Spectral Graph Theory and High Performance Computing
Master Thesis

David Wobrock david.wobrock@gmail.com

January 18, 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 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 image processing algorithms shown by Milanfar's work on 2D images at Google. We want to observe the performance and results of these algorithms to 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 large in the 3D case. That is why we want to apply the algorithm on high performance architectures in order to achieve reasonable computation times. These will apply preconditioning our systems of linear equations with domain decomposition methods and use Krylov methods for solving.

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 2D image processing using the graph Laplacian operator, focusing three applications: denoising, sharpening and image segmentation. For each of them, we show a classical algorithm, a newer spectral approach and then a short comparison of them. Next, we present what has been done in 3D image processing so far using spectral graph theory, essentially using the graph Laplacian operator again. Finally, we touch a few words on domain decomposition 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 with 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], 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 [5]. Let M 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.

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 [6] 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

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

with $E(G)$ the edges 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 [7], 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 2D 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 has to hold the Shannon-Nyquist theorem [8]. 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 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 [8] 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 [9], 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 [10]. 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 must sample the image and compute an approximated $\tilde{K}$ on this subset using the Nyström extension. Because K is symmetric, it can in fact be approximated through its eigenvalues and eigenvectors with

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

Knowing that the eigenvalues of K decay very fast [1], the first eigenvectors of the subset of pixels are enough to compute the approximated $\tilde{K}$.

Generally, as proposed in [10] and [11], 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 [11] 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 [12] 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.

Sharpening

Background Sharpening consists basically in a high-pass filter which will magnify high frequency details [12]. The two main issues 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 artefacts.

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 [13]. 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 in [14] 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 [12]. 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 [14] 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 artefacts. 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 [14] 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.

Image Segmentation

Background To recognise patterns, find groups and cut images are a similar problems 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 [4]

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 [15] [16] [17]. But the results can only be qualified as proposals for recognised regions, such does [16] and would need further optimisation to be the final groups. For instance, [15] introduced a cut criterion using *minimum cut* on a graph but it was observed that it favors finding small components and [17] tries to optimise this fact.

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

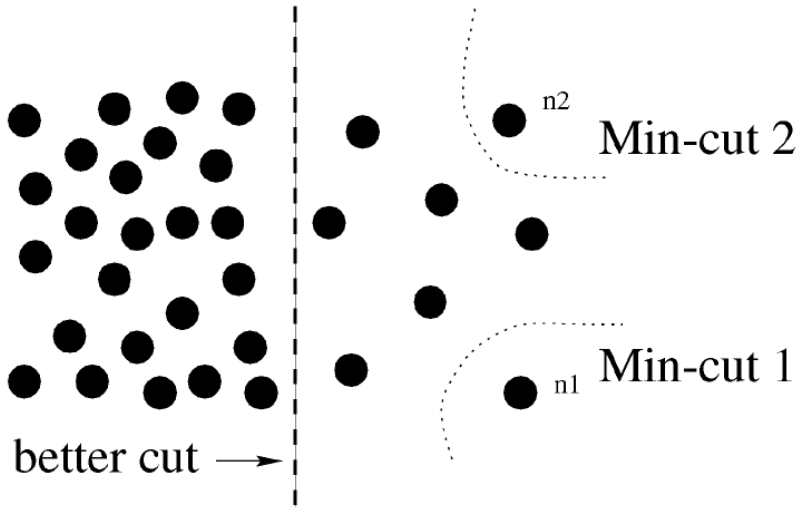


Figure 1.1: Bad partitioning from *minimum cut* in a particular case [18]

Normalised cut To improve this state [18] 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 [19] and [18] that minimising the $Ncut$ exactly is a NP-complete problem. However, [18] proposes a method to approximate 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 [18]. The second-smallest eigenvector is used to bipartition the graph by finding the best splitting point to minimise $Ncut$, and we see that this approach links to spectral graph theory. 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 [20], edge separators of a graph are created by iteratively reweighting edges until the graph eventually disconnects into groups. At each iteration, only a small number of eigenvectors are computed to reweight the edges. This is called spectral rounding by [20] where the detailed algorithm is exposed.

The general idea is an iterative algorithm that reweights edges so it produces a k-way partition. At the beginning, we define the graph $G = (V, E, w^0)$ with its initial weights w^0 which can be defined as wanted. [20] uses the *Intervening Contour* such as described in [18] to define the initial weights. A step of the algorithm is called *SR-Step* and computes $\{w^{(N)}\}$ of N weightings such that the graph $G^N = (V, E, w^N)$ is disconnected into k components. These weights are computed according to a valid reweighting scheme, using the eigendecomposition of the graph. We can observe that partitioning the graph in this manner actually plays on the Cheeger constant and its bounds because we are directly working on the "bottlenecks" of the graph. The reweighting function is based on formal

fractions, defined by a pair of real numbers $\frac{a}{b}$ and its value as a real number a/b. We can define the fractional average as $\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$ which serves as the basis for defining the inverse fractional reweighting, used as reweighting scheme. [20] proves that the so-called *SR-Algorithm* converges into a k-way partition by showing, among others, that the *SR-Step* also converges in a finite number of iterations.

Comparison This spectral rounding method produces directly discrete solutions, and seems to compare nicely against the $Ncut$ approximation by producing more human-like results [21]. [20] claims that spectral rounding finds better cuts on average than $Ncut$ based on an empirical evaluation and comparison of the partition entropy. Anyway, it has been generally accepted that the $Ncut$ produces better cuts than *minimum cut* for graph partitioning of natural images.

1.3.3 3D image processing

In general, [5] sets a very good overview of most variants and applications of spectral processing. A section is dedicated to mesh Laplacian operators, which are the most commonly used operators for spectral mesh processing, and it summarises that they can be divided into two groups. First, as studied in [3], *combinatorial mesh Laplacians* which are solely based on the mesh topological information, but still have eigenfunctions that show a remarkable conformity to the mesh geometry. Second, *geometric mesh Laplacians* encode explicitly the geometric information by a discretisation of the Laplace-Beltrami operator from Riemannian geometry.

Even though both of these categories are noticeably distinct, they can be expressed in a single mathematical definition. We find again the comparison with the center-surround average when we define that a mesh Laplacian operator takes the difference of the value of a function at a vertex and a weighted average of its neighbour vertices values.

One of the simplest Laplacian definitions is the *first order Laplacian*, where a given vertex only involves the 1-ring neighbours. A triangle mesh with n vertices is represented as $M = (G, P)$ where the graph $G = (V, E)$ and $P \in \mathbb{R}^{n\times 3}$ represents the geometry of the mesh. For each vertex $i \in V$, there is an associated position vector denoted $p_i = [x_i, y_i, z_i]$. Finally, the set of 1-ring neighbours of i is formulated as $N(i) = \{j \in V | (i, j) \in E\}$.

1.3.4 Domain decomposition methods

Domain decomposition methods will also be an important topic of this degree project. Those methods are used to solve problems of linear algebra involing partial differential equations (PDEs). Functional analysis is used to study PDEs, which is necessary for a numerical approximation. 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 [22] 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 focus on Schwarz methods for preconditioning, and mostly iterative Krylov methods, such as the conjugate gradient and GMRES. The algorithms will be able to find the solution to a Poisson problem $\Delta\phi = f$, where Δ the Laplacian operator and f a function that is often given. The goal is to determine ϕ with f and knowing some boundary value conditions. These types of problems are well-posed: a solution exists and is unique.

Schwarz algorithms are iterative methods of solving subproblems alternatively in domains Ω_1 and Ω_2. Also important is the use of domain decomposition methods as preconditioners for Krylov methods, which are iterative solvers for linear systems. We shall also look into direct solvers, for instance the LU factorisation, to understand their limitation on large problems.

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. 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 use the latter to build the filter W containing the weights.

According to [23], 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 positive definite diagonal matrix with the normalising factors along its diagonal such as $\forall i, D = diag\{\sum_j K_{ij}\}$. D corresponds in fact to the degrees of each node in graph theory terminology. Other definitions of the Laplacian are shown later in 2.3.3.

Interestingly, we can define from [10] that, by approximating W by a symmetric, positive definite, doubly-stochastic matrix, this new $\hat{W}$ can be computed thanks to its eigendecomposition

$$\hat{W} = VSV^T,$$

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

Our global filter can be expressed as

$$\hat{z} = \hat{W}y = VSV^T y.$$

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 eigendecomposition of K_A.

As stated in [10], 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

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

One might wonder how an approximation of the first elements of the eigendecomposition can be enough to approximate the whole matrix. In fact, the eigenvalues decay quickly as shown in [1] and [24]. This means that the whole matrix is mainly defined by the first eigenelements.

Orthogonalisation 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 [25] 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 $Q = \Phi_Q \Pi_Q \Phi_Q^T$. As it is proven in the appendix of [25], $\hat{W} = V \Pi_Q V^T$ with

$$V = \begin{bmatrix} W_A \\ W_B^T \end{bmatrix} W_A^{-\frac{1}{2}} \Phi_Q \Pi_Q^{-\frac{1}{2}}.$$

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

Summary The figure below summarises the described processing chain.

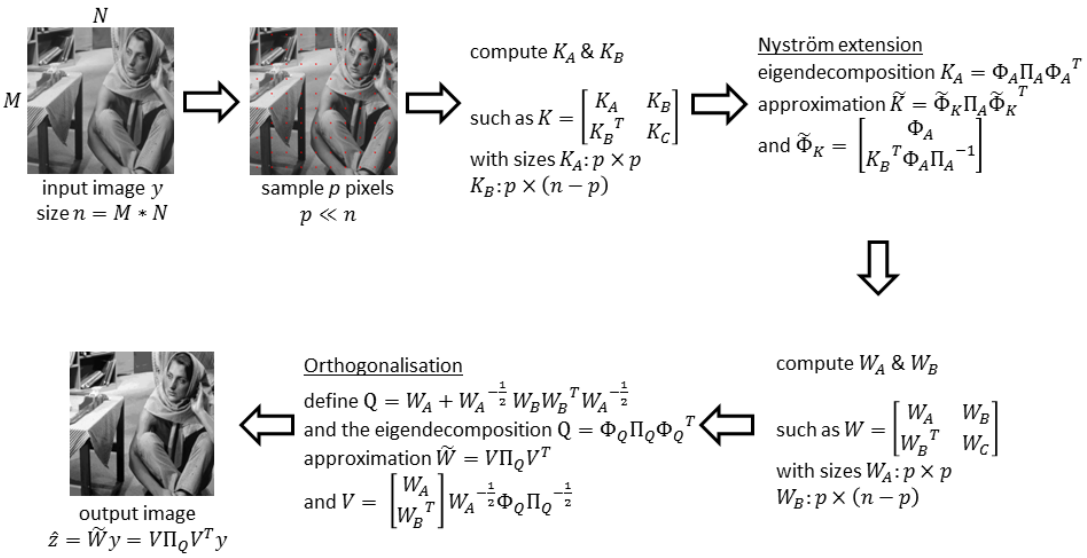


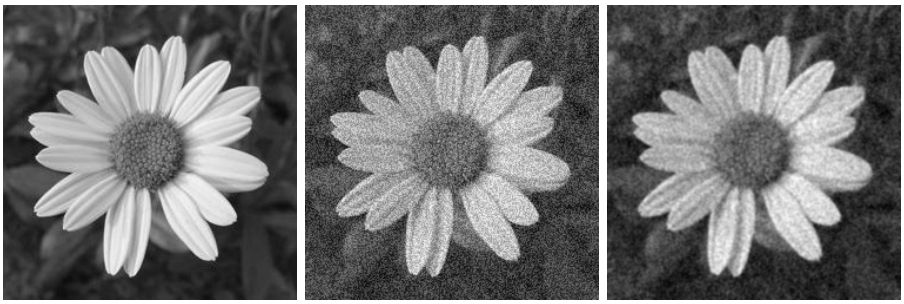
Figure 2.1: The 2D image processing chain

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 [8] as kernel method to compute the pixels similarity. The Laplacian matrix is defined through the "Sinkhorn" iterative algorithm [4], where the resulting approximated eigenvectors require to be orthogonalised, which can be done in one step, as discussed in [25]. We finally apply our approximated filter to the noisy image.

The implementation is generally inspired by the MatLab implementation of [10].

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



(a) Original image (b) Noisy image ($\sigma = 100$) (c) 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. The noise as been added artificially using a normal distribution of $\sigma = 100$.

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 application of 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 [26].

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 [27] [28].

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 [10].

Figure 2.3: 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 [26].

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 [27].

2.3.2 Affinity function

The kernel function 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 512x512 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. Generating the affinity matrix takes approximately 120 seconds on an Intel i3 processor.

Spatial Gaussian Kernel takes only into account the spatial distance between two pixels [1]. The formula of this kernel is

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

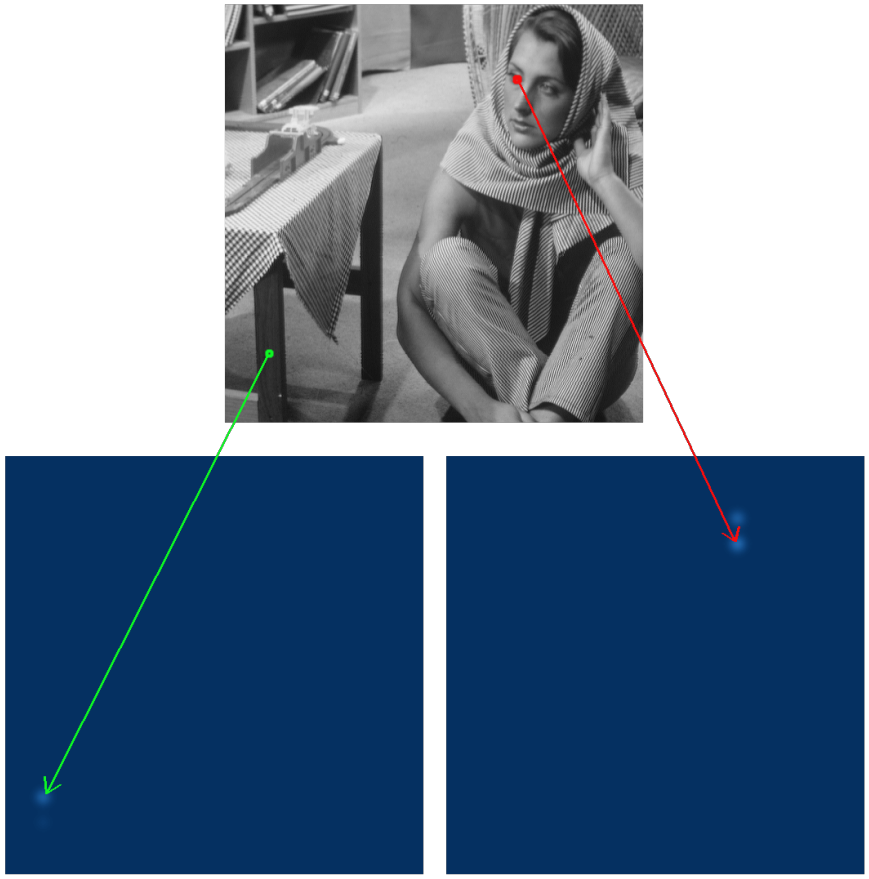


Figure 2.4: Affinity matrices with $h_x = 10$

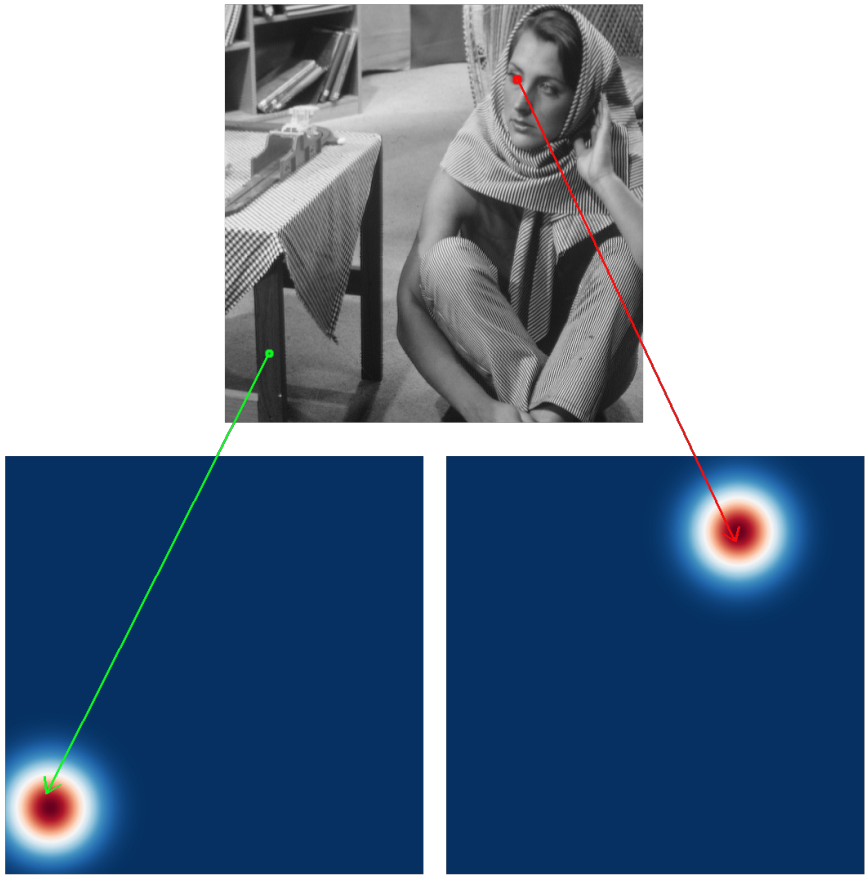


Figure 2.5: Affinity matrices with $h_x = 50$

As we can see, the parameter is influencing on the normalisation of the values and gaussian standard deviation. The bigger 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

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

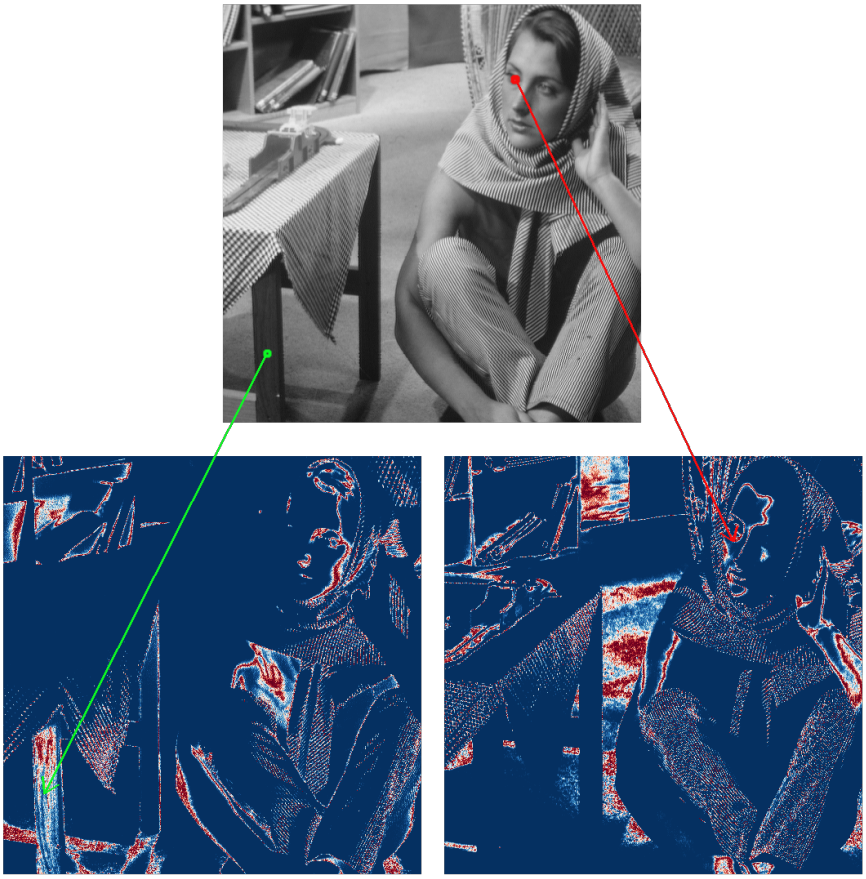


Figure 2.6: Affinity matrices with $h_z = 10$

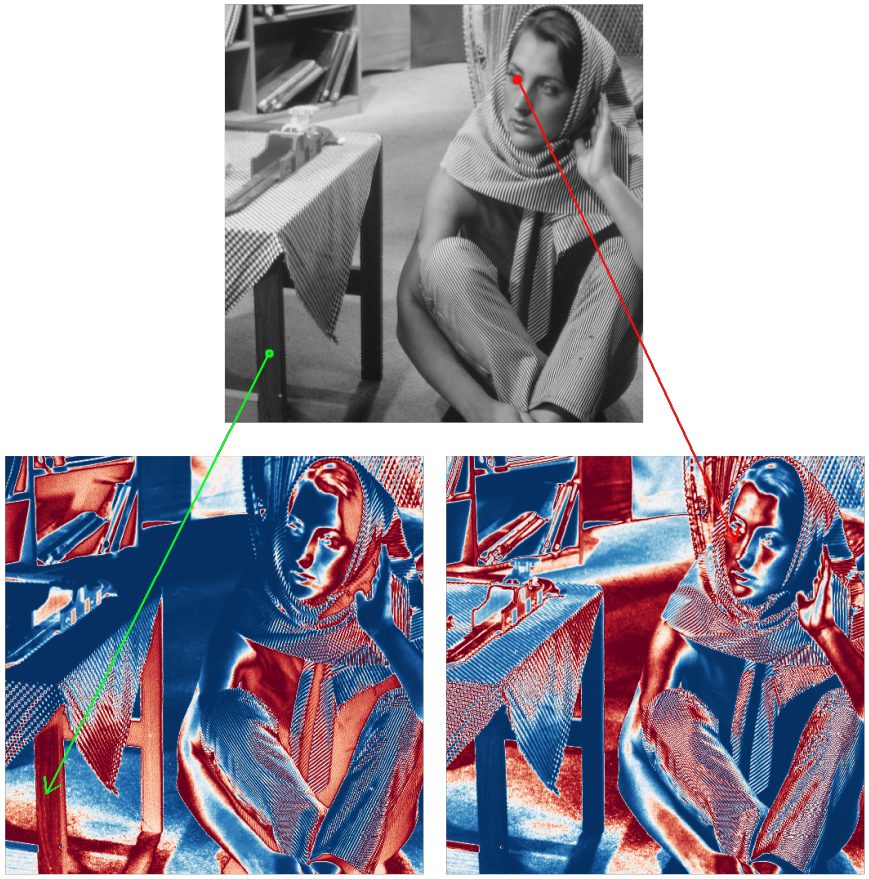


Figure 2.7: Affinity matrices with $h_z = 50$

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] [10].

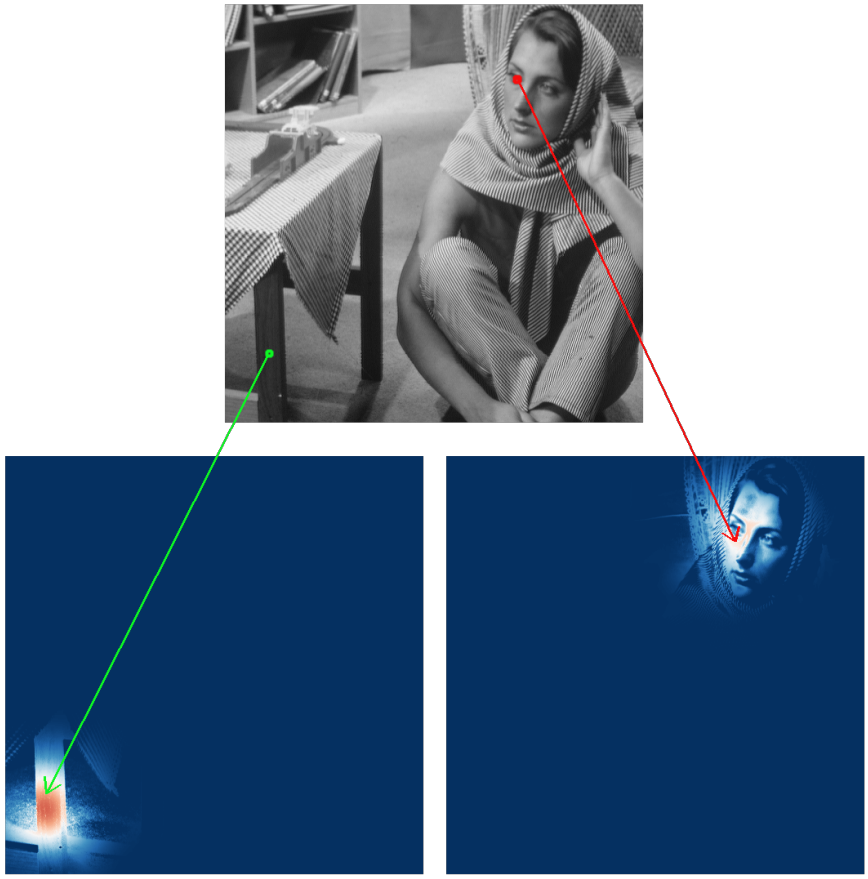


Figure 2.8: Affinity matrices with $h_x = 50$ and $h_z = 35$

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.

Non-Local Means (NLM) is similar to the bilateral kernel, a data-dependent filter, except that the photometric affinity is captured patch-wise [10].

Locally Adaptive Regression Kernel (LARK) uses the geodesic distance based on estimated gradients [4] [29].

2.3.3 Graph Laplacians

The 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 eigendecomposition 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. And 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	DC eigenvector	Spectral Range
Un-normalised	$D - K$	Yes	Yes	[0, n]
Normalised	$I - D^{-1/2}KD^{-1/2}$	Yes	No	[0, 2]
Random Walk	$I - D^{-1}K$	No	Yes	[0, 1]
"Sinkhorn" [4]	$I - C^{-1/2}KC^{-1/2}$	Yes	Yes	[0, 1]
Re-normalised [30]	$\alpha(D - K)$, $\alpha = \mathcal{O}(n^{-1})$	Yes	Yes	[0, 1]

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

2.4 Discussions and remarks

For our implementations, we choose to focus first on grayscale images. We sample the picture using the spatially uniform techniques which yields good results for natural images since they contain redundancy. The affinity function the photometric gaussian kernel with the parameter $h = 10$.

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

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

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

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

This is supported by [30]. However, as final step, we want to approximate W by $\hat{W}$ through its orthonormal eigenvectors and eigenvalues. For this step, W has to be both symmetric and positive definite. The latter condition is not satisfied with this Laplacian definition. Indeed, W can contain negative values and the eigenvalues are both positive and negative. So W is an indefinite matrix. We will stick with for now stick with the "Sinkhorn" [4] definition of the graph Laplacian.

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.

Chapter 3

Linear Systems and Domain Decomposition Methods

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* [31] and the *Non-local Mean filter* [32]. In general, the kernel functions create a symmetric positive semi-definite (PSD) matrix K with $k_{ij} \geq 0$. For these specific functions, we can even define $0 \leq k_{ij} \leq 1$.

We shall use the re-normalised [1] Laplacian, which will result in a normalisation-free filter [30]. We define the Laplacian operator as

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

with $alpha = \mathcal{O}(\bar{d}^{-1})$ and $\bar{d} = mean(d_j)$.

For this definition, we know that $\mathcal{L}$ is symmetric and its eigenvalues $0 \leq \mu_i \leq 1$, meaning that $\mathcal{L}$ is symmetric positive definite (SPD).

The filter is defined as $W = I - \mathcal{L}$. The identity I is obviously SPD, so the filter is also SPD. We know from [10] 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 is computing 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, λ_{max}^A be the largest eigenvalue of A and u the associated eigenvector.

$$\lambda_{max}^A = max(\frac{A(u,u)}{(u,u)}).$$

Let R be the restriction operator, such as $Ru = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ 0 \end{pmatrix}$ for example.

So $\lambda_{max}^{RAR^T} \leq \lambda_{max}^{A}$ and we can say that $\lambda_{max}^{RAR^T} = max(\frac{A(R^T v, R^T v)}{(v,v)})$.
So $(v, v) = (u, u)$. (??)

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

$$W_A x = \lambda x \Leftrightarrow (I - \mathcal{L}_A)x = \lambda x$$

3.2 Theoretical basis

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.

3.2.1 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 [33]. The LU factorisation, closely related to Gaussian elimination, is hard to parallelise. A block version of the LU factorisation exists that can be parallelised. Other direct solvers, like MUMPS, exist and can be used, but generally they reach a computational limit above 10^6 degrees of freedom in a 2D problem, and 10^5 in 3D.

3.2.2 Iterative solvers

For large problems, iterative methods must be used to achieve a reasonable running time. 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. The most relevant iterative methods are the conjugate gradient and GMRES [34].

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

3.2.3 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.

3.3 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 [10] [25], 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 to speed up the algorithm by parallalising it on supercomputers. For this, we use the PETSc library (Portable, Extensible Toolkit for Scientific Computation) [35], 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) [36], based on PETSc, is used to solve the eigenvalue problems efficiently.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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).

[11] 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).

[12] 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).

[13] 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.

[14] 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.

[15] 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.

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

[17] 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).

[18] 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).

[19] 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).

[20] 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.

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

[22] 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.

[23] 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).

[24] 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).

[25] 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).

[26] 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).

[27] 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.

[28] 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).

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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).

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

[36] 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.