Convolutional Laplacian Sparse Coding

Xiyang Luo
Department of Mathematics
University of California, Los Angeles
Los Angeles, CA, United States

Brendt Wohlberg
Theoretical Division
Los Alamos National Laboratory
Los Alamos, NM, United States

***Abstract*—We propose to extend the the standard convolutional sparse representation by combining it with a non-local graph Laplacian term. This additional term is chosen to address some of the deficiencies of the ℓ^1 norm in regularizing these representations, and is shown to have an advantage in both dictionary learning and an example image reconstruction problem.**

***Index Terms*—Sparse Representation, Convolutional Sparse Coding, Laplacian Sparse Coding**

I. Introduction

Convolutional sparse coding [1] is a relatively recent variant of sparse coding in which an entire signal or image is decomposed into a sum of convolutions of a set of *coefficient maps*, each of the same size as the input signal or image, with a corresponding dictionary *filter*. One of the most prominent formulations of this problem is Convolutional Basis Pursuit DeNoising (CBPDN)

$$\operatorname*{arg\,min}_{\{\mathbf{x}_m\}} \frac{1}{2}\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\Big\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 \ , \quad (1)$$

where $\{\mathbf{d}_m\}$ is a set of M dictionary filters, $*$ denotes convolution, and $\{\mathbf{x}_m\}$ is a set of coefficient maps. Recent fast algorithms [2], [3] for solving this problem have begun to make it a viable approach for a wider variety of applications.

Convolutional sparse representations have a number of advantages over the standard approach of independently sparse coding of overlapping image patches, including providing a single-valued representation that is optimal over the entire image instead of just locally within each patch. There are, however, also some challenges to using this representation, aside from the computational cost. One of these is the tendency for the set of coefficient maps to be sparse both down the stack of maps at each pixel location, as well as spatially within each map. This latter property is undesirable in some applications, including denoising of Gaussian white noise, where the spatial averaging of independent pixel estimates obtained from the standard patch-based method is beneficial, or in dictionary learning where high spatial sparsity reduces the number of patches in the training images that play a role in forming the dictionary. The work reported here represents an attempt to remedy this weakness by incorporating a non-local regularization that reduces the spatial sparsity in an appropriate way, while retaining the local sparsity of the representation at each pixel location.

The work of both authors was supported by the UC Lab Fees Research grant 12-LR-236660.
The work of Wohlberg was also supported by the U.S. Department of Energy through the LANL/LDRD Program.

II. Convolutional Laplacian Sparse Coding

We propose to augment Eq. (1) with the graph Dirichlet energy $\sum_m \langle \mathbf{x}_m, L\mathbf{x}_m \rangle$, where L is the graph Laplacian [4] of the image non-local graph [5]. Each image patch corresponds to a vertex of the graph, and the weights w_{ij} between vertices represent the similarity between the corresponding image patches, typically computed as

$$w_{ij} = \exp\left(-d_{ij}^2/\tau\right) \ , \quad (2)$$

where d_{ij} is some metric (typical choices are Euclidean or Cosine) between an image patch centered at pixel i and that at pixel j , and τ controls the scaling of the metric. Given the weight matrix $W = (w_{ij})$, the graph Laplacian L is defined as $L = D - W$, where D is the diagonal matrix $D_{ii} = \sum_{i \neq j} w_{ij}$. Our model is motivated by the non-local smoothing properties of the Dirichlet energy, which are apparent from the equation

$$\langle u, Lu \rangle = \sum_{\alpha,\beta \in V} w_{\alpha\beta}(u_\alpha - u_\beta)^2 \ . \quad (3)$$

Here α, β range through all vertices on the graph, and u is any real-valued function defined on the graph. Since $w_{\alpha\beta}$ is smaller if the vertices α, β are more similar, the Dirichlet energy will be small if similar vertices have similar u values.

In our context, the vertices α are image patches indexed by their spatial location (i, j), and the u corresponds to the sparse coefficients $\mathbf{x}_m$. Thus by the analysis above, the regularizer is an explicit penalty to force similar image patches to have similar sparse representations. In practice, we actually use the normalized Laplacian $L_s = I - D^{-1/2}WD^{-1/2}$ [4], since it handles outliers better and is the more common choice for non-local image graphs. However, the motivation remains the same and we will not make a distinction from here on.

Formally, our model can be written as

$$\operatorname*{arg\,min}_{\{\mathbf{x}_m\}} \frac{1}{2}\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\Big\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 + \frac{\mu}{2}\sum_m \langle \mathbf{x}_m, L\mathbf{x}_m \rangle \ . \quad (4)$$

This model can be considered as a convolutional variant of the previously-proposed Laplacian Sparse Coding method [6], which has been applied to image classification tasks [6], [7] as well as image restoration tasks [8], [9]. The key difference between the proposed approach and the patch based Laplacian Sparse Coding in [6] is that the sparse code is learned over *every* single patch and *jointly* over the entire image, due to the nature of the convolutional model. Thus unlike [6], [7], there is no need to use the SIFT local descriptor to pre-define a set of patches to learn on, and there is no need for patch averaging to resolve the multi-valued estimation as in [8].

III. Algorithm

Our two alternative algorithms for solving Eq. (4) are both based on the Alternating Direction Method of Multipliers (ADMM) [10] framework. Their differences correspond to whether we perform an additional splitting in $\langle \mathbf{x}_m, L\mathbf{x}_m \rangle$, or include it in the ℓ^1 subproblem.

A. ADMM Double-Split

In this approach, we perform an additional splitting to give

$$\underset{\{\mathbf{x}_m\}}{\arg\min} \frac{1}{2}\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\Big\|_2^2 + \lambda \sum_m \|\mathbf{y}_m\|_1 + \frac{\mu}{2}\sum_m \langle \mathbf{z}_m, L\mathbf{z}_m \rangle \;\; \text{s.t.} \;\; \mathbf{x}_m = \mathbf{y}_m, \;\; \mathbf{x}_m = \mathbf{z}_m \; . \quad (5)$$

The corresponding ADMM primal updates are

$$\{\mathbf{x}_m\}^{(j+1)} = \underset{\{\mathbf{x}_m\}}{\arg\min} \frac{1}{2}\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\Big\|_2^2 + \rho \sum_m \Big\|\mathbf{x}_m - \frac{1}{2}\left(\mathbf{u}_m^{(j)} + \mathbf{y}_m^{(j)} + \mathbf{v}_m^{(j)} + \mathbf{z}_m^{(j)}\right)\Big\|_2^2 \quad (6)$$

$$\{\mathbf{y}_m\}^{(j+1)} = \underset{\{\mathbf{y}_m\}}{\arg\min} \lambda \sum_m \|\mathbf{y}_m\|_1 + \frac{\rho}{2}\sum_m \Big\|\mathbf{x}_m^{(j+1)} - (\mathbf{y}_m + \mathbf{u}_m^{(j)})\Big\|_2^2 \quad (7)$$

$$\{\mathbf{z}_m\}^{(j+1)} = \underset{\{\mathbf{z}_m\}}{\arg\min} \frac{\mu}{2}\sum_m \langle \mathbf{z}_m, L\mathbf{z}_m \rangle + \frac{\rho}{2}\sum_m \Big\|\mathbf{z}_m - (\mathbf{x}_m^{(j+1)} + \mathbf{v}_m^{(j)})\Big\|_2^2 \; . \quad (8)$$

The $\mathbf{x}_m$ and $\mathbf{y}_m$ updates are the same as in the standard convolutional learning case, and can be efficiently solved in the Fourier domain and by soft thresholding respectively, as in [2], [3]. The $\mathbf{z}_m$ update involves solving a linear system.

It is worth emphasizing that, despite the double splitting, this algorithm can be expressed in the standard ADMM form if the two split variables are appropriately combined in block form by defining the matrix $\mathbf{A}$ mapping $\mathbf{x} \mapsto (\mathbf{x}, \mathbf{x})^T$ and $\mathbf{u} = (\mathbf{y}, \mathbf{z})^T$, and imposing the constraint $\mathbf{u}_m = \mathbf{A}\mathbf{x}_m$.

B. ADMM Single-Split

Instead of performing an additional splitting, we can also group the Laplacian term together with the ℓ^1 term and solve an $\ell^2 + \ell^1$ minimization as a sub-problem. The resulting iterations are

$$\{\mathbf{x}_m\}^{(j+1)} = \underset{\{\mathbf{x}_m\}}{\arg\min} \frac{1}{2}\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\Big\|_2^2 + \frac{\rho}{2}\sum_m \Big\|\mathbf{x}_m - \mathbf{y}_m^{(j)} - \mathbf{u}_m^{(j)}\Big\|_2^2 \quad (9)$$

$$\{\mathbf{y}_m\}^{(j+1)} = \underset{\{\mathbf{y}_m\}}{\arg\min} \lambda \sum_m \|\mathbf{y}_m\|_1 + \frac{\mu}{2}\sum_m \langle \mathbf{y}_m, L\mathbf{y}_m \rangle \; \frac{\rho}{2}\sum_m \Big\|\mathbf{x}_m^{(j+1)} - \mathbf{y}_m - \mathbf{u}_m^{(j)}\Big\|_2^2 \; . \quad (10)$$

An efficient implementation of the algorithm is obtained by warm-starting the $\mathbf{y}_m$ sub-problem from the previous iterate. Moreover, each sub-problem can be solved inexactly with an adaptive tolerance ϵ_n compatible with the primal and dual residuals of the main ADMM iteration (we choose $\epsilon_k = \max\{r_k, s_k\}/10$). Finally, the $\mathbf{y}_m$ problem itself can be solved via a standard algorithm such as ADMM or FISTA [11].

C. Eigenspace Decomposition

A common trick when dealing with the graph Laplacian is to decompose L in the spectral domain, diagonalizing L using its eigenbasis $\{\mathbf{e}_k\}_{k \in \{1,\ldots,n\}}$. Using this formulation, iterates involving L can be computed explicitly by computing inner products with eigenvectors. For example, the $\mathbf{z}$ update of Eq. (8) becomes

$$\mathbf{z}_m = \sum_k \frac{\rho}{\rho + \mu\lambda_k} \langle \mathbf{x}_m + \mathbf{v}_m, \mathbf{e}_k \rangle \mathbf{e}_k \; , \quad (11)$$

where λ_k is the k-th eigenvalue of L corresponding to e_k. This approach would still be infeasible if we were to compute all the eigenvectors of L, but for many non-local graphs derived from images, most of the larger eigenvalues are indeed close to unity if the graph Laplacian is normalized. Thus only the smallest few eigenvectors are needed to approximate the matrix L. This technique, called spectral truncation, has been successfully applied in graph cut algorithms for clustering [12], [13].

D. Speed of Algorithms

Here we compare the computational performance of various algorithm options. We have a choice of using eigenvectors or using the full matrix, and also using ADMM double-split or ADMM single-split for the main algorithm, giving a total of four combinations. We test each one on a set of problems of varying sizes, and plot the total convergence time relative to that of standard convolutional sparse coding (e.g. 2.0 means it takes twice as long to converge as the standard algorithm). The relative residual stopping tolerance [10] is set to 10^{-3}. All algorithms are tested on the same image with the same parameters $\lambda = 0.1, \mu = 0.1$ except for the standard convolutional case, which is tested with $\lambda = 0.1$. As Figures 1 and 2 show, ADMM double-split is faster when using eigenvector truncation, and single-split is faster when using the full matrix. This discrepancy is due to different implementations of the $\{\mathbf{z}_m\}$ update in ADMM double-split. In the full matrix case, $\{\mathbf{z}_m\}$ is updated by solving a symmetric linear system which can be costly, while in the eigenvector case the update only involves inner products with the eigenvectors.

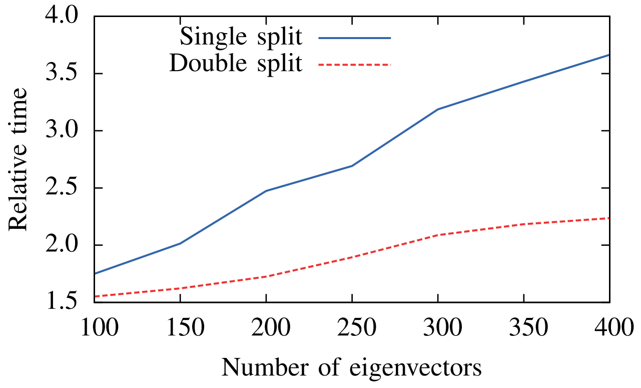

Fig. 1. Eigenvector Time Test

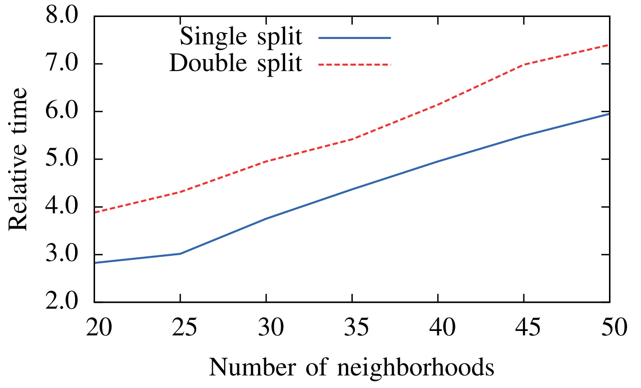

Fig. 2. Full Matrix Time Test

E. Efficient Graph Computation

In general, it is too computationally expensive to generate the full non-local graph of the image. One way to deal with this is to use eigenvector decomposition as described in Sec. III-C. Since only the first few eigenvectors are needed, it makes sense to use an algorithm that computes the eigenvectors without constructing the full graph. We use the Nystrom Extension [14], a sampling strategy used to compute an approximation to the true eigenvectors. Error bounds for the Nystrom Extension have been studied in [15].

There are cases where too many eigenvectors are needed to accurately reflect the full graph Laplacian. In this case, we have to resort to using the full matrix L. A straightforward way to reduce cost is to sparsify the graph. Graph sparsification can be done via building a k-nearest neighbor graph or spatial localization, i.e., to restrict connections of pixels to only its spatial neighborhood. An interesting observation is that the case where too many eigenvectors are needed often occurs when the graph construction parameters have made the graph too disconnected, i.e., sparse. This suggests a guideline for choosing the best algorithm: if we intend the graph to be well connected, use eigenvector decomposition; otherwise, use a sparse Laplacian.

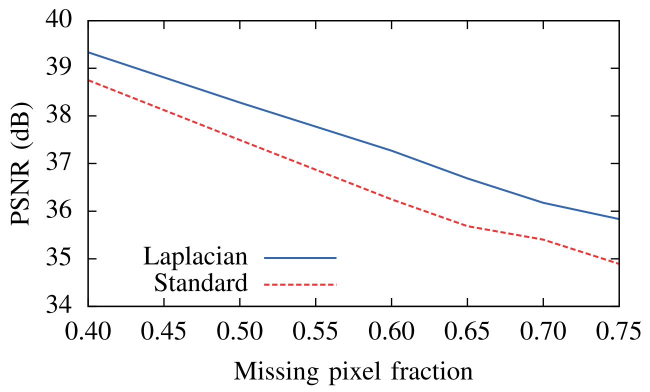

Fig. 3. Lena Inpainting Comparison

IV. Results

A. Image Inpainting

Laplacian convolutional sparse coding can improve the performance of image inpainting compared to standard convolutional sparse coding. The model for inpainting using the standard convolutional sparse coding is

$$\underset{\{\mathbf{x}_m,\mathbf{z}_1,\mathbf{z}_2\}}{\arg\min}\ \frac{1}{2}\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m + \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{s}\Big\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 + \sum_i \chi(i)\mathbf{z}_1(i) + \frac{\nu}{2}\|\nabla \mathbf{z}_2\|_2^2 \ , \tag{12}$$

where $\chi(i) = 0$ if i is a missing pixel, and $+\infty$ otherwise. Here $\mathbf{s}$ will be the corrupted image, $\mathbf{z}_1$ will absorb the missing pixel values, $\mathbf{z}_2$ will be a low frequency component of the image[1], and the reconstruction will be $\mathbf{s}_{\text{rec}} = \sum_m \mathbf{d}_m * \mathbf{x}_m + \mathbf{z}_2$. The corresponding model for the Laplacian case is

$$\underset{\{\mathbf{x}_m,\mathbf{z}_1,\mathbf{z}_2\}}{\arg\min}\ \frac{1}{2}\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m + \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{s}\Big\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 + \sum_i \chi(i)\mathbf{z}_1(i) + \frac{\mu}{2}\sum_m \langle \mathbf{x}_m, L\mathbf{x}_m\rangle + \frac{\nu}{2}\|\nabla \mathbf{z}_2\|_2^2 \,. \tag{13}$$

We use the standard model Eq. (12) to inpaint the image first to construct the graph Laplacian L.

Inpainting is tested on the 512×512 "Lena" image with missing pixel fraction ranging from 40% (PSNR 10.67dB) to 75% (PSNR 8.65dB), using a separately trained $12 \times 12 \times 36$ dictionary. A parameter search on λ and ν is done first to produce the best performance for the standard model. The same set of parameters is then used for the Laplacian model with μ set to 0.1, which has proved empirically to be a good choice, and with a K-Nearest neighbor graph with $K = 40$, constructed using the cosine distance metric. A comparison of PSNR values for both cases is given in Fig. 3. The Laplacian model is consistently better than the standard convolutional model for all noise levels, with an average PSNR increase of around 0.85 dB.

[1]Employed here for similar reasons to the usual subtraction of the patch mean in patch based sparse coding.

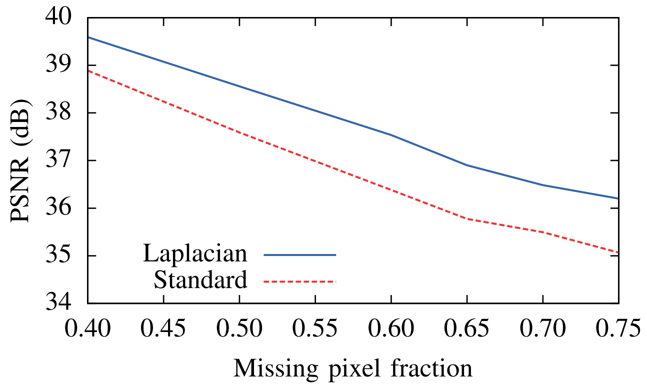

Fig. 4. Straw Inpainting Comparison

As might be expected, the Laplacian model yields better performance for images with more structural similarity. If we repeat the same experiment on "Straw", a texture-rich image consisted of vertically aligned straws [16], the average PSNR increase is around 1 dB, as shown in Fig. 4. More importantly, the performance gap is wider for the "Straw" image when the corruption level is higher, showing that the model has better performance on images with more structural similarity.

B. Dictionary Learning

Dictionary learning with the graph Laplacian regularizer can be achieved by adding a constraint $\|\mathbf{d}_m\| \leq 1$ to Eq. (4) and updating $\mathbf{d}$ and $\mathbf{x}$ in a interleaved manner, as in [3]. In some applications it is desirable to train dictionaries from images corrupted by Gaussian white noise. Convolutional dictionary learning has relatively poor resistance to noise in the training images due to the homogeneous treatment of sparsity in spatial and filter indices that is inherent in the ℓ^1 regularizer. This is substantially improved by incorporating the graph Laplacian regularization proposed here. This improvement is due to the nonzero coefficients of the Laplacian model having more spatial structure when given the same amount of sparsity as a result of the non-local smoothing effect of the graph Laplacian. A comparison using dictionaries trained on 5 randomly selected images from the MIRFlickr dataset [17] is presented in Fig. 5 and 6; note that the dictionary filters in Fig. 6 have substantially less noise in smooth regions.

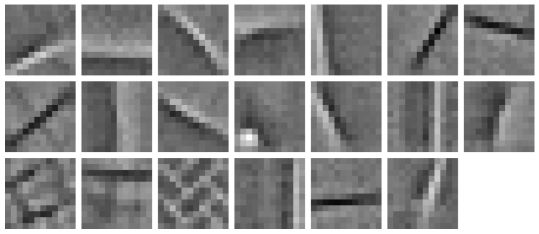

Fig. 5. Best Standard Dictionary for $N = 20$

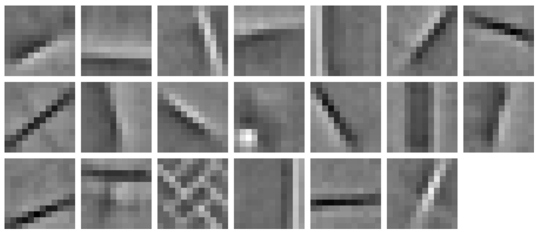

Fig. 6. Best Laplacian Dictionary for $N = 20$

V. Conclusions

We have proposed a modified form of Convolutional BPDN that includes an additional regularization term based on the Laplacian of the image non-local graph. Two algorithm variants have been developed for solving the resulting optimization problem, and initial experiments indicate that the modified form provides some advantages in both dictionary learning and image restoration applications.

References

[1] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, June 2010, pp. 2528–2535.

[2] B. Wohlberg, "Efficient convolutional sparse coding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2014, pp. 7173–7177.

[3] ——, "Efficient algorithms for convolutional sparse representations," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 301–315, Jan. 2016.

[4] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[5] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, 2009, pp. 2272–2279.

[6] S. Gao, I. W.-H. Tsang, L.-T. Chia, and P. Zhao, "Local features are not lonely – Laplacian sparse coding for image classification," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, 2010, pp. 3555–3561.

[7] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 92–104, 2013.

[8] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1327–1336, 2011.

[9] W. Dong, X. Li, L. Zhang, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, 2011, pp. 457–464.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[11] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[12] A. L. Bertozzi and A. Flenner, "Diffuse interface models on graphs for classification of high dimensional data," *Multiscale Modeling & Simulation*, vol. 10, no. 3, pp. 1090–1118, 2012.

[13] E. Merkurjev, E. Bae, A. L. Bertozzi, and X.-C. Tai, "Global binary optimization on graphs for classification of high-dimensional data," *J. Math. Imaging Vis.*, vol. 52, no. 3, pp. 414–435, 2015.

[14] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nystrom method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, 2004.

[15] A. Gittens, "The spectral norm error of the naive Nystrom extension," *arXiv preprint arXiv:1110.5305*, 2011.

[16] "Image of Vertical Straw Texture," http://texturee.deviantart.com/art/Straw-Texture-260793536 (Nov. 2015).

[17] M. J. Huiskes and M. S. Lew, "The MIR Flickr retrieval evaluation," in *Proc. ACM Intl. Conf. Multimedia Info. Retrieval*, 2008, pp. 39–43.