Spectral Graph Theory and High Performance Computing
Master Thesis

David Wobrock david.wobrock@gmail.com

October 10, 2017

Contents

1 Image Processing using the Graph Laplacian Operator

1.1 Introduction

The talk by Peyman Milanfar [1], working at Google Research, about using the Graph Laplacian Operator for Image Processing purposes awakes curiosity.

1.2 Theory basis

Multiple image processing filters can be built by Graph Laplacians. As Milanfar mentions in [1], smoothing, deblurring, sharpening, dehazing, and other filters can be created. Laplacians can also be used for compression artifact removal and low-light imaging.

As it is known, an image filter consists of a function which outputs one pixel, and taking all pixels as input and applying weights to them. We can write this as

$$z_i = \sum_j W_{ij} y_j,$$

z_i being the output pixel, W_{ij} the weight and y_j all input pixels. This means that we have a vector of weights for each pixel.

So, as a practical notation, we can say that, with W the matrix of weights and y the input image as a vector,

$$z = Wy.$$

Now we want to represent the image as a graph. Each pixel is a node and has edges to multiple other nodes. We can define how the pixels connect to each other, and we can ultimately say that the graph is complete, each node connects to all other nodes. But we can weigh the edges to measure the similarity between pixels.

Affinity, or similarity, is a subjective term which we can also define as we need. Pixel can be similar if they are spatially close, or if they have the same color, or both for example. These similarities give us the affinity matrix[1] called K.

By extending this affinity matrix, we obtain the Graph Laplacian $\mathcal{L}$. And we want to build this filter W from the Laplacian.

[1] Also called kernel matrix or similarity matrix

According to [2], to build a filter from the Laplacian, we have

$$W = I - \mathcal{L},$$

and so reciprocally

$$\mathcal{L} = I - W.$$

The Graph Laplacian has multiple definitions, but the simplest is the unnormalised one:

$$\mathcal{L}_U = D - K,$$

with D a positive definite diagonal matrix with the normalising factors as $D_{jj} = diag\{\sum_i K_{ij}\}$ along its diagonal. Other definitions of the Laplacian are shown later in this document.

Interestingly, we can define from [3] that, by approximating W by a symmetric, positive definite, double stochastic matrix, this symmetric W can be computed thanks to this eigen-decomposition

$$W = VSV^T,$$

V being the eigenvectors and S the eigenvalues as a diagonal matrix.

Our global filter can be expressed as

$$z = Wy = VSV^T y.$$

This will be useful since the filter matrix W becomes huge very quickly and we will need a way to approximate it. Indeed, W is a square matrix with $n * n$ elements, n the number of pixels in the picture.

Approximation by Nyström Extension To compute the filter, the first matrix that is needed is the affinity matrix K. Both matrices have the same size and are computationally expensive.

That is why we approximate the affinity matrix by sampling the picture. We sample p pixels, such as $p << n$.

Different sampling techniques exist and are shown later in this document.

Once sampled, we compute K_A and K_B, which are parts of K such as

$$K = \begin{bmatrix} K_A & K_B \\ K_B^T & K_C \end{bmatrix}.$$

K_A represents the affinity matrix of size $p * p$ between the sample pixels, and K_B the affinity matrix of size $p * m$, such as $m = n - p$, between the sample pixels and the remaining pixels.

These matrices will be computed thanks to a kernel function (or affinity function). This can be the bilateral filter, non-local means or another.

Once computed, we can approximate the eigenvectors Φ and eigenvalues Π of K thanks to the eigen-decomposition of K_A.

As in [3], we can define the decomposition of K_A

$$K_A = \Phi_A \Pi_A \Phi_A^T$$

and the approximation of K such as

$$\tilde{K} = \tilde{\Phi} \Pi_A \tilde{\Phi}^T$$

and finally the approximation of the p first eigenvectors of K

$$\tilde{\Phi} = \begin{bmatrix} \Phi_A \\ K_B^T \Phi_A \Pi_A^{-1} \end{bmatrix}$$

From this eigen-decomposition approximation of K, we can then approximate the eigenvectors and eigenvalues of W.

One might wonder how an approximation of the first elements of the eigen-decomposition can be enough to approximate the whole matrix. Indeed, the eigenvalues decay quickly as shown in [1] and [4].

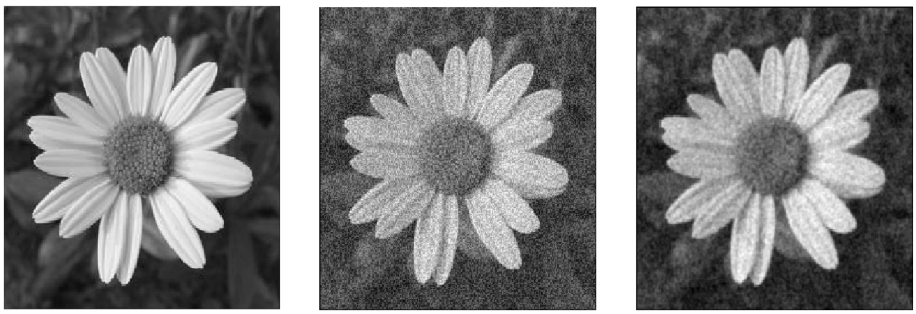
As our experiment below shows, the eigenvalues decay very quickly which means that the whole matrix is mainly defined by the first elements.

1.3 First implementation

Details The first implementation of this flow is aimed to denoise the input image. The sample pixels are selected in a spatially uniform manner. It uses the NLM from [5] as kernel method to compute the pixels similarity. The Laplacian matrix is defined through the Sinkhorn iterative algorithm [6], where the resulting approximated eigenvectors require to be orthogonalised, which can be done in one step, as discussed in [7]. We finally apply our approximated filter to the noisy image.

The implementation is inspired by the MatLab implementation of [3].

Results The results of this very first experiment are not yet satisfying as we can observe visually:



Original image | Noisy image (σ=100) | Output image

We observe that the output picture is still quite noisy and blurry, we can certainly do better.

1.4 Algorithm variations

1.4.1 Sampling method

The sample required less than 1% of the pixels of the image. To achieve this, we can use different approaches. The chosen method is decisive for the Nyström method.

Random sampling (RS) most common and simple sampling scheme, but no deterministic guarantee of the output quality. Can produce good results for images with poor resolution, but with a huge amount of data, random sampling is limited because it cannot reflect the structure of the data set [8].

K-means sampling (KS) associate to each pixel a 5-D space (R, G, B, X, Y) and divide the pixels into K clusters (K centers). These clusters are a good sampling scheme for images with simple and uniform backgrounds [9] [10].

Uniform spatially sampling the uniformity of the sample gives good results for image sampling because of the spatial correlation of pixels. This method remains simple but effective [3].

Incremental sampling (INS) is an adaptive sampling scheme, meaning that it select points according to the similarity, so that we can have an approximate optimal rank-k subspace of the original image [8].

Mean-shift segmentation-based sampling this scheme performs good for complex backgrounds. The method consists in over-segmenting the image into n regions and only one pixel of each region will be sampled using the spatially closest pixel to the center of the region given a formula in [9].

1.4.2 Affinity function

The kernel function K_{ij} measures the similarity between the pixel y_i and y_j.

Listing

Spatial Gaussian Kernel takes into account only the spatial distance between two pixels [1].

Photometric Gaussian Kernel considers the intensity and color similarity of the pixels [1].

Bilateral Kernel one of the most used kernel which smooths images by a nonlinear combination of the spatial and photometric gaussian kernels [1] [3].

Non-Local Means (NLM) is similar to the bilateral kernel, a data-dependent filter, except that the photometric affinity is captured patch-wise [3].

Examples To illustrate the impact of the affinity function, here are some examples of the affinity matrix for certain pixels. The more a pixel is colored in red, the more similar it is to the selected pixel, with respect to the chosen function. A blue colored pixel is dissimilar to the considered pixel.

To generate these examples, we use an image of a house of dimension 200x300 pixels. Generating the affinity matrix takes approximately 60 seconds on an Intel i5 processor and the generation takes less than a 1 GB of memory. We use a spatially uniform sampling technique and select 1% of the pixels. We show two affinity matrices for each techniques, the first one is on the grass in front of the house and the second on the roof.

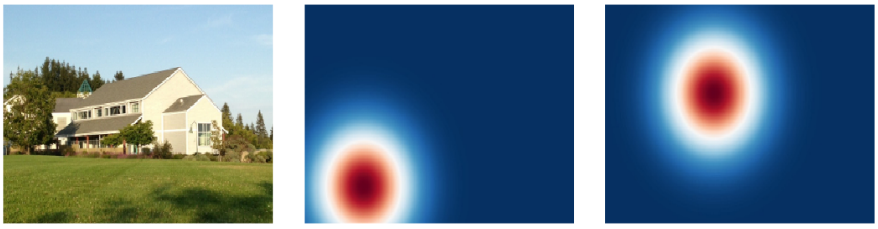
Spatial Gaussian Kernel The formula of this kernel is

$$K(x_i, x_j) = exp(-||x_i - x_j||^2/h_x^2).$$

Affinity matrices with $h_x = 5$:

Original image | Affinity matrix of pixel 165x65 using spatial function | Affinity matrix of pixel 80x120 using spatial function

Affinity matrices with $h_x = 50$:



Original image | Affinity matrix of pixel 165x65 using spatial function | Affinity matrix of pixel 80x120 using spatial function

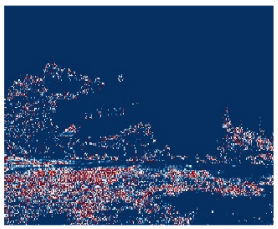
As we can see, the parameter is influencing on the normalisation of the values and gaussian standard deviation. The bigger is it, the more tolerant the spatial distance computation will be.

Spatial Gaussian Kernel The formula of this kernel is

$$K(z_i, z_j) = exp(-||z_i - z_j||^2/h_z^2).$$

Affinity matrices with $h_z = 5$:

Original image

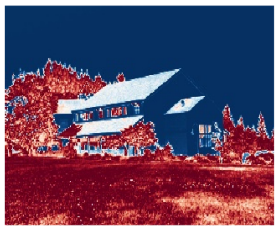


Affinity matrix of pixel 165x65
using photometric function

Affinity matrix of pixel 80x120
using photometric function

Affinity matrices with $h_z = 50$:

Original image



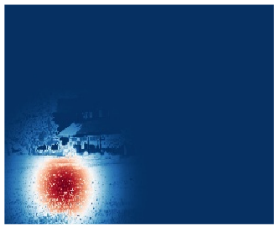
Affinity matrix of pixel 165x65
using photometric function

Affinity matrix of pixel 80x120
using photometric function

Generally, the h parameters in both kernel functions here are smoothing parameters. If h is small, it is more discriminating between the affinity of different pixels.

Bilateral Kernel Affinity matrices with $h_x = 40$ and $h_z = 30$:

Original image



Affinity matrix of pixel 165x65
using bilateral function

Affinity matrix of pixel 80x120
using bilateral function

1.4.3 Graph Laplacian

Graph Laplacian has multiple possible definitions and each has its own properties. A good summary can be found in [1]. A Graph Laplacian can be symmetric which is important for eigen-decomposition of the matrix. It can have a DC eigenvector, which means that the Laplacian has to give 0 if we apply it to a constant image. This is also useful to have. And the spectral range, corresponding to the range of the eigenvalues, is important because we will use the filters derived from the Laplacian multiple times, and if the eigenvalues are not between 0 and 1, then the filters tend to be unstable. With K being the affinity matrix, $d_i = \sum_j K_{ij}$ and $D = diagd_i{}_1^n$:

Generally, it is a good practice to stick to one definition of the Laplacian.

Laplacian Name	Formula	Symmetric	DC eigenvector	Spectral Range
Un-normalised	$D - K$	Yes	Yes	[0, n]
Normalised	$I - D^{-1/2}KD^{-1/2}$	Yes	No	[0, 2]
Random Walk	$I - D^{-1}K$	No	Yes	[0, 1]
"Sinkhorn" [6]	$I - C^{-1/2}KC^{-1/2}$	Yes	Yes	[0, 1]
Re-normalised	$\alpha(D - K)$, $\alpha = \mathcal{O}(n^{-1})$	Yes	Yes	[0, 1]

References

[1] Peyman Milanfar. *Non-conformist Image Processing with Graph Laplacian Operator*. 2016. URL: https://www.pathlms.com/siam/courses/2426/sections/3234.

[2] Peyman Milanfar. "A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical". In: *IEEE Signal Processing Magazine* 30.1 (Jan. 2013), pp. 106–128. ISSN: 1053-5888. DOI: 10.1109/MSP.2011.2179329. URL: http://ieeexplore.ieee.org/document/6375938/ (visited on 10/03/2017).

[3] Hossein Talebi and Peyman Milanfar. "Global Image Denoising". In: *IEEE Transactions on Image Processing* 23.2 (Feb. 2014), pp. 755–768. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2013.2293425. URL: http://ieeexplore.ieee.org/document/6678291/ (visited on 10/03/2017).

[4] Franois G. Meyer and Xilin Shen. "Perturbation of the eigenvectors of the graph Laplacian: Application to image denoising". In: *Applied and Computational Harmonic Analysis* 36.2 (Mar. 2014), pp. 326–334. ISSN: 1063-5203. DOI: 10.1016/j.acha.2013.06.004. URL: http://www.sciencedirect.com/science/article/pii/S1063520313000626 (visited on 10/05/2017).

[5] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. "A Review of Image Denoising Algorithms, with a New One". In: *SIAM Journal on Multiscale Modeling and Simulation* 4 (Jan. 2005). DOI: 10.1137/040616024.

[6] Peyman Milanfar. "Symmetrizing Smoothing Filters". en. In: *SIAM Journal on Imaging Sciences* 6.1 (Jan. 2013), pp. 263–284. ISSN: 1936-4954. DOI: 10.1137/120875843. URL: http://epubs.siam.org/doi/10.1137/120875843 (visited on 10/04/2017).

[7] Charless Fowlkes et al. "Spectral grouping using the Nystrom method". In: *IEEE transactions on pattern analysis and machine intelligence* 26.2 (2004), pp. 214–225. URL: http://ieeexplore.ieee.org/abstract/document/1262185/ (visited on 10/03/2017).

[8] Qiang Zhan and Yu Mao. "Improved spectral clustering based on Nystrm method". en. In: *Multimedia Tools and Applications* 76.19 (Oct. 2017), pp. 20149–20165. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-017-4566-4. URL: http://link.springer.com/10.1007/s11042-017-4566-4 (visited on 10/03/2017).

[9] Chieh-Chi Kao et al. "Sampling Technique Analysis of Nystrm Approximation in Pixel-Wise Affinity Matrix". In: July 2012, pp. 1009–1014. ISBN: 978-1-4673-1659-0. DOI: 10.1109/ICME.2012.51.

[10] Kai Zhang, Ivor W. Tsang, and James T. Kwok. "Improved Nystrm low-rank approximation and error analysis". In: *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1232–1239. URL: `http://dl.acm.org/citation.cfm?id=1390311` (visited on 10/04/2017).