Spectral Graph Theory and High Performance Computing
Master Thesis

David Wobrock david.wobrock@gmail.com

October 19, 2017

Contents

Chapter 1

Introduction

1.1 Background

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

Indeed, Milanfar reports that these techniques to build filters are used on smartphones, which means it is done in a reasonable time with limited computational resources. Over 2 billion photos are shared daily on social media [1], with very high resolutions and filters are often applied on them.

1.2 Objective

The objective of this thesis is to, in a first place, understand and implement some image processing algorithms shown by Milanfar's work on 2D images. We want to observe the performance and results of these algorithms to be validate the usage of intrinsic spectral properties of an image. For instance, the spectrum of the graph associated to an image can serve for image segmentation. Applied in a fine-grained manner, this segmentation could lead to image compression. More classic algorithm can also be achieved with the described approach, such as image denoising, deblurring, sharpening, etc.

The next step includes extending these techniques on 3D images. Given the results, new scalable processing approaches for 3D images could be explored. These also include standard processing filters such as denoising and sharpening, but also segmentation and compression techniques.

These approaches, even though faster than other filters, still require computations which can grow fast in the 3D case. That is why we want to apply the algorithm on high performance architectures in order to achieve reasonable computation times.

1.3 Related work

1.3.1 Denoising

Background Even with high quality cameras, denoising and improving a taken picture remains important. The two main issues that have to be addressed by denoising are blur and noise. The effect of blur is internal to cameras since the number of samples of the continuous signal is limited and it has to hold the Shannon-Nyquist theorem [2]. Noise comes from the light aquisition system that fluctuates in relation to the amount of incoming photons.

To model these problems, we can formulate the deficient image as,

$$y = z + e,$$

where e is the noise vector of variance σ^2, z the clean signal vector and y the noisy picture.

What we want is a high-performance denoiser, capable of scaling up in relation to increasing the image size and keeping reasonable performances. The output image should come as close as possible to the clean image. As an important matter, it is now accepted that images contain a lot of redundancy. This means that, in a natural image, every small enough window has many similar windows in the same image.

Traditional, patch-based methods The image denoising algorithms review proposed by [2] suggests that the NL-means algorithm, compared to other reviewed methods, comes closest to the original image when applied to a noisy image. This algorithm takes advantage of the redundancy of natural images and for a given pixel i, predicts its value by using the pixels in its neighbourhood.

In [3], the authors propose the BM3D algorithm, a denoising strategy based on grouping similar 2D fragments of the image into 3D data arrays. Then, collaborative filtering is performed on these groups and return 2D estimates of all grouped blocks. This algorithm exposed state-of-the-art performance in terms of denoising at that time. The results are still one of the best for a reasonable computational cost.

Global filter In the last couple of years, global image denoising filters came up, based on spectral decompositions [4]. This approach considers the image as a complete graph, where the filter value of each pixel is approximated by all pixels in the image. We define the approximated clean image $\hat{z}$ by,

$$\hat{z} = Wy,$$

where W is our data-dependent global filter, a $n \times n$ matrix, n the number of pixels in the picture. W is computed from the graph affinity matrix[1] K, also of size $n \times n$, such as

$$K = \mathcal{K}_{ij},$$

[1] Also called kernel matrix or similarity matrix

where $\mathcal{K}$ is a similarity function between two pixels i and j. As the size of K can grow very large, we sample the image and compute an approximated $\hat{K}$ on this subset using the Nyström extension. K can in fact be approximated through its eigenvalues and eigenvectors. Knowing that the eigenvalues of K decay very fast [1], the first eigenvectors of the subset of pixels are enough to compute the approximated $\hat{K}$.

Generally, as proposed in [4] and [5], to improve the denoising performance of global filters, pre-filtering techniques are used. It is proposed to first apply a NL-means algorithm to the image to reduce the noise, but to still compute the global filter on the noisy input and apply the filter to the pre-filtered image.

Comparison between patch-based and global methods As [5] suggests, global filter methods have the possibility to converge to a perfect reconstruction of the clean image, which seems to be impossible for techniques like BM3D. Global filtering also seems promising for creating more practical image processing algorithms.

The thesis [6] proposes a normalised iterative denoising algorithm which is patch-based. The work reports that this technique has slightly better results than the global filter but essentially has a better runtime performance.

1.3.2 Sharpening

Background Sharpening consists basically in a high-pass filter which will magnify high frequency details [6]. The two main issues again with this approach are that noise often displays high frequency attributes, which will be amplified by the filter. Secondly, the phenomenon of overshooting and undershooting, which appears when sharpening already sharp parts (edges for example), will exhibit unpleasant artifacts.

Modeling the problem is the same as for denoising, $\hat{z} = Fy$, F being our filter.

Classical Difference of Gaussian (DoG) operator This technique consists of computing the difference between two gaussian kernels, which will produce a range of different kernels with various frequencies. Indeed, DoG involves subtracting a blurred version of the input image to another, less blurred version of this original image [7]. A blurred image can be obtained by convolving the input image with a gaussian kernel. This gaussian blurring actually removes high-frequency spatial information. So the idea is that by substracting this blurred image from a less blurred one, we will keep the high-frequency information, like a high-pass filter[2], which results in image sharpening.

[2]More precisely, like a band-pass filter here

Structure-aware sharpening We know from its definition [8] that the smoothing filter W is a symmetric and doubly stochastic matrix[3]. So the largest eigenvalue $\lambda_1 = 1$ shows that it has low pass filter characteristics. With the definition of the Laplacian, in relation to the matrix W such as $\mathcal{L} = I - W$, we can consider that it is a data-adaptive high-pass filter [6]. So we can define a data-adaptive unsharp mask filter as

$$F_1 = I + \beta(I - W),$$

with $\beta > 0$. This is basically a weighted high-pass filtered version of the input image [1].

But with this approach, the noise amplifying and overshoot problems remain. The proposed solution in [8] applies first a smoothing filter to the image, then the unsharp mask filter and finally the same smoothing filter again. The smoothing and sharping filters use possibly two different kernels matrices as basis. The first smoothing filter aims to reduce the noise, while we still avoid over-smoothing. The second and final smoothing controls the effect of the amplified noise and the overshoot artifacts. This gives us the filter

$$F = W_1(I + \beta(I - W_2))W_1,$$

where W_1 and W_2 are constructed from the affinity matrices K_1 and K_2, which can differ in relation to the parameters of their respective kernel function.

Comparison As [8] suggests, the structure-aware sharpening is data-adaptive and more noise robust than a classical DoG filter. Even though, both approaches use a similar concept. The proposed filter is actually a data-derived DoG-based filter. We can rewrite the proposed filter as

$$F = (1 + \beta)W_1^2 - \beta W_1 W_2 W_1,$$

where we can actually see the difference between two versions of the input image.

1.3.3 Image Segmentation

Background To recognise patterns, find groups and cut images is a similar problem to graph partitioning. As is it known, most formulations of partitioning a graph result in a NP-hard problem. So it will be our goal to approximate quickly good partitions instead of looking for an optimal solution.

A graph $G = (V, E)$ is partitioned into two disjoint sets A and B such as $A \cup B = V$ and $A \cap B = \emptyset$ by removing edges between A and B. One can compute the value of such a *cut* with the degree of dissimilarity between A and B, which is calculated from the total weight of the removed edges:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v).$$

[3]Depending on the used Laplacian definition. For example the one obtained by "Sinkhorn" iterations [9]

Minimum cut The most common way of partitioning a graph is the *minimum cut*. This problem consists in looking for the optimal bipartition of a graph, which is one that minimises the *cut* value. The *minimum cut* problem is well-studied and has efficient algorithms for solving it, even though it has an exponential amount of partitions that can be explored.

Some work on image segmentation using the *minimum cut* has been done [10] [11] [12]. But the results can only be qualified as proposals for recognised regions, such does [11] and would need further optimisation to be the final groups. For instance, [10] introduced a cut criterion using *minimum cut* on a graph but it was observed that it favors finding too small components and [12] tries to optimise the latter.

However, this problem still remains and *minimum cut* can produce bad groups on certain cases. Additionnaly, these cases often appear in natural images, where it is more important to group a large area of the background before small details.

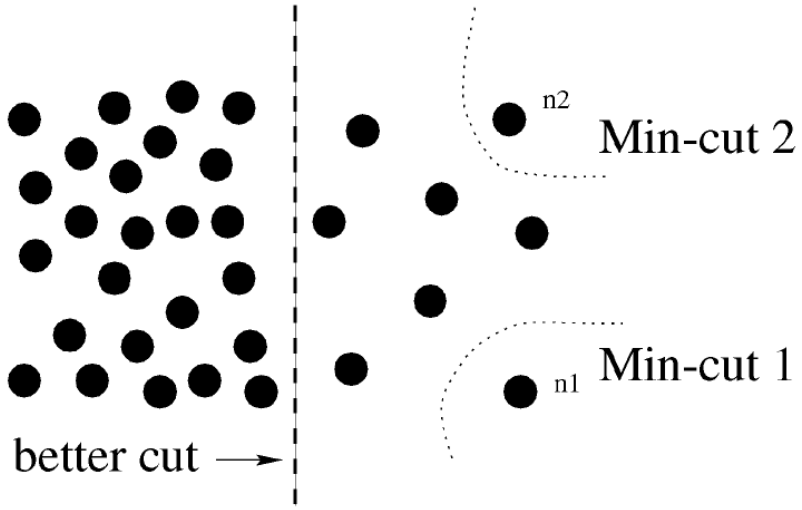


Figure 1.1: Bad partitioning from *minimum cut* in a particular case [13]

Normalised cut To improve this state, [13] proposed, instead of looking at the total weigh of the edges connecting two groups, to look at a fraction of the total edge connections to all nodes. This is the *normalised cut*, or $Ncut$. The disassociation measure can be formulated as:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)},$$

where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$, the total connection from the nodes in A to all nodes in the graph.

The goal is still to minimise the $Ncut$ value, but when cut is small for isolated pixels, the fraction will be greater because the similarity $assoc$ between this one pixel and all pixels will be small. This fraction's denominator will be bigger if the similarity between the selected pixels and all pixels is more important, which makes the fraction smaller.

As it follows, we can define the normalised association measure within groups, such as

$$Nassoc(A, B) = \frac{assoc(A, A)}{assoc(A, V)} + \frac{assoc(B, B)}{assoc(B, V)},$$

where $assoc(A, A)$ is the total weights of edges inside group A. This measure quantifies how tightly the nodes of a group are connected to each other. Deriving the $Ncut$ equation, we can define that

$$Ncut(A, B) = 2 - Nassoc(A, B).$$

It was shown by [14] and [13] that minimising the $Ncut$ exactly is a NP-complete problem. However, [13] proposes a method to appoximate discrete solutions efficiently for the *normalised cut* problem.

Indeed, the proposed algorithm starts by computing the associated similarities for graph $G = (V, E)$ for the image and store it in the matrices K and D.

Then, solve $(D - K)x = \lambda Dx$ for the eigenvectors with the smallest eigenvalues. The way this equation is derived is explicitly shown in [13]. The eigenvector with the second smallest is used to bipartition the graph by finding the best splitting point to minimise $Ncut$. After that, it is possible to use the other eigenvectors to bipartition the graph even more. Once this is done, we can recursively apply the same algorithm on the segmented groups, stoping when the maximum number of groups is reached or the partitions are not good enough depending on an empirical threshold.

Edge separators and spectral rounding Introduced by [15], edge separators of a graph are created by iteratively reweighting edges until the graph disconnects into a certain number of groups. At each iteration, only a small number of eigenvectors are computed to reweight the edges. This spectral rounding method produces directly discrete solutions, and seems to compare nicely against the $Ncut$ approximation by producing more human-like results [16].

1.4 Delimitations

The number of investigated algorithms and variants depend on the pre-study and objectives that make the most sense for 3D images. It is obviously also limited by the amount of time given to the degree project.

Chapter 2

2D Image Processing using the Graph Laplacian Operator

2.1 Theoretical 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, low-light imaging and image segmentation.

As it is known, an image filter consists of a function which outputs one pixel, 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 arbitrarily define how the pixels connect to each other, so we can 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. Pixels can be similar if they are spatially close or if they have the same color, or both. These similarities give us the so-called affinity matrix K.

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

According to [17], 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 such as $D_{jj} = diag\{\sum_i K_{ij}\}$ along its diagonal. D corresponds in fact to the degrees of each node in graph theory terminology. Other definitions of the Laplacian are shown later in 2.3.3.

Interestingly, we can define from [4] 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 in 2.3.1.

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, whereas K_B is 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, as shown in 2.3.2.

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

As stated in [4], 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. In fact, the eigenvalues decay quickly as shown in [1] and [18].

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

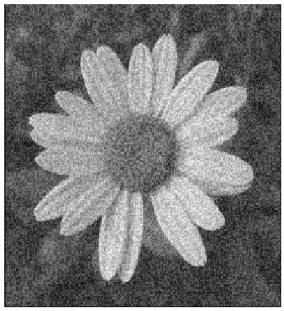
2.2 First implementation

Details The first implementation of this flow is aimed to denoise the input image. This represent an early stage of the internship. The sample pixels are selected in a spatially uniform manner. It uses the NLM from [2] as kernel method to compute the pixels similarity. The Laplacian matrix is defined through the Sinkhorn iterative algorithm [9], where the resulting approximated eigenvectors require to be orthogonalised, which can be done in one step, as discussed in [19]. We finally apply our approximated filter to the noisy image.

The implementation is generally greatly inspired by the MatLab implementation of [4].

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. But still, as a first result, this is better than nothing.

2.3 Algorithm variations

2.3.1 Sampling method

The sample requires to represent only 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 [20].

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 [21] [22].

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 [4].

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 [20].

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 [21].

2.3.2 Affinity function

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

Listing

Spatial Gaussian Kernel takes only into account 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] [4].

Non-Local Means (NLM) is similar to the bilateral kernel, a data-dependent filter, except that the photometric affinity is captured patch-wise [4].

Locally Adaptive Regression Kernel (LARK) uses the geodesic distance based on estimated gradients [9] [23].

2.3.3 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 $D = diag d_1^n$:

Laplacian Name	Formula	Symmetric	DC eigenvector	Spectral Range
Un-normalised	$D - K$	Yes	Yes	[0, n]
Normalised	$I - D^{-1/2} K D^{-1/2}$	Yes	No	[0, 2]
Random Walk	$I - D^{-1} K$	No	Yes	[0, 1]
"Sinkhorn" [9]	$I - C^{-1/2} K C^{-1/2}$	Yes	Yes	[0, 1]
Re-normalised	$\alpha(D - K)$, $\alpha = \mathcal{O}(n^{-1})$	Yes	Yes	[0, 1]

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

2.4 Algorithm variations 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 160 seconds on an Intel i5 processor and the generation takes not much more 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.

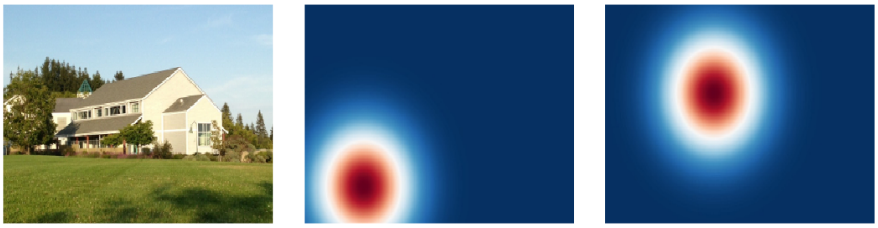
Spatial Gaussian Kernel The formula of this kernel is

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

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

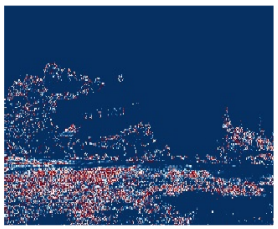
As we can see, the parameter is influencing on the normalisation of the values and gaussian standard deviation. The bigger it is, the more tolerant the spatial distance computation will be.

Photometric 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

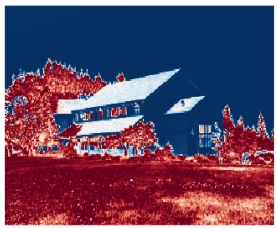


Affinity matrix of pixel 165x65
using photometric function

Affinity matrix of pixel 80x120
using photometric function

Affinity matrices with $h_z = 50$:

Original image



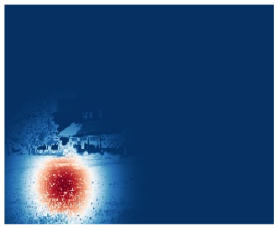
Affinity matrix of pixel 165x65
using photometric function

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

Remember that each matrix here is only the affinity matrix of one pixel. In a very heterogeneous image, the bilateral kernel will be useful to keep the spatial similarity, but with excluding very dissimilar neighbour pixels.

2.5 Adaptive Sharpening

Sharpening is usually using a highpass filter and therefore leads to amplifying high frequency noise components even more. We will have an effect of over-sharpening.

β controls the amount of sharpening of the image.

Bibliography

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

[2] 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.

[3] K. Dabov et al. "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering". In: *IEEE Transactions on Image Processing* 16.8 (Aug. 2007), pp. 2080–2095. ISSN: 1057-7149. DOI: 10.1109/TIP.2007.901238.

[4] 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).

[5] Hossein Talebi and Peyman Milanfar. "Asymptotic Performance of Global Denoising". en. In: *SIAM Journal on Imaging Sciences* 9.2 (Jan. 2016), pp. 665–683. ISSN: 1936-4954. DOI: 10.1137/15M1020708. URL: http://epubs.siam.org/doi/10.1137/15M1020708 (visited on 10/12/2017).

[6] Amin Kheradmand. "Graph-based image restoration". PhD thesis. University of California, Santa Cruz, 2016. URL: http://search.proquest.com/openview/29f820ea2c8cd1f23d36a6a2bc4d3e7b/1?pq-origsite=gscholar&cbl=18750&diss=y (visited on 10/12/2017).

[7] Wikipedia. *Difference of Gaussians — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Difference\%20of\%20Gaussians&oldid=747011119. [Online; accessed 19-October-2017]. 2017.

[8] Amin Kheradmand and Peyman Milanfar. "Non-Linear Structure-Aware Image Sharpening with Difference of Smoothing Operators". In: *Frontiers in ICT* 2 (Oct. 2015). DOI: 10.3389/fict.2015.00022.

[9] 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).

[10] Z. Wu and R. Leahy. "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.11 (Nov. 1993), pp. 1101–1113. ISSN: 0162-8828. DOI: 10.1109/34.244673.

[11] Francisco J. Estrada, Allan D. Jepson, and Chakra Chennubhotla. "Spectral Embedding and Min Cut for Image Segmentation." In: *Bmvc*. 2004, pp. 1–10.

[12] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Efficient Graph-Based Image Segmentation". en. In: *International Journal of Computer Vision* 59.2 (Sept. 2004), pp. 167–181. ISSN: 0920-5691, 1573-1405. DOI: 10.1023/B:VISI.0000022288.19776.77. URL: https://link.springer.com/article/10.1023/B:VISI.0000022288.19776.77 (visited on 10/17/2017).

[13] jianbo shi and jitendra malik. "normalized cuts and image segmentation". In: *ieee transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905. URL: http://ieeexplore.ieee.org/abstract/document/868688/ (visited on 10/11/2017).

[14] Christos H. Papadimitriou. "NP-completeness: A retrospective". en. In: *Automata, Languages and Programming*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, July 1997, pp. 2–6. ISBN: 978-3-540-63165-1 978-3-540-69194-5. DOI: 10.1007/3-540-63165-8_160. URL: https://link.springer.com/chapter/10.1007/3-540-63165-8_160 (visited on 10/18/2017).

[15] David A. Tolliver and Gary L. Miller. "Graph partitioning by spectral rounding: Applications in image segmentation and clustering". In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*. IEEE Computer Society, 2006, pp. 1053–1060.

[16] David A. Tolliver. *Spectral rounding and image segmentation*. Vol. 67. 11. 2006.

[17] 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).

[18] 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).

[19] 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).

[20] 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).

[21] 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.

[22] 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).

[23] H. Takeda, S. Farsiu, and P. Milanfar. "Kernel Regression for Image Processing and Reconstruction". In: *IEEE Transactions on Image Processing* 16.2 (Feb. 2007), pp. 349–366. ISSN: 1057-7149. DOI: 10.1109/TIP.2006.888330.