Master Thesis

–

Spectral Graph Theory and High Performance Computing

David Wobrock david.wobrock@gmail.com

January 29, 2018

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 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 most of the time some processing or filter applied to them. The algorithm must be well-tuned to be deployed at such scale.

We want to take a look into the used spectral methods, see if they reveal systems of linear equations and what can be done with those dense pixel matrices in the nice use case of image processing.

1.2 Objective

The objective of this thesis is to, in a first place, understand and implement the image processing algorithm shown by Milanfar's work on 2D images. We want to observe the performance and results of the algorithm to validate the usage of intrinsic spectral properties of an image. We know that the spectrum of the graph associated to an image can serve for image segmentation. This segmentation can lead to image compression. More classic image processing can also be achieved with the algorithm, such as denoising, deblurring, sharpening, etc.

The next step includes extending these techniques on high-resolution and/or 3D images. Given the results, new scalable processing approaches for 3D images could be explored. These approaches, even though faster than other filters, still require heavy computational costs. That is why we want to apply the algorithm on high-performance architectures in order to achieve reasonable computation times. The HPC component will help solving systems of linear equations with

domain decomposition methods, which are naturally parallel methods, but also making the overall computations parallel.

1.3 Related work

We will explore three topics concerning this project. First of all, we will have an overview of what spectral graph theory is about. Then, we will dive into image processing using the graph Laplacian operator, focusing on denoising. We show a classical algorithm, a newer spectral approach and then a short comparison of these two approaches. Finally, we touch a few words on domain decomposition methods and Krylov methods.

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 real 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.

Cheeger's inequality One of the most fundamental theorems of spectral graph theory concerns the Cheeger's inequality and Cheeger constant. It approximates the sparsest cut of a graph with the second eigenvalue of its Laplacian.

The Cheeger constant [7] measures the degree of "bottleneck" of a graph, useful for constructing well-connected graphs. Considering a graph G of n vertices, the Cheeger constant h is defined as

$$h(G) = min_{0<|S|\leq \frac{n}{2}} \frac{|\partial S|}{|S|},$$

where S is a subset of the vertices of G and ∂S is the *edge boundary* of S to have all edges with exactly one endpoint in S, or formally

$$\partial S = u, v \in V(G) : u \in S, v \notin S,$$

with $V(G)$ the vertices of graph G.

Cheeger's inequality defines a bound and relationship on the smallest positive eigenvalue of the Laplacian matrix $\mathcal{L}$ such as

$$\lambda_1(\mathcal{L}) \geq \frac{h^2(\mathcal{L})}{4}.$$

When the graph G is d-regular, thanks to [8], we also have an inequality between $h(G)$ and the second smallest eigenvalue λ_2 such as

$$\frac{1}{2}(d - \lambda_2) \leq h(G) \leq \sqrt{2d(d - \lambda_2)},$$

where $d - \lambda_2$ is also called the *spectral gap*.

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 - The case of 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 [9], 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 [9] 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 i, predicts its value by using the pixels in its neighbourhood.

In [10], 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 [11]. 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 $N \times N$, using a similarity function between two pixels. As the size of W and

[1] Also called kernel matrix or similarity matrix

K can grow very large, we must sample the image and compute an approximated $\tilde{W}$ and $\tilde{K}$ using the Nyström extension. Because those are symmetric, they can in fact be approximated through the eigenvalues and eigenvectors with

$$\tilde{K} = \Phi_K \Pi_K \Phi_K^T.$$

Generally, as proposed in [11] and [12], to improve the denoising performance of global filters, pre-filtering techniques are used. It is proposed to first apply a Non-local 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 [12] 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 [13] 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.

1.3.3 Solving systems of linear equations 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 [14]. 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 [15], 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 [16].

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). The discrete equations this is linked to are $F(u) = b \in \mathbb{R}^n$, with n the number of degrees of freedom of the discretisation. Whether F is linear or not, solving this problem leads to solving linear systems.

Our main reference will be [17] 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

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

We will only consider image denoising in a first time, since it is the most straightforward application and the algorithm can be derived to many other applications.

Chapter 2

Image Processing using the Graph Laplacian Operator

2.1 Theoretical basis and algorithm overview

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.

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,

$$z = Wy.$$

Nowadays, the filter matrix W is often data-dependent and built on the input image y.

Let's think of an image as a graph now. 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. 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. This matrix is symmetric and contains for each pixel, its similarity to every other pixel in the image. Thus, it has a size of $N \times N$, N the number of pixels in the image.

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

According to [18], to build a denoising filter from the Laplacian, we have

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

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

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

with D a diagonal matrix with the normalising factors along its diagonal such as $\forall i \in [1, N], D = diag\{\sum_j K_{ij}\}$. D corresponds in fact to the degrees of each node in graph theory terminology. Other definitions of the Laplacian are shown later in 2.2.3.

Interestingly, we can define from [11] that, we can compute the eigendecomposition a symmetric positive definite filter W such as

$$W = VSV^T,$$

V being a matrix of 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, just as K, is a square matrix with $N * N$ elements, N the number of pixels in the picture.

Approximation by Nyström Extension To compute the filter matrix, the first step is to compute the affinity matrix K. Both matrices have the same size and are computationally expensive. That is why we approximate the affinity matrix.

We start by sampling p pixels, such as $p \ll N$. Different sampling techniques exist and are shown in 2.2.1.

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 $p \times p$ between the sample pixels, whereas K_B is the affinity matrix of size $p \times (N - p)$ 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 2.2.2. Once computed, we can approximate the eigenvectors Φ and eigenvalues Π of K thanks to the eigendecomposition of K_A.

As stated in [11], 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 first eigenvectors of K using the Nyström extension [19]

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

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 [20]. This means that the whole matrix is mainly defined by the first eigenelements.

Computing the filter From this eigendecomposition approximation of K, we can then approximate the eigenvectors and eigenvalues of W. Indeed, with the same procedure as for K, we can first compute similarly W_A and W_B such as

$$W = \begin{bmatrix} W_A & W_B \\ W_B^T & W_C \end{bmatrix}$$

We could apply the Nyström extension again, but the resulting eigenvectors are not orthonormal. To approximate orthonormal eigenvectors, we can use the proposed method by [19] which consists, if W_A is positive definite, in computing a matrix Q such as

$$Q = W_A + W_A^{-\frac{1}{2}} W_B W_B^T W_A^{-\frac{1}{2}},$$

where $W_A^{-\frac{1}{2}}$ is the inverse of the symmetric positive definite square root of W_A. By diagonalising Q we obtain $Q = V_Q S_Q V_Q^T$. As it is proven in the appendix of [19], the approximation of the filter matrix $\tilde{W} = \tilde{V} \Pi_Q \tilde{V}^T$ with

$$\tilde{V} = \begin{bmatrix} W_A \\ W_B^T \end{bmatrix} W_A^{-\frac{1}{2}} V_Q S_Q^{-\frac{1}{2}}.$$

$\tilde{V}$ is a base of orthonormal eigenvectors where $\forall i, \tilde{V}_i$ is the i-th eigenvector of $\tilde{W}$, which can numerically be shown by $\tilde{V}^T \tilde{V} = I$, $||\tilde{V}_i|| = 1$ and $\forall j, i \neq j, \tilde{V}_i \tilde{V}_j = 0$.

Summary The algorithm and figure below summarises the processing chain.

Algorithm 1 Global image processing

Input: y an image of size $n \times m$
Output: z the output image

$N \leftarrow n * m$
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 $K \leftarrow \begin{bmatrix} K_A & K_B \\ K_B^T & K_C \end{bmatrix}$
Eigendecomposition $K_A \leftarrow \Phi_A \Pi_A \Phi_A^T$
Nyström extension $\tilde{\Phi} \leftarrow \begin{bmatrix} \Phi_A \\ K_B^T \Phi_A \Pi_A^{-1} \end{bmatrix}$
$\tilde{K} \leftarrow \tilde{\Phi} \Pi_A \tilde{\Phi}^T$
{Filter approximation}
Compute W_A (size $p \times p$) and W_B (size $p \times (N-p)$) such as $W \leftarrow \begin{bmatrix} W_A & W_B \\ W_B^T & W_C \end{bmatrix}$
with $W \leftarrow I + \alpha(K - D)$
Eigendecomposition $W_A \leftarrow V_A S_A V_A^T$
———————-
Nyström extension $\tilde{V} \leftarrow \begin{bmatrix} V_A \\ W_B^T V_A S_A^{-1} \end{bmatrix}$
Gram-Schmidt orthonormalisation of $\tilde{V}$
———-OR———-
Compute $W_A^{-\frac{1}{2}} \leftarrow V_A S_A^{-\frac{1}{2}} V_A^T$
$Q \leftarrow W_A + W_A^{-\frac{1}{2}} W_B W_B^T W_A^{-\frac{1}{2}}$
Eigendecomposition $Q \leftarrow V_Q S_Q V_Q^T$
$\tilde{V} \leftarrow \begin{bmatrix} W_A \\ W_B^T \end{bmatrix} W_A^{-\frac{1}{2}} V_Q S_Q^{-\frac{1}{2}}$
———————-
$\tilde{W} \leftarrow \tilde{V} S \tilde{V}^T$
$z \leftarrow \tilde{W} y$

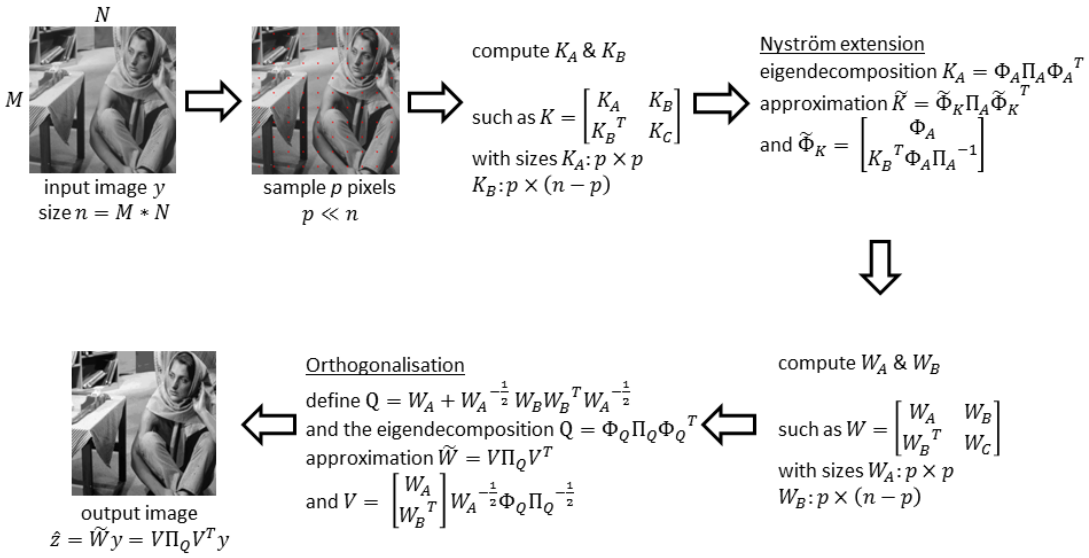

Figure 2.1: The image processing algorithm

TODO IMAGE A REFAIRE avec bonnes notations

2.2 Algorithm variations

2.2.1 Sampling method

The sample requires to represent only less than 1% of the pixels of the image [11]. 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 [21].

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 [22] [23].

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 [11].

Figure 2.2: 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 [21].

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 [22].

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. To illustrate the impact of the affinity function, here is a list of affinity functions, some with 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 the examples, we use the famous image of Barbara of dimension 512×512 pixels (grayscale image). We use a spatially uniform sampling technique and select 0.1% of the pixels. We show two affinity matrices for some affinity functions, the first one is of a pixel on the table leg and the second on Barbara's eye.

Keep in mind that each affinity image shown represents only one row of the affinity matrix K.

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,

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

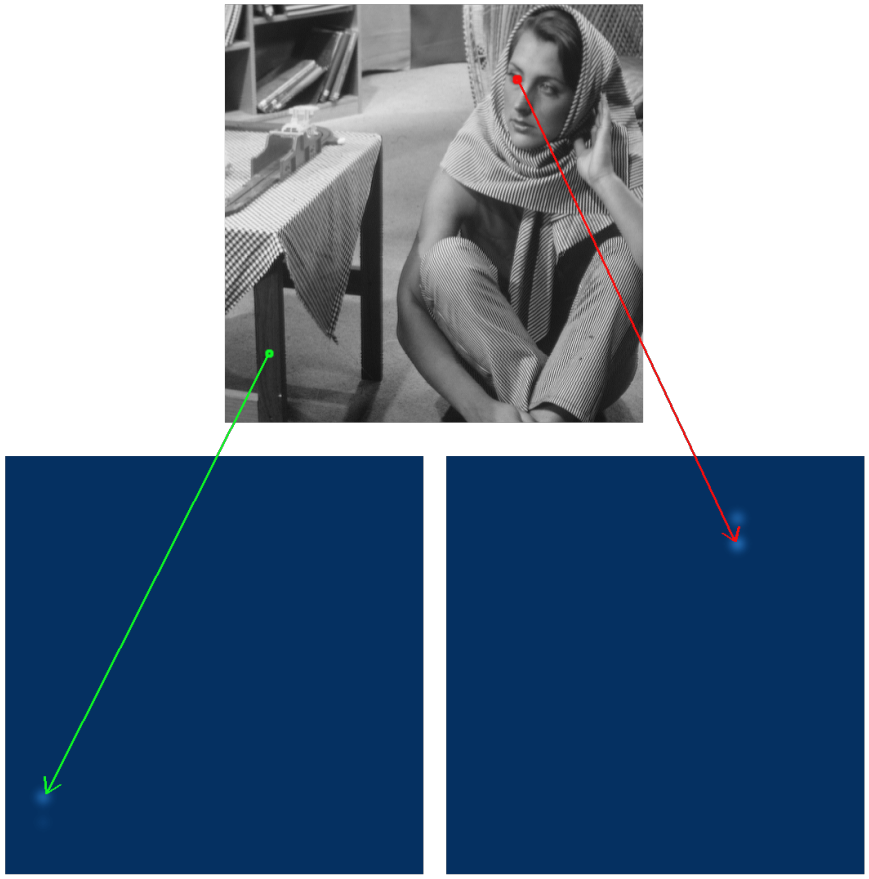

Figure 2.3: Affinity matrices with $h_x = 10$

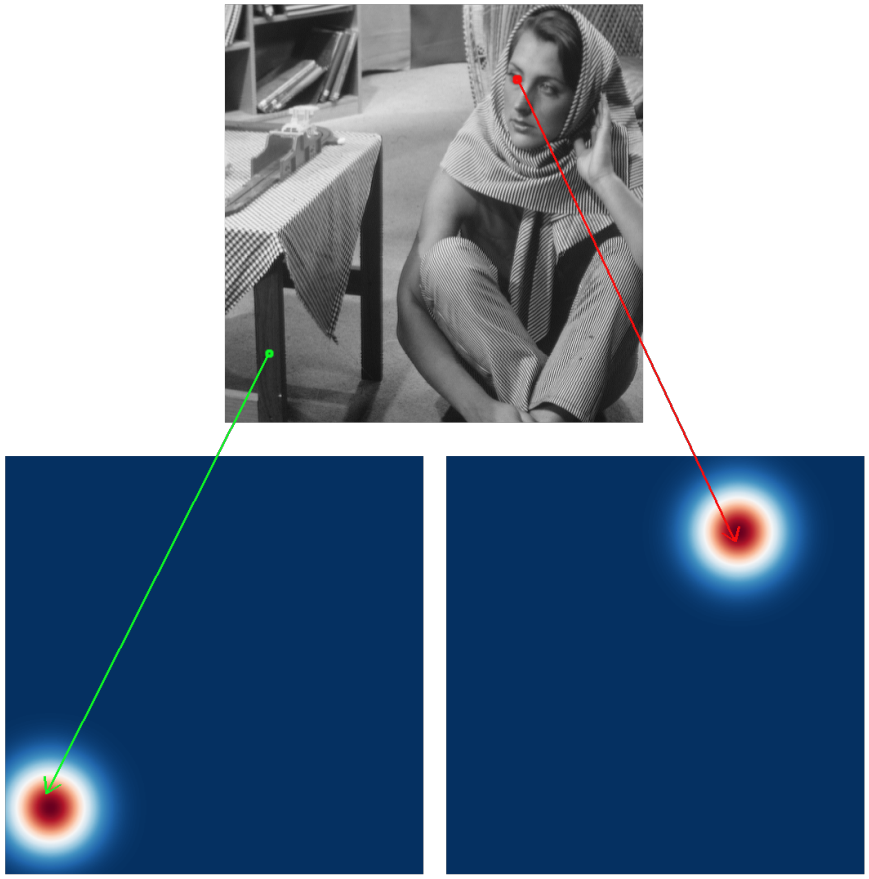

Figure 2.4: Affinity matrices with $h_x = 50$

As we can see, 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,

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

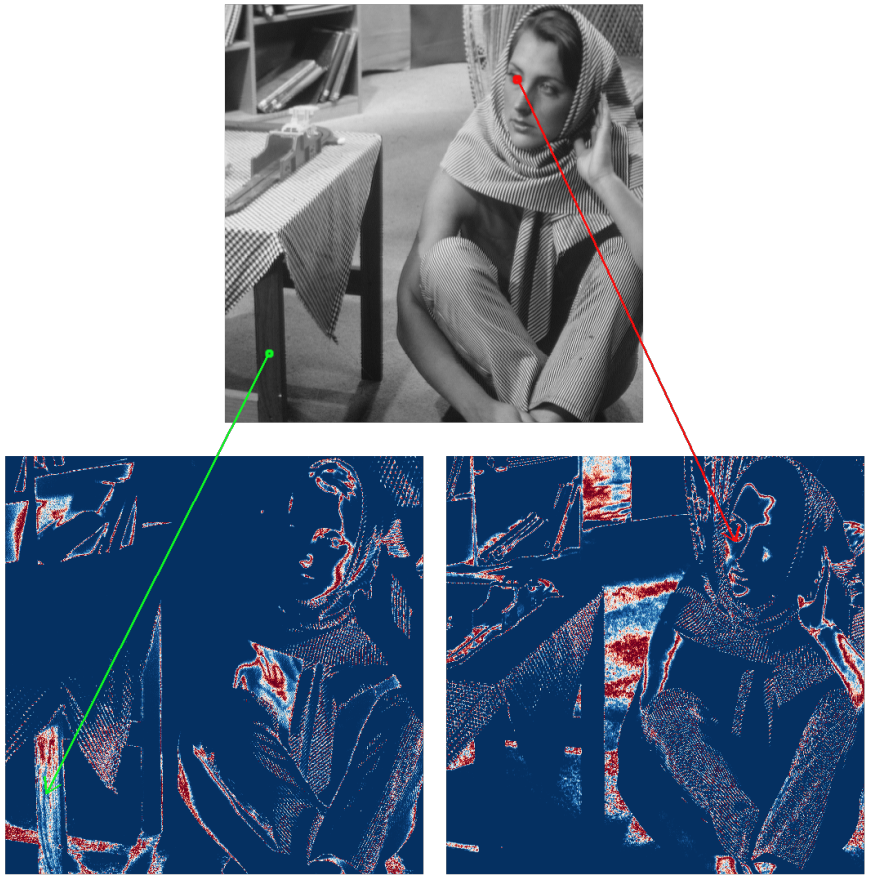

Figure 2.5: Affinity matrices with $h_z = 10$

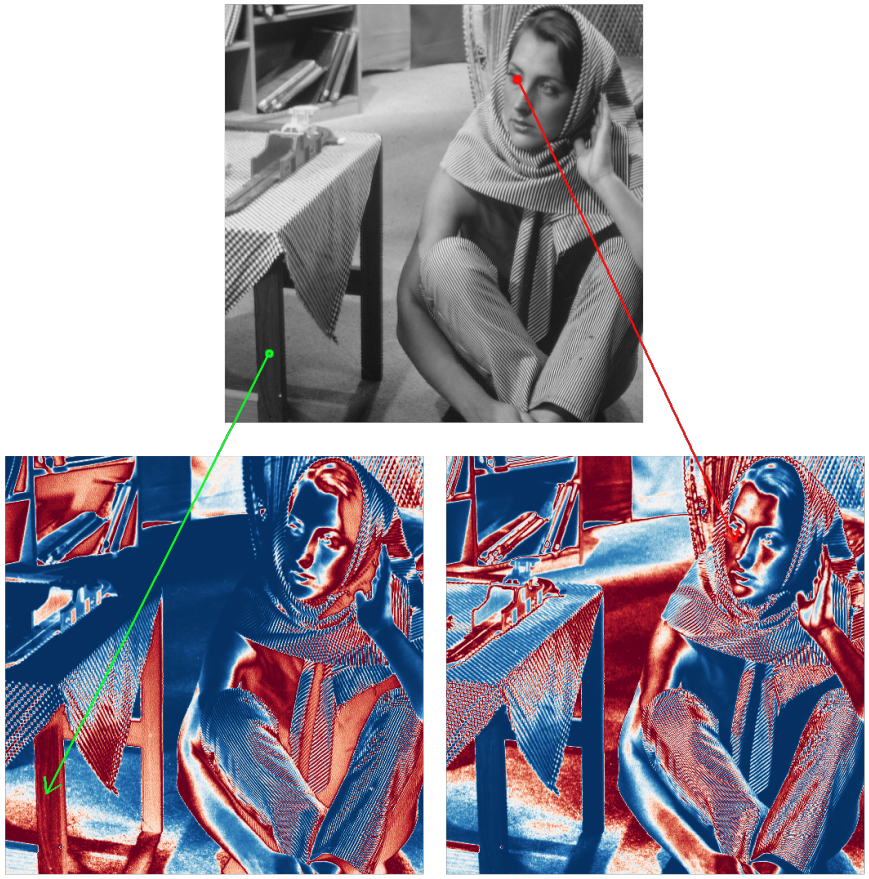

Figure 2.6: Affinity matrices with $h_z = 50$

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] [11] [24]:

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

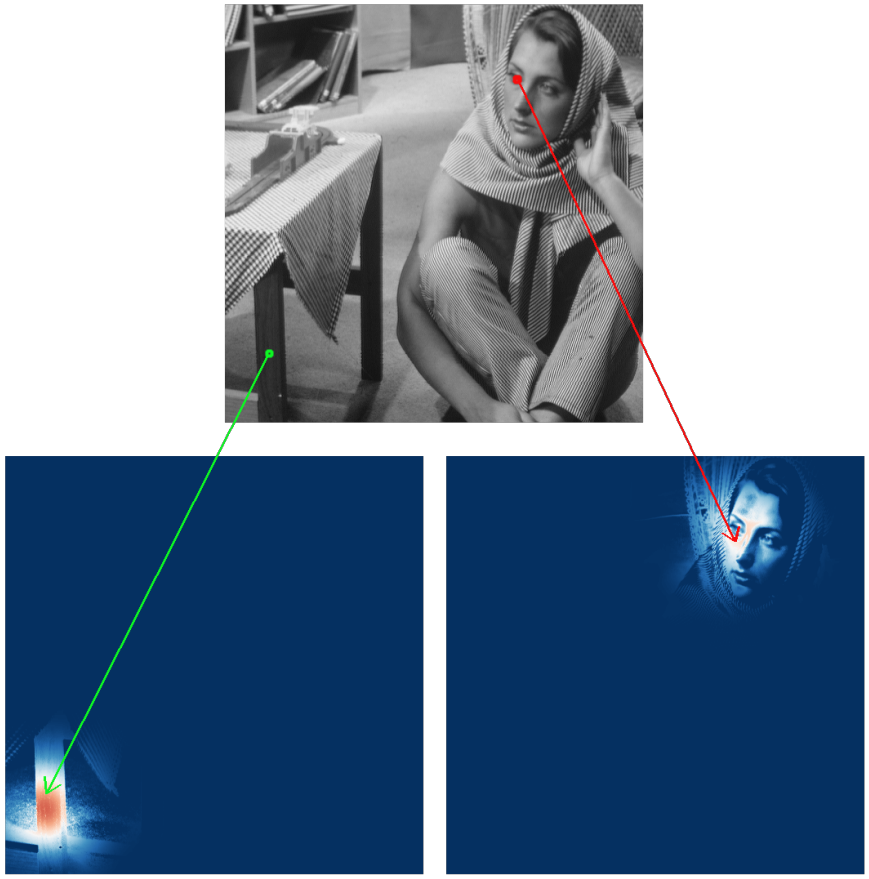

Figure 2.7: Affinity matrices with $h_x = 50$ and $h_z = 35$

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 [11] [25].

Locally Adaptive Regression Kernel (LARK) uses the geodesic distance based on estimated gradients [4] [26].

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, 1]

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

2.3 Discussions and remarks

Pixel degree An interesting matrix that can be observed is the degree of the pixels. A pixel, in our case, is a node of the graph, so we sum the weights of outgoing edges from each node. On the picture below, red pixels indicate that the pixel has a lot of similarity with all other pixels in the image, which can be roughly interpreted as the pixel is a common one:

(a) Original image (b) Pixel degrees

We observe that the most uncommon pixels are very light ones and very dark ones in this image.

Implementation For our implementations, we choose to focus on grayscale images. For the sake of ease, we sample the picture using the spatially uniform techniques which yields good results for natural images since they contain redundancy.

We want to use the re-normalised Laplacian definition to build a smoothing/denoising data-dependent filter. The Laplacian is built such as

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

With the Laplacian operator formulation as $\mathcal{L} = \alpha(D-K)$, so $I-W = \alpha(D-K)$ and finally

$$W = I + \alpha(K - D)$$

as stated in [5].

Results and performances TODO some results + performances

Chapter 3

Dense Linear Systems and High Performance Computing

3.1 Theory

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* [24] and the *Non-local Mean filter* [25] 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 [27]. 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 $K_{ij} \geq 0$.

In our case, we shall use the re-normalised Laplacian [1], which will result in a normalisation-free filter. [1] and [5] define this Laplacian operator as

$$\mathcal{L} = \alpha(D - K),$$

with $alpha = \mathcal{O}(\bar{d}^{-1})$, $\bar{d} = \sum_{i=1}^{N} \frac{d_i}{N}$, $d_i = \sum_{j=1}^{N} K_{ij}$ and $D = diag(d_i)$. From the definition given by [1], $\mathcal{L}$ is symmetric positive definite (SPD) and its eigenvalues $0 \leq \mu_i \leq 1$.

The filter is implicitely defined by [18] as $W = I - \mathcal{L}$ and W is SPD as we consider a SPD Laplacian. We know from [11] that the eigenvalues of W are defined as $0 \leq \lambda_i^W \leq 1$ and the largest eigenvalue $\lambda_1^W = 1$.

The image processing algorithm contains the computation of the eigendecomposition of the submatrix W_A. From the properties of SPD matrices, since W_A

is a principal submatrix of W, is it also SPD. Furthermore, we can say that the eigenvalues $0 \leq \lambda_i^{W_A} \leq 1$ and $\lambda_1^{W_A} \leq 1$.

Proof 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

$$\begin{aligned} W_A x = \lambda x &\Leftrightarrow (I - \mathcal{L}_A)x = \lambda x \\ &\Leftrightarrow x - \mathcal{L}_A\, x = \lambda x \\ &\Leftrightarrow \mathcal{L}_A\, x = x - \lambda x \\ &\Leftrightarrow \mathcal{L}_A\, x = (1 - \lambda)x \end{aligned} \tag{3.1}$$

So the eigenvalues of the Laplacian submatrix $\mu = 1 - \lambda$. We know that $\mu \geq 0$, so $1 - \lambda \geq 0$.

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 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.

3.2 Motivation and tools

In the case of our image processing algorithm, choosing a sufficient number of sample pixels is essential for a good approximation of the matrices eigenvalues and eigenvectors. As exposed in the litterature [11] [19], less than 1% of the pixels seems to be enough to capture most of the image information. However, applied to very high resolution images, 1% of the number of pixels is still a large amount and causes to compute large dense matrices. Additionally, we intend to apply this algorithm on 3D images where the problem size grows even further.

As exposed previously, the algorithm contains matrix computations and eigenvalue problems. The computations are mostly independent (row independent for most matrix computations). Therefore, there is an opportunity and a need

[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.

to speed up the algorithm by parallalising it on supercomputers. For this, we use the PETSc library (Portable, Extensible Toolkit for Scientific Computation) [28], which makes use of HPC tools like the MPI standard to distribute and compute efficiently matrices and vectors. Furthermore, the library SLEPc (Scalable Library for Eigenvalue Problem Computations) [29], based on PETSc, is used to solve the eigenvalue problems efficiently.

3.3 Experimental results

Architecture precisions

Algorithm precision

Observed runtimes

Image results

Chapter 4

Conclusion

4.1 Discussions

4.2 Perspectives

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] Jeff Cheeger. "A lower bound for the smallest eigenvalue of the Laplacian". In: *Proceedings of the Princeton conference in honor of Professor S. Bochner*. 1969, pp. 195–199.

[8] Drago M. Cvetkovi, Michael Doob, and Horst Sachs. *Spectra of graphs: theory and application*. en. Google-Books-ID: 4u7uAAAAMAAJ. Academic Press, 1980. ISBN: 978-0-12-195150-4.

[9] 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.

[10] 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.

[11] 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).

[12] 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).

[13] 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).

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

[15] 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.

[16] 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).

[17] 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.

[18] 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).

[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] 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).

[21] 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).

[22] 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.

[23] 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).

[24] 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.

[25] 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.

[26] 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.

[27] 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).

[28] Satish Balay et al. *PETSc Web page*. http://www.mcs.anl.gov/petsc. 2017. URL: http://www.mcs.anl.gov/petsc.

[29] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems". In: *ACM Trans. Math. Software* 31.3 (2005), pp. 351–362.