Master Thesis

–

Spectral Graph Theory and High Performance Computing

David Wobrock
david.wobrock@gmail.com

January 30, 2018

Abstract

TODO

Contents

Chapter 1

Introduction

1.1 Background

The talk by Milanfar [1], working at Google Research, about using the graph Laplacian operator for image processing purposes awakes curiosity.

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

1.2 Objective

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

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

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

1.3 Related work

1.3.1 Spectral graph theory

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

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

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

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

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

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

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

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

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

1.3.2 Linear solvers and domain decomposition methods

Solving a system of linear equations such that

$$Ax = b,$$

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

Direct solvers The most commonly used solvers for systems of linear equations are direct solvers. They provide robust methods and optimal solutions to the problem. However, they can be hard to parallelise and have difficulties with large input. The most famous is the backslash operator from MATLAB which performs tests to determine which special case algorithm to use, but ultimately falls back on a LU factorisation [7]. 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 [8], 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 [9].

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 [10] 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

Chapter 2

Image Processing using the Graph Laplacian Operator

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

2.1 Algorithm

2.1.1 Overview

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

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

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

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

$$z = Wy.$$

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

Image as graph Let's think of an image as a graph. Each pixel is a node and has edges to some other nodes. The simplest way to connect pixels to other pixels is their direct neighbours, in which case each node has four edges.

To avoid losing any information, we will instead consider the opposite case, a complete graph, each node connects to all other nodes.

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

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

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

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

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

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

Approximation by sampling and Nyström extension To avoid storing a huge matrix, approximation will be necessary.

Approximating the filter will start by sampling the image and only select a subset of p pixels, with $p \ll N$. The rows and columns of the filter W are re-organised in order have $\begin{bmatrix} W_A & W_B \\ W_B^T & W_C \end{bmatrix}$.

Smallest eigenvalues Knowing the spectral range of the Laplacian, it will be equivalent to

[1] Similarity is also called affinity.

Algorithm 1 Image processing using Graph Laplacian

Input: y an image of size $n \times m$
Output: z the output image
$N \leftarrow n * m$
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 $K \leftarrow \begin{bmatrix} K_A & K_B \\ K_B^T & K_C \end{bmatrix}$
Compute the Laplacian $\mathcal{L}_A$
Compute the smallest eigenvalues Π_A of $\mathcal{L}_A$ and the associated eigenvectors Φ_A
Nyström extension $\tilde{\Phi} \leftarrow \begin{bmatrix} \Phi_A \\ K_B^T \Phi_A \Pi_A^{-1} \end{bmatrix}$
$\tilde{W} \leftarrow \tilde{V} S \tilde{V}^T$
$z \leftarrow \tilde{W} y$

Overview

2.1.2 Variations

According to [11], 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 **??**.

Interestingly, we can define from [12] 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 **??**.

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 **??**. Once computed, we can approximate the eigenvectors Φ and eigenvalues Π of K thanks to the eigendecomposition of K_A.

As stated in [12], 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 [13]

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

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 [14]. 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 [13] 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 [13], 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 $\forall j, i \neq j, \tilde{V}_i \tilde{V}_j = 0$.

Results TODO some results + performances

Chapter 3

Conclusion

3.1 Discussions

3.2 Perspectives

Image processing On the image processing side, [15] proposes an enhancement. [15] argues that the eigendecomposition remains computationaly too expense and shows results of an improvement. The presented results and performances seem astonishing, however, the method is hardly described and replicable with difficulty. This is understandable since this algorithm seems to be in the latest Pixel 2 smartphone by Google and they want to preserve their advantage in the field of image processing.

Linear solvers

Bibliography

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

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

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

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

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

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

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

[8] 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.

[9] 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).

[10] 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.

[11] 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).

[12] 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).

[13] 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).

[14] 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).

[15] 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).