Master Thesis

–

Spectral Graph Theory and High Performance Computing

David Wobrock
david.wobrock@gmail.com

February 2, 2018

Abstract

TODO

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 image filters are used on smartphones, which implies a reasonable execution time with limited computational resources. Over 2 billion photos are shared daily on social media [1], with very high resolutions and most of the time some processing or filter is applied. The algorithm must be well-tuned to be deployed at such scale.

1.2 Objective

The aim of this degree project is not to explore and improve the state of image processing. Instead, the spectral methods are interesting. Those will inevitably expose eigenvalue problems, which may involve solving systems of linear equations.

Concerning the challenges about solving linear systems, on one hand, the size of the systems can be large considering high-resolution images with millions of pixels, or even considering 3D images. On the other hand, images are dense matrices. And so will be the resulting linear systems. Classically in physics, linear systems result from discretising partial differential equations which yield sparse matrices, and therefore most linear solvers are specialised in sparse systems.

We want to explore the performance of linear solvers on dense problems, their stability and convergence. This will include preconditioning the linear systems, especially using domain decomposition methods, and analyse their behavior on dense systems.

1.3 Related work

1.3.1 Spectral graph theory

Spectral graph theory has a long history starting with matrix theory and linear algebra that were used to analyse adjacency matrices of graphs. It consists in studying the properties of graphs in relation to the eigenvalues and eigenvectors of the adjacency or Laplacian matrix. The eigenvalues of such a matrix are called the spectrum of the graph. The second smallest eigenvalue has been called "algebraic connectivity of a graph" by Fiedler [2], and is therefore also known as *Fiedler value*, because it contains interesting information about the graph. Indeed, it can show if the graph is connected, and by extending this property, we can count the number of connected components in the graph through the eigenvalues of the graph Laplacian.

The field of spectral graph theory is very broad and the eigendecomposition of graphs is used in a lot of areas. It was first applied in chemistry because eigenvalues can be associated with the stability of molecules. Spectral graph theory has many other applications such as graph colouring, graph isomorphism testing, random walks and graph partitioning among others.

One of the most complete works about spectral graph theory is [3] by Fan Chung. This monograph exposes many properties of graphs, the power of the spectrum and how spectral graph theory links the discrete world to the continuous one.

Laplacian matrix Since the adjacency matrix of a graph only holds basic information about it, we usually augment it to the Laplacian matrix. Multiple definitions of the Laplacian matrix are given in [3] and [1], and each one holds different properties. The most common ones are the normalised Laplacian and the Random Walk Laplacian. However, more convenient formulations, like the "Sinkhorn" Laplacian [4] and the re-normalised Laplacian [1] [5], have been proposed since.

The Spectral Theorem Some Laplacian definitions result in a symmetric matrix, which is a property that is particularly interesting for spectral theory because of the Spectral Theorem [6]. Let S be a real symmetric matrix of dimension n, then

$$S = \Phi\Pi\Phi^T = \sum_{i=1}^{n} \lambda_i \phi_i \phi_i^T,$$

the eigendecomposition of S with $\Phi = [\phi_1 \phi_2 \dots \phi_n]$ the matrix of eigenvectors of S and Π the diagonal matrix of the eigenvalues of S. We note that the eigenvalues of S are real and that the eigenvectors are orthogonal, i.e., $\Phi^T\Phi = I$, with I the identity matrix of an appropriate rank.

The Laplacian is the foundation of the heat equation, fluid flow and essentially all diffusion equations. It can generally be thought that the Laplacian

operator is a center-surround average [1] of a given point. Applying the graph Laplacian operator on an image provides useful information about it and enables possibilities of interesting image processing techniques.

1.3.2 Image processing - 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 should hold the Shannon-Nyquist theorem [7], stipulating a sufficient condition on the number of samples required to discretise a countinous signal without losing information. Noise comes from the light acquisition 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 generally 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 [7] suggests that the Non-local 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 predicts its value by using the pixels in its neighbourhood.

In [8], 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 [9]. 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 z by,

$$z = Wy,$$

where W is our data-dependent global filter, a $N \times N$ matrix, N the number of pixels in the picture. This kind of filters are considered in this report.

1.3.3 Linear solvers and domain decomposition methods

Solving a system of linear equations such that

$$Ax = b,$$

is often critical in scientific computing. When discretising equations coming from physics for example, a huge linear system can be obtained. Multiple methods exist to solve such systems, even when the system is large and expensive to compute. We present in the following the most used and known solvers.

Direct solvers The most commonly used solvers for systems of linear equations are direct solvers. They provide robust methods and optimal solutions to the problem. However, they can be hard to parallelise and have difficulties with large input. The most famous is the backslash operator from MATLAB which performs tests to determine which special case algorithm to use, but ultimately falls back on a LU factorisation [10]. The LU factorisation, closely related to Gaussian elimination, is hard to parallelise. A block version of the LU factorisation exists that can be parallelised more easily. Other direct solvers, like MUMPS [11], exist but generally they reach their computational limit above 10^6 degrees of freedom in a 2D problem, and 10^5 in 3D.

Iterative solvers For large problems, iterative methods must be used to achieve reasonable runtime performances. The two types of iterative solvers are fixed-point iteration methods and Krylov type methods. Both require only a small amount of memory and can often be parallelised. The main drawback is that these methods tend to be less robust than direct solvers and convergence depends on the problem. Indeed, ill-conditioned input matrices will be difficult to solve correctly by iterative methods. Generally, Krylov methods are preferred over fixed-point iteration methods because they perform better. The most relevant iterative Krylov methods are the conjugate gradient (CG) and GMRES [12].

To tackle the ill-conditioned matrices problem, there is a need to precondition the system.

Preconditioners - Domain decomposition methods One of the ways to precondition systems of linear equations is to use domain decomposition. The idea goes back to Schwarz who wanted to solve a Poisson problem on a complex geometry. He decomposed the geometry into multiple smaller simple geometric forms, making it easy to work on subproblems. This idea has been extended and improved to propose fixed-point iterations solvers for linear systems. However, Krylov methods expose better results and faster convergence, but domain decomposition methods can actually be used as preconditioners to the system. The most famous Schwarz preconditioners are the Restricted Additive Schwarz (RAS) and Additive Schwarz Method (ASM). For example, the formulation of

the ASM preconditioning matrix

$$M_{ASM}^{-1} = \sum_i R_i^T A_i^{-1} R_i,$$

with i subdomains and R_i the restriction matrix of A to the i-th subdomain. With such a preconditioner we will be able to solve

$$M^{-1}Ax = M^{-1}b$$

which exposes the same solution as the original problem.

Domain decomposition methods will also be an important topic of this degree project. These methods are usually applied to solve problems of linear algebra involving partial differential equations (PDEs). Solving the discretised problem leads to solving linear systems.

Our main reference will be [13] which focuses on the parallel linear iterative solvers for systems of linear equations. Domain decomposition methods are naturally parallel which is convenient for the current state of processor progress. Without going into the details, we will make use of Schwarz methods for preconditioning and iterative Krylov subspace methods as solvers.

1.4 Delimitations

TODO

Chapter 2

Image Processing using the graph Laplacian operator

Multiple image processing filters can be built using the graph Laplacian operator. As Milanfar mentions in [1], smoothing, deblurring, sharpening, dehazing, and other filters can be created. Laplacian operators can also be used as the basis for compression artifact removal, low-light imaging and image segmentation.

We shall consider an adapted version of the proposed algorithm in [9] to solve the eigenvalue problem by solving linear systems. Let's introduce step-by-step the algorithm and how we consider solving it.

2.1 Algorithm

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 a vector of weights exists 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,

$$z = Wy.$$

The filter matrix W considered here is data-dependent and built upon the input image y.

Image as graph Let's think of an image as a graph. Each pixel is a node and has edges to other nodes. The simplest way to connect pixels to other pixels is

their direct neighbours, in which case each node has four edges. To avoid losing any information, we will instead consider the case of a complete graph, each node connects to all other nodes.

To preserve the image information in the graph, the graph edges will be assigned a weight, measuring the similarity[1]. There are multiple ways the similarity can be defined.

The most intuitive definition considers spatial proximity. This means that similar pixels are spatially close, which, translated to a basic filter, is the same as a Gaussian filter which computes a weighted average of the pixel's neighbourhood and produces what is known as Gaussian blur. Another similarity definition is to consider the pixel's color. A good compromise is to consider an average of both, spatial and color closeness, with a certain weighting.

From this definition can be computed the adjacency matrix of the graph taking into account the edge weights. We will call this matrix the affinity matrix[2] K which represents the similarity of each pixel to every other pixel in the image. Consequently, this matrix is symmetric and of size $N \times N$ with N the number of pixels in the image, and, depending on the similarity function, its values $0 \leq K_{ij} \leq 1$.

By extending this affinity matrix, we obtain the graph Laplacian $\mathcal{L}$, used to build the filter matrix W.

Building the filter Multiple graph Laplacian definitions, more or less equivalent, exist and can have slightly different properties. In the case of image smoothing, the filter W is defined such as $\mathcal{L} = I - W$ [1] and so $W = I - \mathcal{L}$.

However, all these matrices K, $\mathcal{L}$ and W represent huge computational costs. Only storing one of these matrices is already a challenge since they have a size of N^2.

Approximation by sampling and Nyström extension To avoid storing any of those huge matrices, approximation will be necessary. It starts by sampling the image and only select a subset of p pixels, with $p \ll N$. Numerically, p should represent around 1% of all image pixels. The rows and columns of the matrix K are re-organised such as

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

with K_A being the affinities between the sampled pixels and thus of size $p \times p$. K_B holds the affinities between the sampled pixels and the remaining pixels and is therefore of size $p \times (N-p)$. Finally, the submatrix K_C contains the similarities between the remaining pixels. Knowing that K_C is of size $(N-p) \times (N-p)$ and that $p \ll N$, this submatrix is huge.

[1] Also called affinity.

[2] Or similarity matrix or kernel matrix

To have a numerical approximation of a symmetric matrix K, we can use the eigendecomposition:

$$K = \Phi \Pi \Phi^T$$

where Φ are the orthonormal eigenvectors of K stored as a matrix and Π contains the eigenvalues of K. Nyström suggests to approximate this eigendecomposition of the first p eigenvectors with

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

with $K_A = \Phi_A \Pi_A \Phi_A^T$. The approximated matrix would be of the form

$$\tilde{K} = \begin{bmatrix} K_A & K_B \\ K_B^T & K_B^T K_A^{-1} K_B \end{bmatrix}.$$

We can clearly see that the huge submatrix is now approximated.

Eigendecomposition We need to compute the largest eigenvalues of the filter. For the approximation, we will actually need to compute the largest eigenvalues of the sampled pixels of the filter to use the Nyström extension. We know that these eigenvalues are contained by the eigenvalues of the full filter, which is known as the interlacing property of principal submatrices. We know also that computing the largest eigenvalues of the submatrix filter is equivalent to computing the smallest eigenvalues of the corresponding submatrix of the Laplacian operator. The proofs to these statements can be found in Appendix A.

The goal of this observation is the way of computing the eigenvalues. For the largest eigenvalues, the most famous algorithm is the power method. For the smallest eigenvalues, the inverse power method is usually used. The latter requires either to invert a matrix, or to solve a linear system. We will solve systems of linear equations to compute the first eigenvalues of the Laplacian in order to observe the behavior of solvers on the dense matrices.

The computation also returns the eigenvectors of the Laplacian submatrix $\mathcal{L}_A$. With μ_A the eigenvalues of $\mathcal{L}_A$ and λ_A the eigenvalues of W_A, if $\mu_A = 1 - \lambda_A$ then

$$\mathcal{L}_a \, x = \mu_A x \Leftrightarrow W_A x = \lambda_A x.$$

They share the same eigenvectors. We will therefore approximate the eigenvectors of $\mathcal{L}$ using the Nyström extension, and use them to .

Algorithm 1 Image processing using graph Laplacian

Input: y an image of size $n \times m$
Output: z the output image
$N \leftarrow n * m$
{Sampling}
Sample p pixels, $p \ll N$
{Kernel matrix approximation}
Compute K_A (size $p \times p$) and K_B (size $p \times (N-p)$) such as $K = \begin{bmatrix} K_A & K_B \\ K_B^T & K_C \end{bmatrix}$
Compute the Laplacian $\mathcal{L}_A$
{Eigendecomposition}
Compute the smallest eigenvalues Π_A of $\mathcal{L}_A$ and the associated eigenvectors Φ_A
Nyström extension $\tilde{\Phi} \leftarrow \begin{bmatrix} \Phi_A \\ K_B^T \Phi_A \Pi_A^{-1} \end{bmatrix}$
TODO filter and Laplacian relation
{Apply the filter}
$\tilde{W} \leftarrow \tilde{V} S \tilde{V}^T$
$z \leftarrow \tilde{W} y$

2.2 Variations

2.2.1 Sampling method

The sample requires to represent only less than 1% of the pixels of the image [9]. To achieve this, we can use different approaches. The chosen method is decisive for the application of the Nyström method.

Random sampling (RS) most common and simple sampling scheme, but no deterministic guarantee of the output quality. It 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 [14].

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 [15] [16].

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 [9].

Figure 2.1: Spatially uniform sampling. Red pixels are sampled. Here 100 pixels are sampled, which represents 0.04% of all pixels

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 [14].

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 [15].

2.2.2 Affinity function

The kernel function $\mathcal{K}_{ij}$ measures the similarity between the pixel y_i and y_j. The chosen function is important because it decides on which features the similarity of pixels will be evaluate and the tolerance of it. Some of the most used affinity functions are:

Spatial Gaussian Kernel takes only into account the spatial distance between two pixels [1]. The formula of this kernel is, with $\forall i, j \in [1, N]$, x_i the coordinate vector of a pixel and h_x a normalisation parameter,

$$K(y_i, y_j) = exp(-\frac{||x_i - x_j||^2}{h_x^2}).$$

The parameter is influencing on the normalisation of the values and gaussian standard deviation. The greater it is, the more tolerant the spatial distance computation will be.

Photometric Gaussian Kernel considers the intensity and color similarity of the pixels [1]. The formula of this kernel is, with z_i the color or grayscale of a pixel,

$$K(y_i, y_j) = exp(-\frac{||z_i - z_j||^2}{h_z^2}).$$

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 one of the most used kernel which smooths images by a nonlinear combination of the spatial and photometric gaussian kernels [1] [9] [17]:

$$K(y_i, y_j) = exp(-\frac{||x_i - x_j||^2}{h_x^2})exp(-\frac{||z_i - z_j||^2}{h_z^2}).$$

To generate the example, we use the famous grayscale image of Barbara of dimension 512×512 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. We use a spatially uniform sampling technique and select 0.1% of the pixels. These are two affinity vectors, the first one is of a pixel on the table leg and the second around Barbara's eye. Keep in mind that each affinity image shown represents only one row of the affinity matrix K.

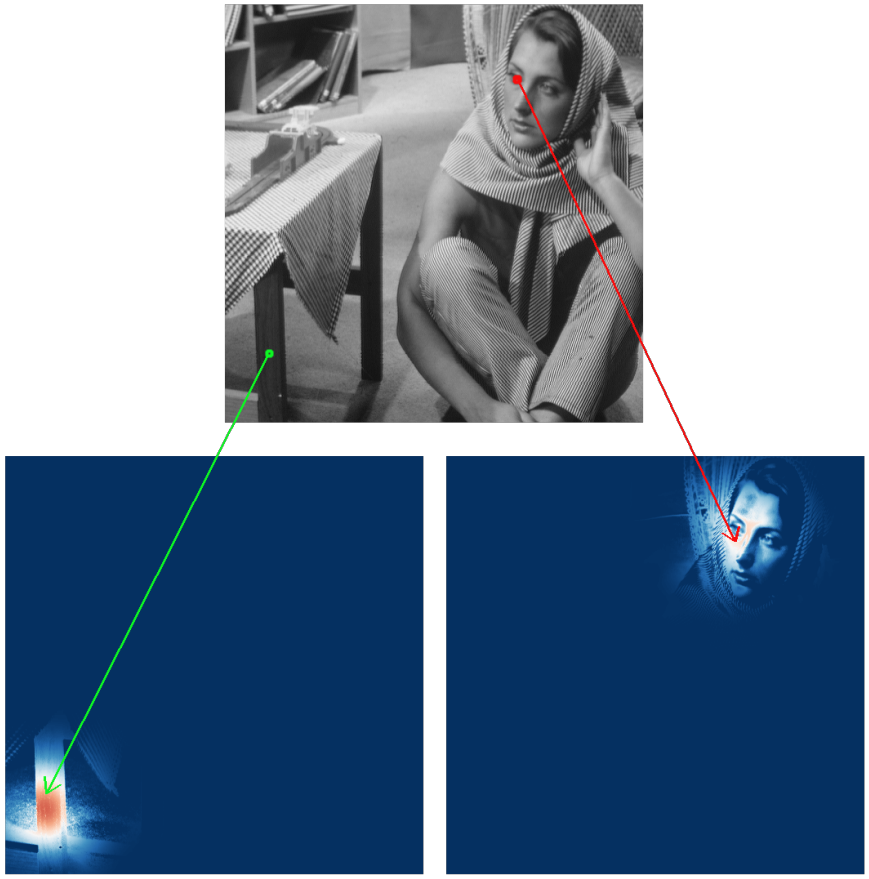

Figure 2.2: Affinity matrices with $h_x = 50$ and $h_z = 35$

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

Non-Local Means (NLM) is similar to the bilateral kernel, a data-dependent filter, except that the photometric affinity is captured patch-wise [9] [18].

Locally Adaptive Regression Kernel (LARK) uses the geodesic distance based on estimated gradients [4] [19].

2.2.3 Graph Laplacian operator

The graph Laplacian operator 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 the eigendecomposition of the matrix.

The spectral range, corresponding to the range of the eigenvalues, is important because we can 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 $D = diag\{d_i\}$:

Laplacian Name	Formula	Symmetric	Spectral Range
Un-normalised	$D - K$	Yes	[0, n]
Normalised	$I - D^{-1/2}KD^{-1/2}$	Yes	[0, 2]
Random Walk	$I - D^{-1}K$	No	[0, 1]
"Sinkhorn" [4]	$I - C^{-1/2}KC^{-1/2}$	Yes	[0, 1]
Re-normalised [5]	$\alpha(D - K)$, $\alpha = \mathcal{O}(N^{-1})$	Yes	[0, n]

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

2.3 Implementation

2.3.1 Algorithm details

In our algorithm, we shall use spatially uniform sampling for the ease of implementation and robustness. The kernel function is the bilateral one.

2.3.2 Results

TODO some results + performances

Chapter 3

Conclusion

3.1 Discussions

3.2 Perspectives

Image processing On the image processing side, [20] proposes an enhancement. [20] argues that the eigendecomposition remains computationaly too expense and shows results of an improvement. The presented results and performances seem astonishing, however, the method is hardly described and replicable with difficulty. This is understandable since this algorithm seems to be in the latest Pixel 2 smartphone by Google and they want to preserve their advantage in the field of image processing.

Linear solvers

Bibliography

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

[2] Miroslav Fiedler. "Algebraic connectivity of graphs". eng. In: *Czechoslovak Mathematical Journal* 23.2 (1973), pp. 298–305. ISSN: 0011-4642. URL: https://eudml.org/doc/12723 (visited on 10/26/2017).

[3] Fan R. K. Chung. *Spectral graph theory*. Vol. 92. CBMS Regional Conference Series in Mathematics. Published for the Conference Board of the Mathematical Sciences in Washington, DC; by the American Mathematical Society in Providence, RI, 1997, pp. xii+207. ISBN: 0-8218-0315-8.

[4] 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).

[5] Peyman Milanfar and Hossein Talebi. "A new class of image filters without normalization". In: *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3294–3298.

[6] Hao Zhang, Oliver Van Kaick, and Ramsay Dyer. "Spectral mesh processing". In: *Computer graphics forum*. Vol. 29. Wiley Online Library, 2010, pp. 1865–1894.

[7] 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.

[8] 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.

[9] 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).

[10] MathWorks. *MATLAB - Solve systems of linear equations*. https://fr.mathworks.com/help/matlab/ref/mldivide.html. (Visited on 11/21/2017).

[11] P. R. Amestoy et al. "A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling". In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 15–41.

[12] Y. Saad and M. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (July 1986), pp. 856–869. ISSN: 0196-5204. DOI: 10.1137/0907058. URL: http://epubs.siam.org/doi/abs/10.1137/0907058 (visited on 11/15/2017).

[13] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An introduction to domain decomposition methods*. Algorithms, theory, and parallel implementation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015. ISBN: 978-1-611974-05-8. URL: http://dx.doi.org/10.1137/1.9781611974065.ch1.

[14] 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).

[15] 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.

[16] 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).

[17] C. Tomasi and R. Manduchi. "Bilateral filtering for gray and color images". In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. Jan. 1998, pp. 839–846. DOI: 10.1109/ICCV.1998.710815.

[18] C. Kervrann and J. Boulanger. "Optimal Spatial Adaptation for Patch-Based Image Denoising". In: *IEEE Transactions on Image Processing* 15.10 (Oct. 2006), pp. 2866–2878. ISSN: 1057-7149. DOI: 10.1109/TIP.2006.877529.

[19] 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.

[20] Hossein Talebi and Peyman Milanfar. "Fast Multilayer Laplacian Enhancement". In: *IEEE Transactions on Computational Imaging* 2.4 (Dec. 2016), pp. 496–509. ISSN: 2333-9403, 2334-0118. DOI: 10.1109/TCI.2016.2607142. URL: http://ieeexplore.ieee.org/document/7563313/ (visited on 01/25/2018).

[21] 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).

Appendix A

Proof of filter and Laplacian eigenvalues equivalence

The filter W is built on top of the kernel matrix K measuring the similarity between each pixel. The most popular kernel functions are the *Bilateral filter* [17] and the *Non-local Mean filter* [18] to measure these similarities. The kernel functions create a symmetric positive semi-definite (PSD) matrix K, so the eigenvalues of K are non-negative, $\lambda_K \geq 0$, as described in [20]. Also, from the definition of the filters, $\forall i, j \in [1, N]$, N the number of pixels, the values of the affinity matrix K are non-negative $K_{ij} \geq 0$.

Without loss of generality, we shall use a Laplacian operator that is symmetric positive definite (SPD) by definition. The filter is implicitely defined by [21] as $W = I - \mathcal{L}$ and W is SPD as we consider a SPD Laplacian.

We can say that for a principal submatrix W_A of size $p \times p$ of the matrix W of size $N \times N$ with $p < N$, the eigenvalues $\lambda_i^W \leq \lambda_i^{W_A} \leq \lambda_{i+N-p}^W$. This is the interlacing property, meaning in general, $\lambda_1^W \leq \lambda_i^{W_A} \leq \lambda_N^W$.

Proof of interlacing Let A be a symmetric matrix of size n, λ_n^A be the largest eigenvalue of A and λ_1^A the smallest one. Let R be the restriction operator, such as, with u a non-zero vector, $Ru = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ for example. This defines RAR^T a $s \times s$ principal submatrix of A with $s \in [1; n]$. Suppose the remaining rows and columns of A in RAR^T are indexed by S of size s.

Let $\mathcal{U} \in \mathbb{R}^s$ and $u \in \mathbb{R}^n$ with $\begin{cases} u_i = \mathcal{U}_i & \text{if } i \in S \\ u_i = 0 & \text{if } i \notin S \end{cases}$. Given a $k \in [1; s]$, the Courant-Fischer theorem, involving the Rayleigh-Ritz quotient, implies that,

$$max\left(\frac{\langle Au, u\rangle}{\langle u, u\rangle}\right) = max\left(\frac{\langle RAR^T\mathcal{U}, \mathcal{U}\rangle}{\langle \mathcal{U}, \mathcal{U}\rangle}\right) \geq \lambda_k^A.$$

So $\lambda_k^{RAR^T} \geq \lambda_k^A$. More over, in the other way, we get

$$min\left(\frac{\langle Au, u\rangle}{\langle u, u\rangle}\right) = min\left(\frac{\langle RAR^T\mathcal{U}, \mathcal{U}\rangle}{\langle \mathcal{U}, \mathcal{U}\rangle}\right) \leq \lambda_{k+n-s}^A.$$

And so again, $\lambda_k^{RAR^T} \leq \lambda_{k+n-s}^A$. This concludes the proof, showing that the eigenvalues of the submatrix are bounded by the eigenvalues of the original matrix. More precisely, we proved the interlacing property of the eigenvalues of RAR^T such as

$$\lambda_k^A \leq \lambda_k^{RAR^T} \leq \lambda_{k+n-s}^A.$$

From the definition of the filter $W = I - \mathcal{L}$, we have the submatrix $W_A = I - \mathcal{L}_A$, with I being the identity of appropriate order. For the algorithm, we need to compute the largest eigenvalues of W_A.

Theorem Computing the largest eigenvalues of W_A is equivalent to computing the smallest eigenvalues of $\mathcal{L}_A$.

Proof

$$\begin{aligned} W_A x = \lambda x &\Leftrightarrow (I - \mathcal{L}_A)x = \lambda x \\ &\Leftrightarrow x - \mathcal{L}_A\, x = \lambda x \\ &\Leftrightarrow \mathcal{L}_A\, x = x - \lambda x \\ &\Leftrightarrow \mathcal{L}_A\, x = (1 - \lambda)x \end{aligned} \tag{A.1}$$

So the eigenvalues of the Laplacian submatrix $\mu = 1 - \lambda$. We can thus get the greatest eigenvalues of W_A by computing the smallest eigenvalues of $\mathcal{L}_A$.

Speed of convergence For both these problems, finding the greatest and smallest eigenvalues, the most famous methods are, respectively, the power method and inverse power method[1].

For the power iteration, the convergence rate is $|\frac{\lambda_2}{\lambda_1}|$, with λ_2 being the second largest eigenvalue. We know that $\lambda_2^{W_A} \leq \lambda_1^{W_A} \leq 1$ and thus $\frac{\lambda_2^{W_A}}{\lambda_1^{W_A}} \leq \frac{\lambda_1^{W_A}}{\lambda_1^{W_A}} = 1$. The convergence rate is lower than 1. The method is fast if the rate is small and

[1]Those are also called power iteration and inverse iteration. The inverse method has a variant called inverse subspace iteration, to find the associated subspace to the eigenvalues.

slow if the rate is close to 1. So the closer the two eigenvalues are, the slower the method converges.

The inverse iteration has a speed of convergence of $|\frac{\mu_1}{\mu_2}|$, with μ_2 the second smallest eigenvalue. Again, we know that $0 \leq \mu_1^{\mathcal{L}_A} \leq \mu_2^{\mathcal{L}_A}$. So the convergence speed is also lower than 1.

We come to the conclusion that both methods depend on the spacing between the eigenvalues. The closer they are, the more iterations will be required to converge. The difference of convergence speeds for both methods therefore depends on the distance between the largest eigenvalues and the distance between the smallest ones.

Inverse iterations implies either to compute the inverse of the matrix $x_{k+1} = A^{-1}x_k$, or to solve a system of linear equations $Ax_{k+1} = x_k$. Since the image processing context suggests having dense matrices, we want to explore the performances of Krylov methods and domain decomposition methods (e.g. the Additive Schwarz method) on such dense matrices.