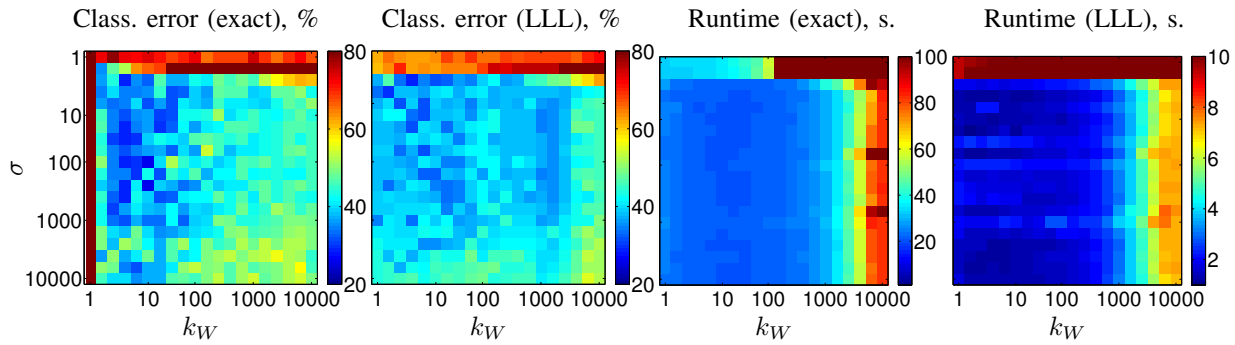

Fig. 6. Model selection of the Gaussian affinities using exact spectral clustering and LLL approximation with different values of bandwidth σ and number of nearest neighbors k_W. First two plots show the classification error and last two show the runtime. Notice that the error pattern in very similar for exact and LLL, but the runtime is an order of magnitude smaller.

Perpiñán [35]), chances are that the closest landmarks can be found within the sliding window neighborhood of a given point. For few points for which this is not the case, we retrieve to computing the distance to all landmarks and truncate it to achieve the K_Z closest ones.

The solution of the spectral embedding (both exact and approximate) results in K-dimensional projections matrix $\mathbf{X}$ (on which k-means is run after the normalization). We characterize the error incurred by our algorithm in three different ways. (1) *Projection error* between the unnormalized exact and LLL eigenvectors, defined as $\|proc(\widetilde{\mathbf{X}}) - \mathbf{X}\|_F / \|\mathbf{X}\|_F$, where we apply Procrustes alignment to remove the influence of translations, rotations and scaling of the eigenvectors. (2) *Clustering error* between final clusters of exact and LLL SC. To align the clusters we used the Hungarian algorithm [5] and report the fraction of misclassified points. (3) *Classification error* (when ground truth is available). We report all errors as % (0% – exact match, 100% – total mismatch). To compute the eigendecomposition we used the MATLAB routine `eigs()`. All the experiments were performed on a single core machine.

A. Classical Spectral Clustering

We show the runtime, the proj. and clust. errors as number of landmarks grow. We compared LLL with Nyström method, Landmark Spectral Clustering (LSC; [8]), LELVM [6] and KASP [38]. We applied the approximations to 256×256 `cameraman` grayscale image ($N = 65\,536, D = 3$) 5 times with different number of randomly chosen landmarks logarithmically spaced from $L = 8$ to $L = N - 10$. We used sparse Gaussian affinities constructed using a sliding window over the pixels with a side $r = 40$ (1 681 nearest neighbors for each point) with $\sigma = 20$. In fig. 4 we show the results. Nyström method gives poor performance for a $L < 200$ because the affinity matrix has isolated points that are not connected to any of the landmarks. As L grows, the error improves, but slowly. LLL gives good performance for any L, with both proj. and clust. error decreasing steadily and consistently with the increase of L.

There are other ways to compute the error (angle between the subspaces defined by the eigenvectors, NMI [31] etc.) We observed that the error that they give is very similar to the one we use.

In fig. 5 we show the final clustering and the largest normalized eigenvectors of the exact clustering and LLL and Nyström approximations that achieve 10% and 1% clustering error. With respect to the exact algorithm, clustering performance of LLL is visually identical for both clustering results and eigenvectors, but with LLL being $30\times$ faster.

B. Model Selection

As a preprocessing step, the spectral clustering algorithm needs to compute the affinity matrix. It should be a good representation of the underlying data, which requires careful parameter selection. For example, for the Gaussian affinities, it is crucial to find a good value of the bandwidth σ as well as the number of neighbors k_W that one wishes to preserve. There is no universal rule that allows a user to pick the best values for those parameters. Ng et al [24] simply suggest choosing them by trying several values until satisfiable result is achieved. Apart from that, couple of heuristic has been proposed that can improve the results. Zelnik-Manor and Perona [39] claim that the results can be improved by setting bandwidth individually per each point of the dataset and computing the Gaussian affinity between points $\mathbf{y}_n$ and $\mathbf{y}_m$ as $w_{nm} = \exp(-\frac{\|\mathbf{y}_n - \mathbf{y}_m\|^2}{\sigma_n \sigma_m})$. The paper suggests to use distance to k nearest neighbor to set individual σ_n, for $n = 1, \dots, N$. Entropic affinities [14], [34] compute individual bandwidth for each point, such that the perplexity, or the effective number of neighbors, is equal to K.

While each of the methods above can result in a good affinity matrix, they all depend on some parameter(s) that need to be tuned (nearest neighbor, perplexity etc.). This can lead to multiple restarts of the whole algorithms, which is expensive. To reduce this cost, we can use LLL for spectral clustering to perform the model selection. Because LLL uses complete affinity matrix, its parameters are also shared between exact method and LLL. Notice that it is not the case for the subset methods (e.g. Nyström), that use smaller affinity matrix which may require different parameter settings for the best performance. The algorithm that we propose is as follows: instead of running full exact spectral clustering several times for different parameter values, we can first try them out with LLL approximation. Then, the best parameters for LLL

will roughly correspond to the best parameters for the exact spectral clustering.

In fig. 6 we show the search for the parameters of the simple Gaussian affinities for $10\,000$ digits from MNIST. We run exact spectral clustering and LLL approximation with 1000 landmarks using 20 different values for σ and k_W chosen on a logarithmic scale from 1 to $10\,000$. Left two plots correspond to the classification error for the exact and LLL spectral clustering respectively. Notice that the error has similar pattern for both exact and approximate clustering. Also, for $k_W = 1$ exact clustering fails, because the affinity matrix decouples into many disconnected components. LLL reduced affinity matrix brings those components together again and gives meaningful error. Right two plots show the runtime of exact spectral clustering and LLL respectively with LLL being more than $10\times$ faster than the exact one (notice different ranges of the color bar).

C. Constrained Spectral Clustering

We applied CSC to a problem of figure/ground segmentation of an image from BSDS500 dataset [1]. To show the scalability, we tested the algorithm using the same image with 3 different sizes: 64×94 small, 160×240 medium and 321×481 large. We used the following bandwidths: $\sigma = 8$ for small, $\sigma = 20$ for medium and $\sigma = 40$ for large image. In all the cases, the sliding window size is equal to σ. We used $L = 730$ landmarks for small image, $L = 2\,100$ for medium and $L = 3\,890$ for large.

We first run the original SC that was able to separate the flower from the background (see fig. 7 for the LLL approximation of the large size image. Then, for the CSC we introduced constraints that separate the petals from the center of the flower (see green and red lines in fig. 7). Below we show the runtimes for both exact and LLL approximations:

	SC		CSC	
Image Size	Exact	LLL	Exact	LLL
Small	4.47	0.87	5.14	0.51
Medium	44	4.66	104.49	6.51
Large	512.01	48.19	–	59.98

LLL gives $10\times$ speed up with almost no compromise on the quality (both proj. and reconst. errors are less then 1% for SC and CSC). For large image exact method run out of memory during the computation of $\overline{\mathbf{W}}$. In comparison, LLL was able to handle large-scale input with no problem.

D. Affinity Aggregation for Spectral Clustering

We tested the algorithm on a subset of CMU-PIE [29], where each point is a 64×64 cropped image of the face in the near frontal position (poses C05, C07, C09, C27, C29). We had $11\,368$ points each with $4\,096$ dimensions that correspond to 67 people. Following Huang et al [15], we used three types of features: Local binary pattern (LBP; [26]) with uniform LBP using 8 neighbors and radius of 1; Gabor texture [23] with 40 filters generated for 5 scales and 8 orientations; and eigenface [33] using top 90% eigenvectors.

All the affinities were computed with Gaussian kernel with $1\,000$ nearest neighbors. To chose the best bandwidth σ we used LLL for the model selection (similar to the way we describe above). We run LLL with $L = 1\,000$ landmarks for 20 different values spaced logarithmically between 10 and 10^7. The best values were: $2.33 \cdot 10^6$ for LBP, 10 for Gabor filter and 42 for the eigenfaces. Notice the scatter of final values. Without LLL, it would be hard to predict those values by random guessing.

In the table I we show the classification error and the runtime of the exact eigendecomposition and LLL approximation using 5 different tries of $1\,000$ randomly chosen landmarks. Using the affinity matrices individually, LLL performs almost as well as exact method, but $10\times$ faster. For AASC the resulting error of LLL is also about the same as the exact method, but $40\times$ faster. In fig. 9 we show the classification error decrease per iterations. LLL follows the exact method closely with a very similar error.

E. Proximity Graph

Carreira-Perpiñán and Zemel [7] define affinity as an ensemble of multiple affinities constructed using minimum spanning tree on a neighborhood graph perturbed with noise. Constructed this way, the affinities tend to be more robust to shortcuts between distant parts of the manifold. In fig. 10 we show the results of the image segmentation for the 512×512 grayscale image of the house using LLL for spectral clustering with $3\,000$ landmarks and $K_Z = 5$. We used a graph ensemble of 10 affinities constructed with MST with every point perturbed using uniform random jitter with standard deviation $0.4\bar{d}$, where $\bar{d}$ is a mean distance to the available neighbors of that point. For comparison, we also show the results of traditional Gaussian affinities (with bandwidth $\sigma = 5$). Both affinities use a neighborhood graph defined with sliding window with $r = 10$. Graph ensemble gives better results than Gaussian affinities. In particular, sky, windows and shadow of the house is better clustered using the former affinities.

F. Motion Segmentation

We also applied SC for motion segmentation in video. We used a dataset of 41 video frames of a person walking around a room. Each frame is represented as 120×160 RGB image. Following [27], we used both spatial and temporal features, with each datapoint given by six coordinates: two for the location on the image plane, one for the number of the frame and three for the color intensity. Overall this gives $N = 787\,200$ points in $D = 6$ dimensions. The neighborhood graph can be easily defined by connecting points along both spatial and temporal domain. For spatial, we used a sliding window with a side $r = 30$. For temporal, we connected each pixel with the same pixel and its adjacent pixels of the previous and the next frames. It gives a neighborhood graph with 963 neighbors for each point. Then we build a Gaussian affinities with bandwidth $\sigma = 15$ for each pair of neighbors. Overall, it took 6.8 minutes to build this affinity matrix.

http://cmp.felk.cvut.cz/multicam/Demos/Students/BorrasJoan/

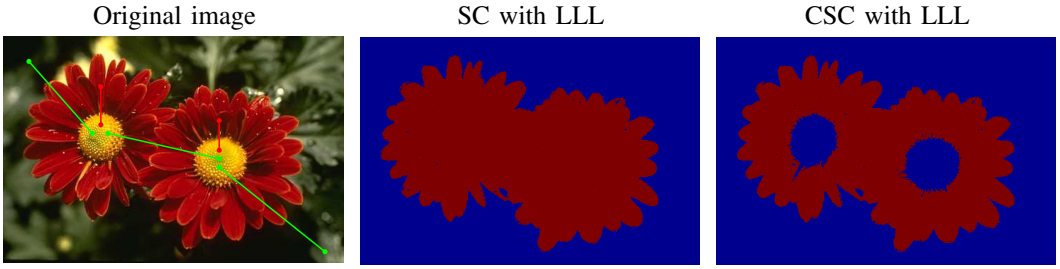

Fig. 7. Figure/ground segmentation of a flower image using SC and CSC (see eq. (5))

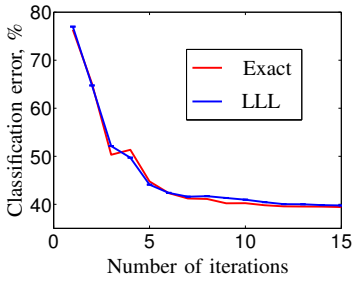

Fig. 9. Classification error for AASC obj. function.

TABLE I
CLUSTERING OF THE 11 368 FACES FROM CMU-PIE DATASET USING SEPARATE FEATURES AND COMBINED FEATURES WITH EXACT SC AND LLL APPROXIMATION. LLL IS AVERAGED OVER 10 RUNS WITH RANDOM SET OF 1 000 LANDMARKS.

Features		Single affinities LBP	Gabor filter	Eigenface	Combined AASC
Class. error, %	Exact	43	48	56	39
	LLL	44 ± 1	49 ± 2	56 ± 1	39 ± 3
Runtime, s	Exact	78	85	105	1 063
	LLL	8.15 ± 0.5	8.76 ± 0.7	8.23 ± 0.8	28.2 ± 2.1
Proj. error, %		4 ± 0.8	3 ± 1	1 ± 0.5	3 ± 2

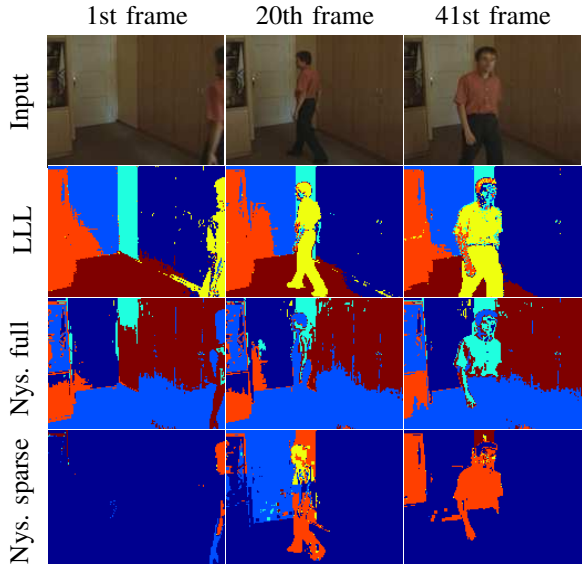

Fig. 8. Spatio-temporal segmentation of 41 frame video with resolution 120×160. We shows 3 frames from the original video. Rows shows the result of clustering with LLL and Nyström methods applied on full or sparse affinities.

It took 3 minutes for LLL to find a solution with $L = 5\,000$ randomly chosen landmarks with 1 minute spent on computing the entries of $\mathbf{Z}$ and 1 minute to compute the reduce affinities (4). Also, as we showed above (fig. 1), k-means with 20 restarts took 1.5 minutes which is comparable to the runtime of LLL. We tried using larger number of landmarks, but the results did not change much.

For Nyström we first tried using the same sparse affinities as we did for LLL. However, with so few non-zeros, it is impossible to cover the entire dataset even with thousands of landmarks. We end up trying 10 000 landmarks before running out of memory. Even for such graph we still got 35 singleton points. We also applied Nyström that approximates full affinities by computing the Gaussian kernel between all the point in the subset. This requires computing eigendecomposition of full $L \times L$ matrix and evaluating an out-of-sample kernel for $N - L$ non-landmark points, which are quite costly. For $L = 3\,000$ these two steps already took more time than the runtime of the LLL and the results were still not as good as LLL.

In fig. 8 we show two frames from the input and their segmentation for LLL and Nyström. The clusters for LLL correspond to the meaningful objects (e.g. floor, person, wall) and are consistent from frame to frame. Interestingly, the pants and the shirt of the person ended up in the same cluster, albeit having very different colors. Nyström results are much worse. Using the full affinities, it is possible to find some useful clusters (e.g. person's torso), but the rest do not represent any good features. Nyström with sparse affinities does not show almost any meaningful segmentation.

VII. DISCUSSION

A further speedup not explored here is the use of multi-processors. The basic operations required are: (1) the computation of the sparse affinity matrix (itself possibly involving constructing a nearest-neighbor graph and computing the affinity values); (2) the computation of the weights $\mathbf{Z}$; (3) the construction of the reduced spectral problem (involving matrix products); (4) the computation of the landmark eigenvectors; (5) the out-of-sample extension of the eigenvectors to the full data; (6) the random k-means restarts on the landmarks; and (7) the final k-means on the entire data. Most of these steps are easily parallelizable, in particular the more expensive ones that involve the entire, $\mathcal{O}(N)$, dataset.

VIII. CONCLUSION

Our contribution is to speed up the eigenproblem and k-means steps of spectral clustering with small approximation

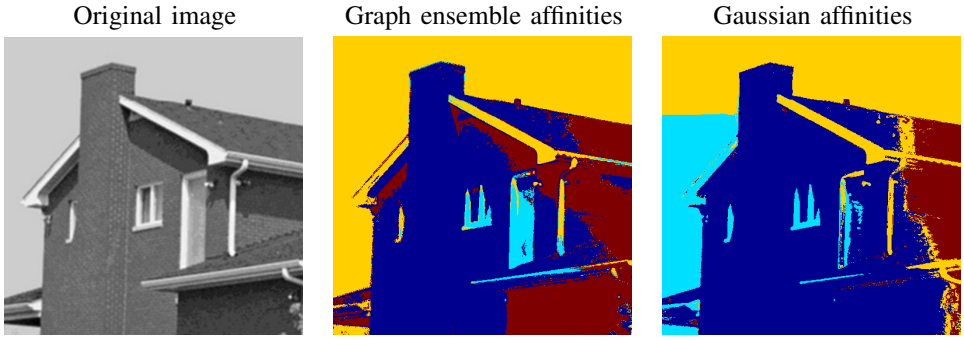

Fig. 10. Image segmentation of 512×512 house image (shown on the left) into four clusters using graph ensemble affinities from [7] (center plot) and Gaussian affinities (right plot). Notice that the former segments the sky, the shadow in a separate cluster.

errors, both with kernel-based affinities and with more complex affinities used by variants of spectral clustering. In the spectral problem, LLL provides two crucial advantages. 1) It uses all the available affinity information in constructing the landmark eigenproblem, but preserving the fast computation properties of subsampling methods. Because fewer landmarks are necessary, this makes it possible to achieve both smaller error and runtime than with the Nyström method. 2) LLL does not rely on an explicitly defined affinity kernel, unlike the the Nyström method, so it applies to variants such as constrained spectral clustering. In the k-means clustering, we run many restarts on the landmarks at little overhead. This gives a good initialization for k-means on the entire data, which then converges in very few iterations, considerably reducing the overall runtime, and is more likely to find a good optimum.

Good results can be obtained for over 700K points in 10 minutes in a single processor. Since most of the required operations are easily parallelizable, in particular the more expensive ones involving the entire data, this could enable the application of spectral clustering to quite larger datasets.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, May 2011.

[2] D. Arthur and S. Vassilvitskii. `k-means++`: The advantages of careful seeding. In *SODA*, p. 1027–1035, Jan. 7–9 2007.

[3] C. T. H. Baker. *The Numerical Treatment of Integral Equations*. Clarendon Press, 1971.

[4] Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, Oct. 2004.

[5] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. Other Titles in Applied Mathematics. SIAM Publ., 2009.

[6] M. Á. Carreira-Perpiñán and Z. Lu. The Laplacian Eigenmaps Latent Variable Model. In *AISTATS*, p. 59–66, 2007.

[7] M. Á. Carreira-Perpiñán and R. S. Zemel. Proximity graphs for clustering and manifold learning. In *NIPS*, p. 225–232, 2005.

[8] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, p. 313–318, 2011.

[9] C. Chennubhotla and A. Jepson. Hierarchical eigensolver for transition matrices in spectral methods. In *NIPS*, p. 273–280, 2005.

[10] T. Cour, F. Bénézit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *CVPR*, p. 1124–1131, 2005.

[11] V. Faber. Clustering and the continuous k-means algorithm. *Los Alamos Science*, 22:138–144, 1994.

[12] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *PAMI*, 26(2):214–225, 2004.

[13] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[14] G. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *NIPS*, p. 857–864, 2003.

[15] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen. Affinity aggregation for spectral clustering. In *Proc. CVPR*, p. 773–780, 2012.

[16] T. H. Kim, K. M. Lee, and S. U. Lee. Learning full pairwise affinities for spectral segmentation. *PAMI*, 35(7):1690–1703, 2013.

[17] D. Kushnir, M. Galun, and A. Brandt. Efficient multilevel eigensolvers with applications to data analysis tasks. *PAMI*, 32(8):1377–1391, 2010.

[18] T. Leung and J. Malik. Contour continuity in region-based image segmentation. In *ECCV*, p. 544–559, 1998.

[19] W. Liu, J. He, and S.-F. Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, 2010.

[20] Z. Lu and M. Á. Carreira-Perpiñán. Constrained spectral clustering through affinity propagation. In *CVPR*, 2008.

[21] M. Maire and S. X. Yu. Progressive multigrid eigensolvers for multiscale spectral segmentation. In *ICCV*, p. 2184–2191, 2013.

[22] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 43(1):7–27, 2001.

[23] B. S. Manjunath and W.-Y. Ma. Texture features for browsing and retrieval of image data. *PAMI*, 18(8):837–842, 1996.

[24] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, p. 849–856. Cambridge, MA, 2002.

[25] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *CVPR*, p. 614–621, 2012.

[26] T. Ojala, M. Pietikäinen, and T. Mäenpää. Gray scale and rotation invariant texture classification with local binary patterns. In *ECCV*, p. 404–420, 2000.

[27] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, p. 1154–1160, 1998.

[28] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[29] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. *PAMI*, 25(12):1615–1618, 2003.

[30] D. Steinley and M. J. Brusco. Initializing k-means batch clustering: A critical evaluation of several techniques. *Journal of Classification*, 24(1):99–121, 2007.

[31] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, 2002.

[32] A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. In *CVPR*, 2008.

[33] M. A. Turk and A. Pentland. Eigenfaces for recognition. *J. Cognitive Neurosci.*, 3(1):71–86, 1991.

[34] M. Vladymyrov and M. Á. Carreira-Perpiñán. Entropic affinities: Properties and efficient numerical computation. In *ICML*, p. 477–485, Atlanta, GA, 2013.

[35] M. Vladymyrov and M. Á. Carreira-Perpiñán. Locally Linear Landmarks for large-scale manifold learning. In *ECML*, p. 256–271, 2013.

[36] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec. 2007.

[37] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, volume 13, p. 682–688, 2001.

[38] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *SIGKDD*, p. 907–916, 2009.

[39] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, p. 1601–1608, 2005.