The Numerical Tours of Signal Processing - Advanced Computational Signal and Image Processing

Gabriel Peyré

The Numerical Tours of Signal Processing

Advanced Computational Signal and Image Processing

Gabriel Peyré

Abstract

The *Numerical Tours of Signal Processing* is an online collection of tutorials to learn advanced computational signal and image processing. These tours allow one to follow a step by step Matlab or Scilab implementation of many important processing algorithms. This implementation is commented and the connexions with the relevant mathematical notions are exposed. These algorithms are applied to various signal, image, movie and 3D mesh datasets. These tours are suitable for practitioners in the field, that can use them to learn about state of the art methods. They are also designed to help undergraduate students to understand recent theoretical or numerical advances in signal and image processing.

Index Terms

Signal processing, image processing, mesh processing, computational education, reproducible research, Matlab, Scilab.

I. Introduction

Modern signal and image processing is about dealing with a huge mass of data that exhibit complicated features. These data should be restored, compressed or analyzed using fast algorithms that can take advantage of mathematical models. Understanding these models and the algorithms requires an interplay between theoretical analysis and numerical methods. This article exposes the Numerical Tours of Signal Processing, that are designed to help the practitioner to master advanced concepts and algorithms.

These tours are not target to the electrical engineering community alone. They should be useful to a larger set of discipline often refereed to as "imaging sciences". This includes for instance medical, seismic and astrophysical imaging, acoustics, computer graphics and vision. All the tours can be applied to arbitrary signals, images or meshes, thus enabling to bench the efficiency of the methods on each field's favorite dataset.

The use of online resource, in conjunction with numerical computing languages such as Matlab[1] or Scilab[2], is especially well suited to signal processing education [1]. The Numerical tours are however more mathematically oriented than most of traditional electrical engineering signal processing courses [2]. We believe that modern numerical processing requires both a strong mathematical background and agility with advanced computing concepts. This is a major departure from classical processings, that are based on simple notions such as filtering and Fourier transform. This also reflects the difficulty to teach recent research advances to graduate students [3].

Numerical experimentation to teach signal processing. Numerical experiment is central to understand the mathematical and algorithmic tools developed in modern numerical processing. An undergraduate

Gabriel Peyré is with the CEREMADE CNRS-Université Paris-Dauphine, 75775 Paris Cedex 16 France, email: gabriel.peyre@ceremade.dauphine.fr

[1] http://www.mathworks.com/

[2] http://www.scilab.org/

course in signal processing should alternate between an exposition of these tools, a theoretical analysis of the achievable performance, and a numerical experimentation on synthetic and real datasets. The Numerical Tours of Signal Processing are especially tailored for this last task. They allow the students to implement and compare different algorithms on its own data. This practice allows to better understand the mathematical concepts, and it also makes the student realize that numerical processing is a science on its own, as advocated by [4]

Reproducible research and teaching. Reproducible research has emerged has a fundamental concept in signal and image processing [5], [6]. We believe that there is a continuum from reproducible teaching to reproducible research, that brings under-graduate and graduate students to understand advanced mathematical concepts that lead to recent state of the art processing methods. The Numerical Tours try to bridge this gap between teaching and research in signal and image processing. They review a few foundational notions such as the Fourier and wavelet transforms, and apply them to problems and methods of increasing complexities.

Computational provenance. The notion of computational provenance is of great importance when dealing with advanced numerical methods that are intended to process large set of data [7]. The Numerical Tours adhere as much as possible to computational provenance by including in generated files various metadata, as described in Section II-B.

II. Presentation of the Numerical Tours

The Numerical Tours are divided into three distinct entities:

- *The Numerical Tour of Signal Processing (NTSP):* are the online web pages, maintained by Gabriel Peyré. They gather only the final HTML[3] files (extension `.html`), and in a private section, the solutions of the exercises. Each NTSP is licensed according to the author policy. The copyright is added at the end of each tour.
- *The Numerical Tours Scripts (NTS):* The Numerial Tours Scripts (NTS) are a collection of scripts, each script being a Matlab file (extension `.m`) intended to generate a single Numerical Tour. This generation is explained in detail in Section II-C. Each NTS is licensed according to the author policy.
- *The Numerical Tour Publishing Pipeline (NTPP):* the NTPP is maintained by Gabriel Peyré, and includes several publishing tools, as well as three helper toolboxes, detailed in Section II-C. The NTPP is licensed as GPL source code, so that the community can contribute and extend the capabilities of the tours.

Figure 1 details the files structure of these three entities, that are stored in three different folders. Note that the folder `www/` storing the NTSP is actually an online folder accessible from the user using a web browser. The following sections describe in details these three components.

A. Numerical Tours of Signal Processing (NTSP)

The Numerical Tours of Signal Processing are a set of online web pages. They can be accessed from

`http://www.ceremade.dauphine.fr/~peyre/numerical-tour/`

Figure 2, left, shows the main homepage of the Numerical Tours.

Each tour in the NTSP is stored in an online directory, that contains a single HTML file (extension `.m`) and a JPEG image file for each figure of the tour (extension `.jpg`).

At time of writing, there are around 100 different Numerical Tours, but this number is rapidly growing. The set of pages are regrouped in approximately 20 topics, such as wavelet processing, computer graphics, audio processing, etc. Each tour is a single, independent, web page, that exposes with a linear progression, a single concept and its use on a signal or image processing problem. Figure 2, right, shows a small extract from a tour.

[3] `http://www.w3.org/html/`

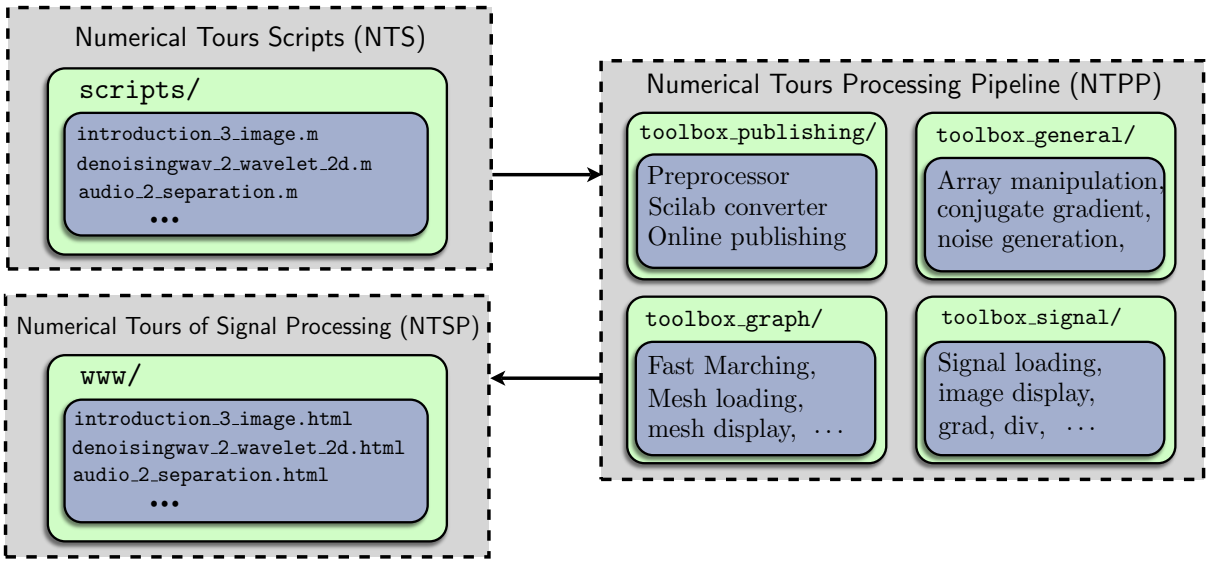


Fig. 1. *Numerical Tours structure.*

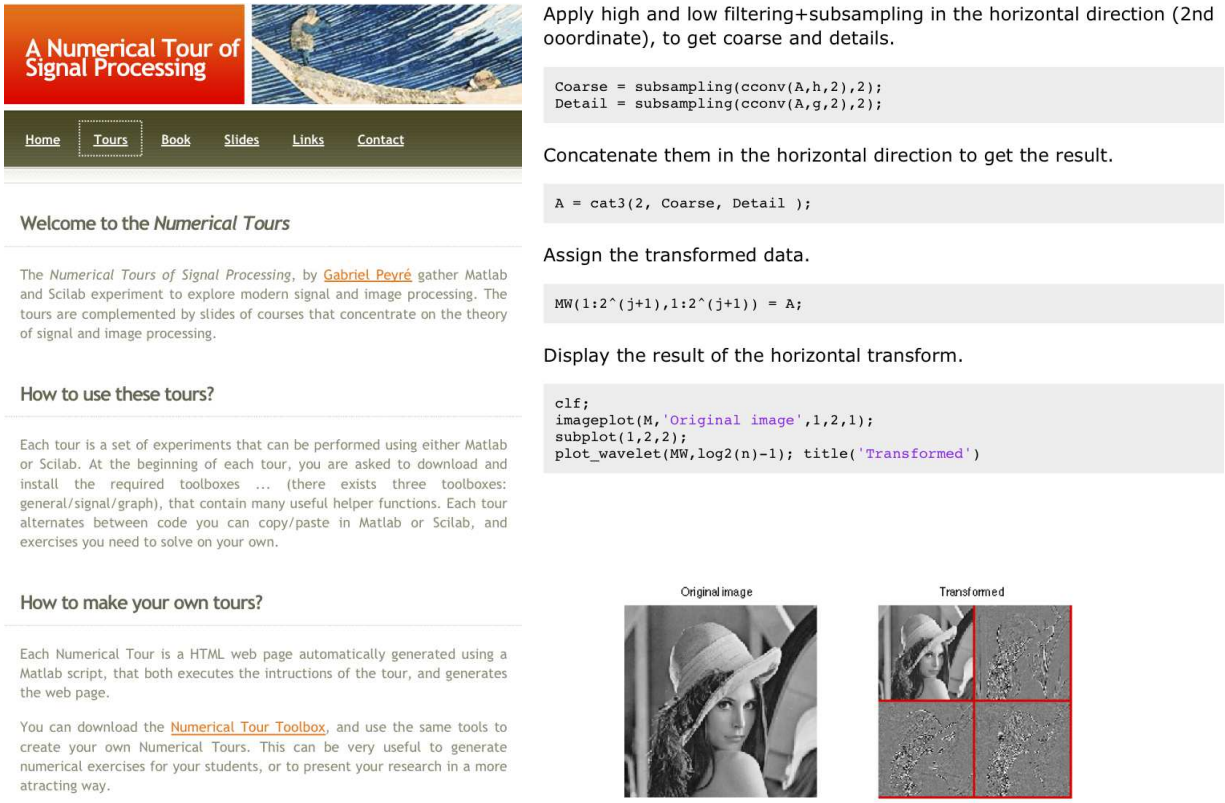


Fig. 2. *Left: numerical tour starting page, right: extract from a numerical tour (left) and sample (right).*

B. Numerical Tour Scripts (NTS)

Each tour in the NTSP is entirely created from a single Matlab file (extension `.m`). This is the only file that needs to be written and maintained by an author contributing to the NTSP. This script contains both the Matlab code to perform the computations, regular Matlab comments, plain text to give details to the reader, references to external web pages and bibliography, and mathematical equations.

Figure 3, left, shows the Matlab code for a simple tour. It contains title, sub-title, some text, an equation, an exercise and a display.

```
%% Sample Numerical Tour
% This tour plots a discretized function.

%% Ploting a function

%%
% One can define the function
%     \[f : x \in [-1,1] \rightarrow
%     x^2 \in \mathbb{R}^+ \]
% in matlab as a discretized vector.

f = linspace(-1,1,256).^2;

% EXO
%% As an exercise, display the function.
clf; h = plot(f); axis tight;
% EXO
```

Sample Numerical Tour

This tour defines and plot a discretized function.

Contents

- Ploting a function

Ploting a function

One can define the function

$$f : x \in [-1,1] \rightarrow x^2 \in \mathbb{R}^+$$

in matlab as a discretized vector.

```
f = linspace(-1,1,256).^2;
```

Exercice 1: (the solution is exo1.m) As an exercise, display the function.

```
exo1;
```

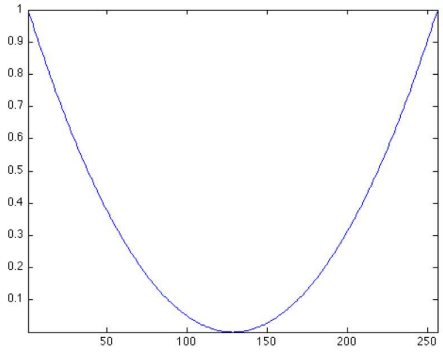


Fig. 3. *Left: matlab code, right: corresponding web page.*

Beside the basic Matlab / Scilab command lines, an author can use the following special markup:

- *Titles, sub-titles and text:* they are indicated by the standard Matlab publishing notations `%%`. Special characters are used to indicated formatting (italic, bold) and add HTML links to external pages.
- *Exercises:* they are encapsulated within a markup `%EXO`. As detailed in the next section, exercises are not shown to the reader, only the output is displayed in the HTML page. The solutions of the exercises is distributed freely to teachers.
- *Unprocessed commands:* they are encapsulated within a markup `%CMT`. This is useful to save figures in high resolution, that are produced by running the Matlab script as usual, but will not be output in the HTML page.
- *Equations:* any valid LaTeX equation can be added, either inline using the command `\( ...\)`, or centered using `\[ ...\]`.
- *Meta-data:* most of the tours are intended to be applicable to a large variety of data. The indication about the parameters of each tour (size and name of the image, amount of compression used, ...) are regrouped within the markup `%META`. This allows for the user to re-run the code with other sets of parameters, as advocated by the computational provenance paradigm [7].

C. Numerical Tour Publishing Pipeline (NTPP)

Each Numerical Tour is generated from a single Matlab scritp from the NTS. This makes the maintenance of the tour simple and fast. One can run the script as a standard Matlab script, which ensures that the code works as expected. One can then process it using the NTPP, as detailed in Figure 4, to produce the final HTML to be part of the NTSP online ressource.

Author file. This is a Matlab file of the NTS, named here `tour.m`, as detailed in Section II-B.

Matlab / Scilab toolboxes. To help the user, the NTPP includes three toolboxes, that can be downloaded from the NTSP web page as `.zip` files. These toolboxes are:

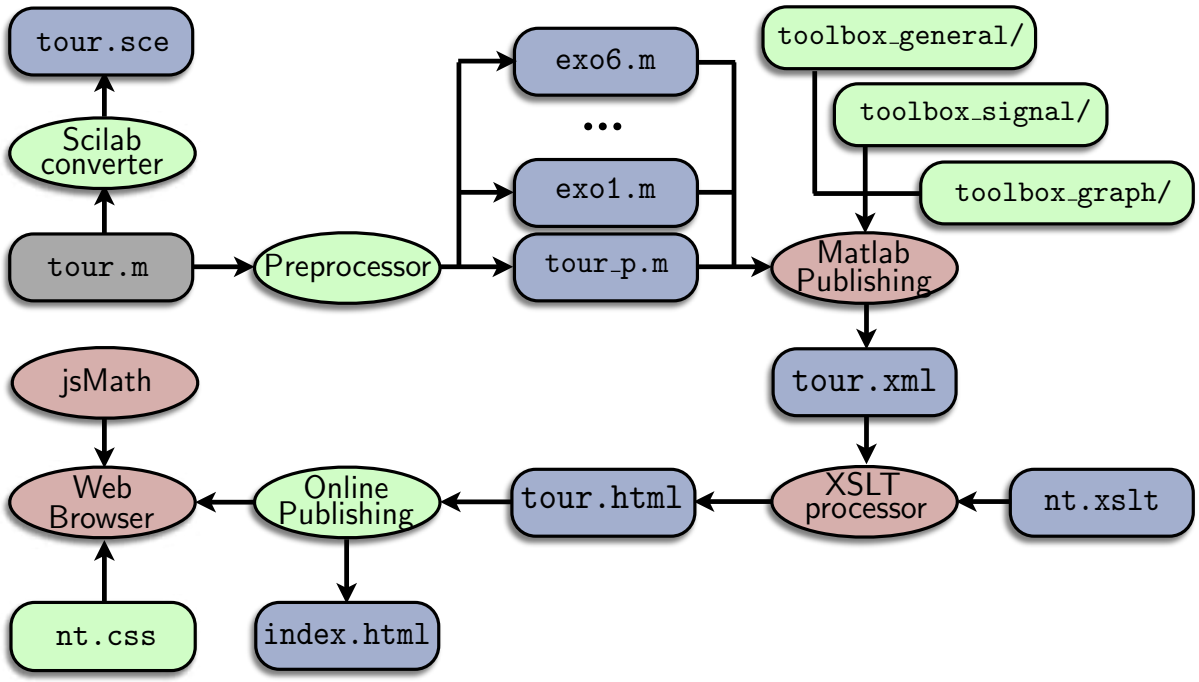


Fig. 4. *Numerical Tours Processing Pipeline (NTPP) diagram.*

- `toolbox_general/`: includes basic helper functions to manipulate arrays. Most of the functions are implemented to ensure a perfect compatibility between Matlab and Scilab versions of the tours.
- `toolbox_signal/`: includes various functions to load, manipulate and display 1-D signals and 2-D images. It also includes a few well known signals and images that are used in the tours, also the user is encouraged to use his own data.
- `toolbox_graph/`: includes various functions to load, manipulate and display planar and surface triangulations. It includes a few samples of 2-D and 3-D meshes. It also includes Fast Marching methods on 2-D, 3-D and triangulated grids for the computation of geodesic distances and minimal paths.

Scilab converter. To ensure portability, a Scilab converter generates a valid Scilab file `tour.sce`. This conversion is simple, and consists in translating comments from Matlab style `%` to scilab style `//`.

The most important part to avoid portability issues is taken care of in the toolboxes. All helper processing functions in the three toolboxes are coded both in Matlab and Scilab. Many basic language functions that are not exactly equivalent between the two languages have been re-coded to ensure a seamless portability.

One can thus run `tour.sce` to ensure that the code is running correctly with Scilab.

Preprocessor. The original script `tour.m` is pre-processed to generate an intermediate Matlab script `tour_p.m`, together with exercises scripts `exo#.m` where # indicates the number of the exercise.

This preprocessing consists in:

- Locating every exercise, that is indicated in the original script using the `%EXO` special markup. Each exercise is removed from the main script and copied in a sub-script `exo#.m`.
- Locating every part that should not be compiled in the numerical tour. Each part is indicated in the original script using the `%CMT` special markup, and is removed from the main script.
- Adding a heading section to the script that details to the user how to download the toolboxes, install them, and give some basic advices before starting a tour.

Matlab publishing. Publishing `.m` files is a built-in Matlab feature, that compiles the script `tour_p.m`

and produces a XML [4] file, named here `tour.xml`. This file encapsulates the original content using several XML markups to indicate wether each part of the script corresponds to a Matlab command, a Matlab comment, a title or a text paragraph. Special comment lines in the script indicated by `%%` are encapsulated into either title, sub-title or plain text, and constitute the body of the tour.

XSTL processor. The XML file `tour.xml` is an intermediate file that is transformed into a final HTML file `tour.html` that is intended to be viewed by the end user. This is achieved using a XSLT[5] stylesheet `nt.xstl` that indicates how to translate each XML markup into a HTML markup. This process translates all XML markups into HTML markups, which allows the NTPP to create a layout style specific to the NTSP. This also allows the NTPP to put additional copyright informations as specified by the author of the tour.

Online publishing. The HTML file of the tour `tour.html`, together with the JPEG images, are copied into the server that hosts the NTSP. The main homepage of the tour `index.html` is also automatically updated to reflect any changes in the tours, such as the addition of a new tour or a modification in a title of a tour.

Final display. The display is controlled using a CSS[6] cascading stylesheet `nt.css`, that indicates to the HTML browser the font color and size, spacings, etc. A Numerical Tour typically includes equations to explain some important mathematical notions. These equations are coded in the text using standard LaTeX notations. They are processed online using the JavaScript mathJax[7] program to display nice looking equations on the screen.

III. Topics Covered by the Tours

We now detail the main mathematical and numerical notions studied by the tours. All the images shown in this section are gathered from the Numerical Tours. They can thus be reproduced by following the step by step description of the method described in each tour.

A. Fourier and Wavelet Processing

A signal or an image is manipulated numerically as a discretized vector $f \in \mathbb{R}^N$ where N is the number of 1-D samples or 2-D pixel samples. Many basic processing tasks are performed using a change of basis, by considering the projections $\{\langle f,\, \psi_m \rangle\}_{m=0}^{N-1}$ where $\mathcal{B} = \{\psi_m\}_m$ is an orthogonal basis of $\mathbb{R}^N$.

Arguably the two most common bases are the Fourier and wavelet systems. The Fourier basis assumes periodic boundary conditions, and is defined in 1-D as $\psi_m(x) = e^{\frac{2i\pi}{N} xm}$, where $0 \leqslant m < N$ indexes the 2-D frequency of the atom and $0 \leqslant x < N$. For images, 2-D Fourier atoms are defined using tensor products $\psi_m(x_1, x_2) = \psi_{m_1}(x_1)\psi_{m_2}(x_2)$ where $m = (m_1, m_2)$ indexes 2-D frequencies. Fourier atoms are useful to compute convolutions that are diagonalized over the Fourier basis. They are thus at the heart of many linear processing tasks, in particular to perform denoising by blurring or to perform Tikhonov regularization to remove some blur in a noisy image. Most importantly, the set of inner products $\{\langle f,\, \psi_m \rangle\}_{m=0}^{N-1}$ is computed in $O(N \log(N))$ operations using the Fast Fourier transform. A windowed Fourier analysis is applied locally to a sound or an image to analyze its local frequency content. This finds applications to sound processing, audio sources separation and texture analysis.

While Fourier basis is useful to deal with smooth periodic signals, it fails to represent compactly discontinuities such as edges in images. A continuous wavelet basis is defined by translating and scaling a mother wavelet function $\psi_{j,n}(x) = \psi(2^{-j}x - n)$. The wavelet function ψ is smooth, oscillating on its compact support, so that a wavelet atom $\psi_{j,n}$ analyzes the signal around a position $2^j n$ on a

[4] `http://www.w3.org/xml/`

[5] `http://www.w3.org/TR/xslt`

[6] `http://www.w3.org/Style/CSS/`

[7] `http://www.mathjax.org/`

segment of size proportional to the scale 2^j. A 2-D wavelet basis is obtained using three mother wavelets $\{\psi^k\}_{k\in\{V,H,D\}}$ to define horizontal, vertical and diagonal atoms that are also scaled and translated $\psi^k_{j,n}(x) = \psi^k(2^{-j}x - n)$ where $2^j(n_1, n_2)$ is the 2-D position of the wavelet. A carful design of the wavelet function ψ allows one to define an orthogonal basis $\{\psi_m\}_{m=(j,n,k)}$ for $j \in \mathbb{Z}, n \in \mathbb{Z}^2, k \in \{V, H, D\}$ of $\mathrm{L}^2(\mathbb{R}^2)$. This basis is defined through a cascade of discrete filters see [8]. This cascade of filters naturally extends to the discrete setting by applying them to a discretized vector $f \in \mathbb{R}^N$. This leads to a discrete orthogonal basis $\{\psi_m\}_{m=(j,n,k)}$ for $0 \leqslant n_1, n_2 < 2^{-j}, 1/N \leqslant 2^j < 1, k \in \{V, H, D\}$. The set of inner products $\{\langle f, \psi_m\rangle\}_{m=0}^{N-1}$ is computed in $O(N)$ operations using the Fast Wavelet Transform, see Section IV-A.

Several Numerical Tours detail the implementation of the Fast Wavelet Transform in 1-D and 2-D. Some tours are focussed on the application of the Fourier basis to the analysis of sounds.

B. Approximation and Compression

The prototypal processing of a signal $f \in \mathbb{R}^N$ is the non-linear approximation using the best M coefficients, that is computed using a hard thresholding operator

$$f_M = H_T(f, \mathcal{B}) = \sum_{|\langle f, \psi_m\rangle| > T} \langle f, \psi_m\rangle \psi_m. \tag{1}$$

Note that the signal f_M is reconstructed from the thresholded coefficients using a fast reconstruction algorithm that has the same complexity as the forward transform. The asymptotic decay of the approximation error $\|f - f_M\|^2$ reflects the efficiency of the basis $\mathcal{B}$ to represent f with a few atoms. It is related to the efficiency of the basis for compression, denoising and regularization. It can be shown that the Fourier approximation decay for a C^α uniformly smooth image satisfies $\|f - f_M\|^2 = O(M^{-\alpha})$ while it is only of $\|f - f_M\|^2 = O(M^{-1/2})$ for a piecewise regular image with step discontinuities. In contrast, the wavelet approximation of a piecewise smooth image with edge singularities of finite length is $\|f - f_M\|^2 = O(M^{-1})$ which is optimal for the class of images with bounded variation, see [8].

Many image compression algorithms are based on transform coding, which is closely related to the non-linear approximation (1). The thresholding is replaced by an integer quantization

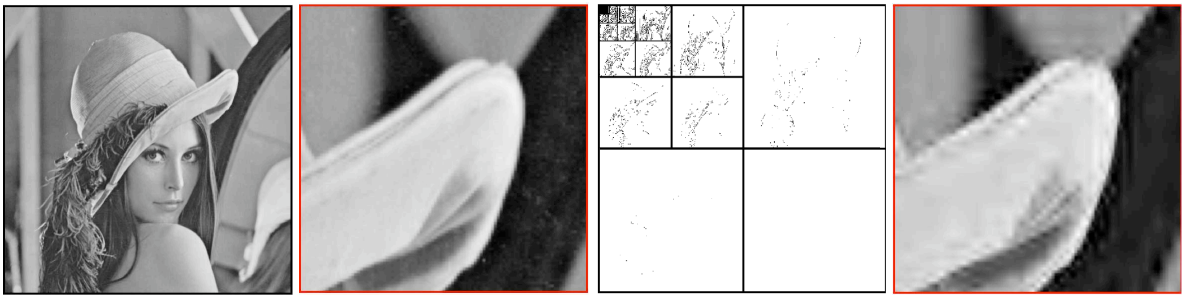
$$q_m = Q_T(\langle f, \psi_m\rangle) \in \mathbb{Z} \quad \text{where} \quad Q_T(x) = \mathrm{sign}(x) \left\lfloor \frac{|x|}{T} \right\rfloor. \tag{2}$$

These quantized coefficients are then entropy coded to produce a bit stream. This stream is expected to be short if the original coefficients are sparse, meaning that most of the q_m are zero. The decoder retrieves the integer q_m by inverting the coding, and then reconstructs a decoded signal or image

$$f_R = \sum_m T\,\mathrm{sign}(q_m) \left(|q_m| + \frac{1}{2}\right) \psi_m.$$

The quantization and de-quantization process produces an error $\|f - f_R\|$ that can be shown to be close to the non-linear approximation error $\|f - f_M\|$. Figure 5 shows an example of compression using an arithmetic coding of the quantized wavelet coefficients. State of the art compressors such as JPEG-2000 use advanced statistical coding schemes to exploit the local dependancies among wavelet coefficients, see [8].

Several Numerical Tours detail the use of orthogonal bases such as Fourier, local cosine and wavelets to perform approximation of signals and images. They also study binary coding using Huffman trees and arithmetic coding, and these technics are applied to perform signal and image compression. Some tours study the extension of these methods to video compression using optical flow and to multi-spectral image compression.



Original f_0 Zoom Wavelet transform Compressed f_R

Fig. 5. *Image compression using wavelet support codding.*

C. Denoising

Acquisition devices deteriorate a measured signal by some small fluctuations. The mathematical model of this noise requires the use of various random distributions. This includes additive Gaussian noise, Poisson shot noise or multiplicative noise.

The simplest model is the additive Gaussian white noise, where the observations are assumed to satisfy $y = f_0 + w \in \mathbb{R}^N$ where w is a Gaussian white noise of variance σ^2. A denoiser computes an estimate $f^\star$ of the unknown clean image f_0 from the observations y alone. The theoritical analysis of a denoiser studies the average risk $E_w(\|f_0 - f^\star\|)$ with respect to w. This risk cannot be computed in a real life situation, but the Numerical Tours proceed in an oracle manner by computing $\|f_0 - f^\star\|$ for a single observation, where f_0 is a fixed known image.

Linear denoising operates with a blurring $f^\star = y \star h$ that removes the high frequencies and thus some noise. Unfortunately, this blurring also destroys edges, so that an optimal filter that minimizes the risk usually leaves a significant amount of noise. More efficient denoisers are obtained by thresholding the observations in a wavelet basis $f^\star = H_T(f, \mathcal{B})$, as first proposed by Donoho and Johnstone [9]. This thresholding allows one to restore sharp edges because wavelets are more efficient than Fourier atoms to capture singularities. The theoritical value of the threshold is $T = \sqrt{2\log(N)}\sigma$ that ensures that $E_w(\|f_0 - f^\star\|)$ decays fast to zero when the noise level σ drops if $\|f - f_M\|$ also decays fast to zero when M increases, see [9], [8]. Empirically, the value $T = 3\sigma$ works well for natural images. Figure 6 shows an example of wavelet denoising.

Image f_0 Observations y Denoised $f^\star$

Fig. 6. *Example of denoising using wavelet thresholding.*

This thresholding in orthogonal bases is significantly improved by adding translation invariance. This can be obtained through cycle spinning, by denoising translated copies of the signal. Other enhancements include the use of more clever thresholding strategies such as thresholding blocks of coefficients [10].

Variational methods remove the noise by minimizing

$$f^\star = \operatorname*{argmin}_{f\in\mathbb{R}^N} \frac{1}{2}\|y-f\|^2 + \lambda J(f), \tag{3}$$

where $J(f)$ is a convex regularization functional and increasing $\lambda > 0$ increases the denoising strength. The Sobolev prior is $J(f) = \sum_x \|\nabla f(x)\|^2$, where $\nabla f(x)$ is a finite difference approximation of the gradient of f at pixel x. Minimizing J corresponds to a low pass filtering that blurs the edges. A popular prior for natural images is the total variation $J(f) = \sum_x \|\nabla f(x)\|$ that better respects the edges [11].

The Numerical Tours perform an exhaustive review of standard denoising methods, including wavelets and variational minimization schemes. They show several variations on these methods such as block thresholding, and detail several algorithms to minimize the convex problem (3). Some Numerical Tours also review recent state of the art methods such as non-local means filtering [12] and dictionary learning [13].

D. Inverse Problems

Acquisition hardware usually introduces some loss of information with respect to a high resolution image $f_0 \in \mathbb{R}^N$. This is modeled as $y = \Phi f_0 + w$ where $\Phi : \mathbb{R}^N \to \mathbb{R}^Q$ gathers only Q low resolution measurements. This models for instance the camera blur that removes high frequencies, or missing pixels because of damaged sensors.

Inverting this operator is impossible because $Q \leqslant N$ and Φ is ill-posed. Regularization theory further imposes some regularity on the solution $f^\star$ by solving a variational minimization

$$f^\star \in \operatorname*{argmin}_{f\in\mathbb{R}^N} \frac{1}{2}\|y-\Phi f\|^2 + \lambda J(f). \tag{4}$$

Regularization with the Sobolev prior avoids the explosion of the noise during the inversion but blurs the discontinuities. The total variation prior performs well for deconvolution when Φ is a blurring, and restores sharper transitions. Iterative schemes for non smooth optimization allows one to minimize the total variation prior [14], see also Section IV-B.

Sparsity regularization assumes that the image to recover is sparse in some ortho-basis $\mathcal{B} = \{\psi_m\}_m$, and makes use of the ℓ^1 prior $J(f) = \sum_m |\langle f, \psi_m\rangle|$. This method is extended to redundant dictionaries where one optimizes the ℓ^1 norm of the coefficients of $f^\star$ in this basis. This optimization is equivalent to the basis pursuit [15] to compute a sparse approximation of y in the dictionary $\{\Phi\psi_m\}_m$. The minimization (4) is computed using dedicated non-smooth solvers such as iterative thresholding [16]. Figure 7 shows an example of inpainting by minimizing (4) with an ℓ^1 prior. In this case, the linear operator is a masking

$$(\Phi f)(x) = \begin{cases} f(x) & \text{if} \quad x \notin \Omega, \\ 0 & \text{otherwise,} \end{cases}$$

where the mask Ω covers 70% of the pixels.

The recent method of compressed sensing [17], [18] makes use of a random operator Φ to acquire the signal f_0 with few samples y. If the signal f_0 is sparse enough, a solution of (4) can be shown to be close to f_0.

The Numerical Tours study a large variety of inverse problems, such as deblurring, inpainting, tomography and compressed sensing. Each time several regularization schemes are proposed such as Sobolev, TV, ℓ^1 in orthogonal and redundant dictionaries. These different scenarios allow one to study different optimization schemes to solve (4) such as gradient descent, projected gradient descent, proximal iterations [16] and convex duality [14].

E. Mesh Processing

Most signal and image processing methods are extended to triangulated meshes. This corresponds to a non-uniform sampling $\mathcal{V} = \{x_n\}_{n=0}^{N-1} \subset \mathbb{R}^d$, where $d = 2$ for planar meshes and $d = 3$ for surface meshes. The connectivity of the mesh is defined by a set of faces $\mathcal{F} \subset \{0,\ldots,N-1\}^3$, so that the mesh

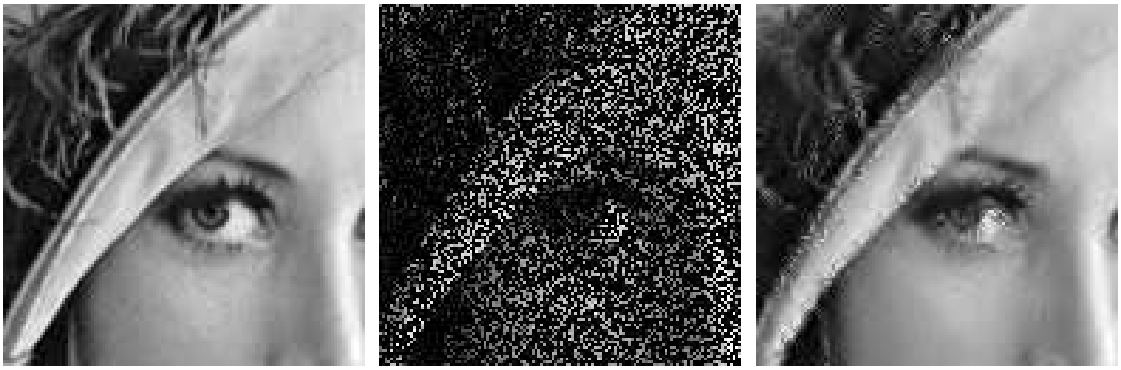


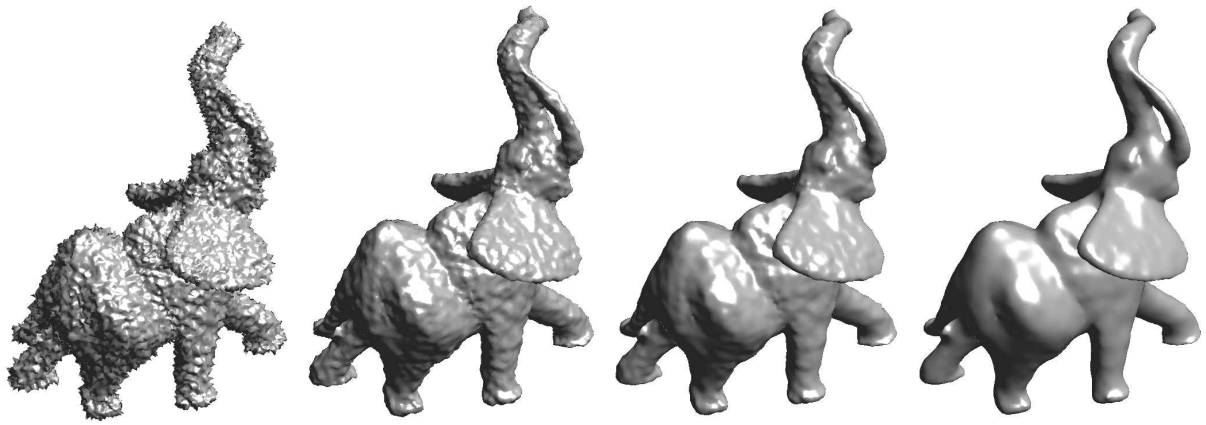
Image f_0 Observations $y = \Phi f + w$ ℓ^1 regularization $f^\star$

Fig. 7. *Image inpainting using ℓ^1 regularization in a translation invariant tight frame.*

is fully described by a matrix of size $3 \times N$ to represent $\mathcal{V}$ and a matrix of size $3 \times |\mathcal{F}|$ to represent $\mathcal{F}$. A couple of indexes $(i, j) \in \mathcal{E}$ is an edge if it belongs to a face. For a triangulation to be topologically valid, each edge should be incident to either two faces or only one face (for a boundary edge).

A signal $f \in \mathbb{R}^N$ defined on the mesh assigns to each vertex index i a value $f_i \in \mathbb{R}$. One can then process this vector to denoise or compress the signal. In some situations, one wants to process the surface itself, in which case the vertices $\{x_i = (x_i^1, \ldots, x_i^d)\}_i$ define d different signals, the coordinates of the points.

The simplest processing operators are linear $W \in \mathbb{R}^{N \times N}$ and sparse, so that $W_{i,j} \neq 0$ only if $(i, j) \in \mathcal{E}$. If $W_{i,j} \geqslant 0$ and $\sum_j W_{i,j} = 1$, then Wf is a low pass filtering that removes noise from a signal. Iterating W to compute $W^k f$ performs a progressive denoising of the signal, see Figure 8. Such an operator can also be used to extend the Sobolev prior to meshes as $J(f) = \sum_{i,j} W_{i,j} |f_i - f_j|^2$. Minimizing this prior also performs a denoising, and is used to interpolate deformations defined only at some vertices.



$k = 0$ $k = 1$ $k = 3$ $k = 5$

Fig. 8. *Examples of mesh denoising with iterated filtering $W^k f$.*

A 2-D parameterization of a 3-D mesh assigns to each vertex x_i a planar position (f_i^0, f_i^1). Classical methods finds a smooth parameterization by minimizing $J(f^0) + J(f^1)$ while fixing the positions of vertices on the boundary of the surface to be on a convex curve.

The Fourier basis is extended to the mesh setting by considering the singular vectors of the filtering operator W. These singular vectors are ordered by their frequency, and are used to perform mesh compression using (2). Wavelet bases can also be extended to meshes using the lifting scheme [19] with applications to compression and denoising.

The Numerical Tours detail the extension of several image processing problems to meshes, such as denoising, interpolation, compression. It also studies mesh deformation, flattening and parameterization.

F. Curve Processing

A central problem in computer vision is image segmentation, that requires to find a salient closed or open curve $t \in [0,1] \mapsto \gamma(t)$ in an image f. The curve is found by minimizing a weighted length

$$L(\gamma) = \int_0^1 P(\gamma(t))\|\gamma'(t)\|\mathrm{d}t \quad \text{where} \quad P(x) = \rho(\|\nabla f(x)\|) \tag{5}$$

which attracts γ to the salient features of f, and corresponds to a gradient based edge detector. The function $\rho > 0$ is decreasing so that J is low if γ passes in regions of high gradients.

Image f Metric $P(x)$ Evolution of γ

Fig. 9. *Example of geodesic active contour evolution for medical image segmentation.*

Object segmentation is obtained by minimizing (5) using a closed curve $\gamma(0) = \gamma(1)$. One can perform a gradient descent of L with respect to γ, which requires to solving a time evolution PDE, that drives the curve toward a local minimum of L. The snake active contours [20] perform this evolution using an explicit parameterization of the curve. It is also possible to use an implicit parameterization of the curve using a level set $\{\gamma(t) \setminus t \in [0,1]\} = \{x \in \mathbb{R}^2 \setminus \varphi(x)\}$. In this case, φ is evolved in time, which allows to track change in topology of the curve [21]. Figure 9 shows an evolution of a closed curve.

To detect curvilinear features in an image, one can use an open curve and impose boundary conditions $\gamma(0) = x_0$, $\gamma(1) = x_1$. One can compute the global minimizer γ of L which is the minimal length geodesic joining x_0 to x_1. The geodesic distance to x_0 is computed using the Fast Marching algorithm [22], and γ is extracted using a gradient descent of this distance starting from x_1.

The Numerical Tours explore object segmentation using parametric and level set active contours, for various energies. They also contain an extensive study of the computation of geodesic curves on 2-D, 3-D and triangulated mesh domains. The computation of geodesic distances is also applied to the problems of surface sampling and shape recognition.

IV. Numerical Tour Examples

This section reviews two numerical tours. It does not get into the implementation details, but sketches the main ideas underlying two important image processing methods. We refer to the online versions of the tour for in-depth explanations.

A. *Example #1 – Computing a Wavelet Transform*

The wavelet transform of a 1-D signal $f \in \mathbb{R}^N$ computes the inner products $d_j[n] = \langle f, \psi_{j,n} \rangle$ with discrete wavelet atoms $\psi_{j,n} \in \mathbb{R}^N$ indexed by the scale 2^j and the position n.

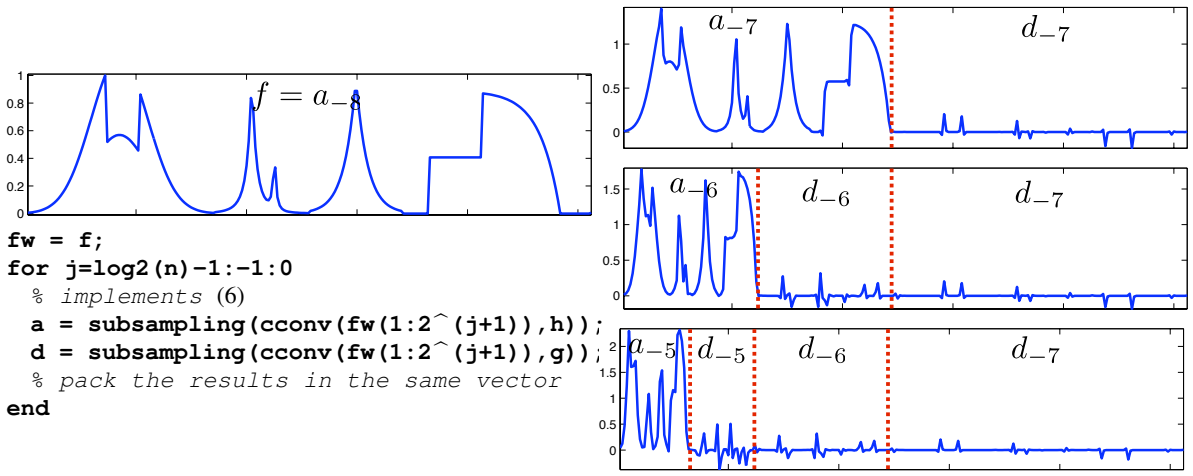
The only parameter of the wavelet transform is a low pass filter h. The associated high pass filter g is defined as $g[i] = (-1)^{i+1} h[1-i]$. Examples of valid h filters can be found in dedicated books such as [8].

The fast wavelet transform computes all the wavelet coefficients $\{d_j\}_{j=J+1}^{0}$ and a residual low frequency coefficient $a_0 \in \mathbb{R}$. The number of scales is $J = -\log_2(N)$. It keeps track of intermediates vectors a_j. The initial vector is $a_J = f \in \mathbb{R}^N$. The wavelet coefficients an the low pass residual are then computed as subsampled convolutions for $j = J, \ldots, -1$

$$a_{j+1} = (a_j \star h) \downarrow 2 \quad \text{and} \quad d_{j+1} = (d_j \star g) \downarrow 2 \tag{6}$$

where $\downarrow 2$ is the downsampling operator defined as $(a \downarrow 2)[k] = a[2k]$.

Figure 10 shows the code to perform the computation of the sub-sampled convolution, together with the display of the resulting wavelet coefficients d_j extracted. Note that the code performs the computation "in place" meaning that a single vector store both the already computed wavelet coefficients and the current vector a_j.




```
fw = f;
for j=log2(n)-1:-1:0
  % implements (6)
  a = subsampling(cconv(fw(1:2^(j+1)),h));
  d = subsampling(cconv(fw(1:2^(j+1)),g));
  % pack the results in the same vector
end
```

Fig. 10. *Wavelet decomposition algorithm. The figure on the right displays the evolution of the variable* `fw` *during the iterations.*

To perform an approximation, the wavelet coefficients are modified using a thresholding, as indicated in (1). This modification reads

$$\forall\, j,\ \forall\, n, \quad d_j[n] \longleftarrow \begin{cases} d_j[n] & \text{if} \quad |d_j[n]| > T, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

From these modified coefficients, one performs the inverse wavelet transform by inverting the step (8) for $j = -1, \ldots, J$

$$a_j = (a_{j+1} \star \tilde{h}) \uparrow 2 + (d_{j+1} \star \tilde{g}) \uparrow 2 \tag{8}$$

where $\tilde{h}[i] = h[-i]$ and $b = \uparrow 2a$ is the upsampling operator defined by $b[2i] = a[i]$ and $b[2i+1] = 0$. The approximated signal is then obtained as $f_M = a_J \in \mathbb{R}^N$. Figure 11 displays examples of wavelet non-linear approximations.

```
% implements (7)
fw = (abs(fw)>T).*fw;
f1 = fw;
for j=0:log2(n)-1
  a = f1(1:2^j);
  d = f1(2^j+1:2^(j+1));
  % implements (8)
  a = cconv(upsampling(a,1),h(end:-1:1),1);
  d = cconv(upsampling(d,1),g(end:-1:1),1);
  f1(1:2^(j+1)) = a + d;
end
```

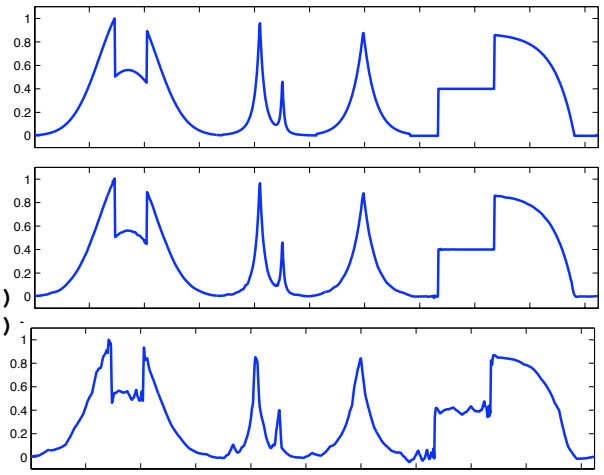


Fig. 11. *Wavelet approximation algorithm using thresholding and the backward wavelet transform. The figures on the right display the original signal f (top) and two approximations f_M computed with two different thresholds T.*

B. Example #2 – Total Variation Deconvolution

Deconvolution corresponds to the recovery from blurry measurements $y = \Phi f_0 + w$ where $\Phi f = f \star h$ and h is a known low pass filter.

To recover sharp edges one solves (4) using a total variation prior $J(f) = \sum_x \|\nabla f(x)\|$. Unfortunately, $J(f)$ is a non-smooth function of f, which makes the optimization difficult. To simplify the optimization, one can use a smoothed total variation prior

$$J_\varepsilon(f) = \sum_x \sqrt{\varepsilon^2 + \|\nabla f(x)\|^2}.$$

As ε tends to zero, $J_\varepsilon(f)$ approaches J but the recovery becomes more difficult to compute.

Since $J_\varepsilon(f)$ is a smooth function of f, one can use a gradient descent scheme to minimize (4). Starting from $f^{(0)} = y$, one iterates

$$f^{(k+1)} = f^{(k)} - \tau \left(\Phi^*(\Phi f^{(k)} - y) + \lambda \mathrm{Grad}(J_\varepsilon)(f^{(k)}) \right). \tag{9}$$

The gradient of J_ε reads

$$\mathrm{Grad}(J_\varepsilon)(f) = -\mathrm{div}\left(\frac{\nabla f}{\sqrt{\varepsilon^2 + \|\nabla f(x)\|^2}} \right), \tag{10}$$

where $\mathrm{div} = -\nabla^*$ is a finite difference approximation of the divergence. The iterates $f^{(k)}$ converge to a minimizer $f^\star$ of (4) if the step size is small enough with respect to ε,

$$\tau < \frac{2}{\|\Phi^*\Phi\| + \lambda \|\mathrm{div} \circ \nabla\| / \varepsilon}.$$

Here $\|\Phi^*\Phi\| = \max_\omega |\hat{h}(\omega)|^2$ is the square of the largest singular value of Φ and $\|\mathrm{div} \circ \nabla\| = 8$ for a standard finite difference approximation of the gradient. Figure 12 shows the results obtained for two different values of λ.

CONCLUSION

We have presented in this paper the Numerical Tour of Signal Processing. It is a large collection of Matlab/Scilab experiments that guides the user in the jungle of recent advanced signal, image and mesh processing algorithms. We encourage the reader to visit the homepage of the tours to discover many useful resources.

```
% shortcut for the convolution
Phi = @(x)real(ifft2(fft2(x).*fft2(h)));
% initialization
f = y;
for i=1:niter
  % compute the gradient (10)
  Gr = grad(f);
  d = sqrt( epsilon^2 + sum3(Gr.^2,3) );
  G = -div( Gr./repmat(d, [1 1 2]) );
  % gradient descent step (9)
  f = f - tau*( Phi(Phi(f)-y) + lambda*G);
end
```

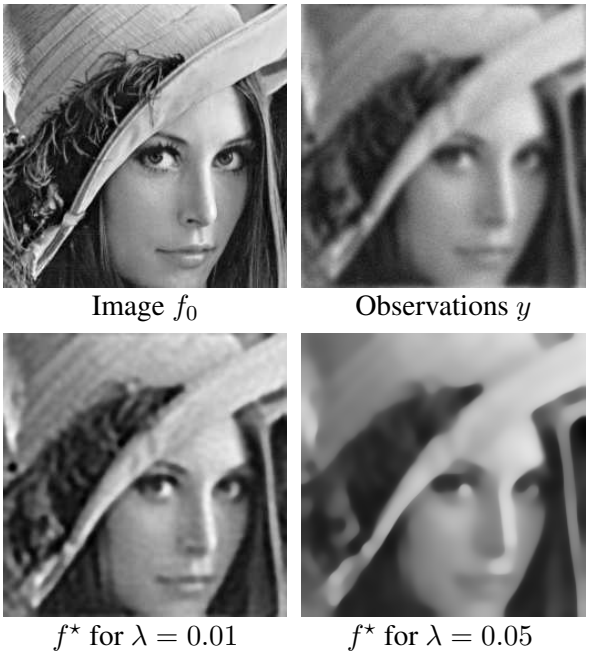


Fig. 12. *Total variation deconvolution.*

REFERENCES

[1] G. C. Orsak and D. M. Etter, "Collaborative SP education using the Internet and MATLAB," *IEEE Signal Processing Magazine*, vol. 12, no. 6, pp. 23–32, Nov. 1995.

[2] R. W. Stewart, S. Weiss, J. D. Quayle, and D. Garcia-Alis, "Digital signal processing education: Technology and tradition," in *1st Signal Processing Education Workshop*, Oct. 2000.

[3] C. Hu, "Integrating modern research into numerical computation education," *Computing in Science and Engg.*, vol. 9, no. 5, pp. 78–81, 2007.

[4] M. Barni and F. Perez-Gonzalez, "Pushing science into signal processing," *Computing in Science and Engineering*, vol. 22, no. 4, pp. 120–119, 2005.

[5] P. Vandewalle, J. Kovacevic, and M. Vetterli, "Reproducible research in signal processing," *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 37–47, May 2009.

[6] D. L. Donoho, A. Maleki, I. U. Rahman, M. Shahram, and V. Stodden, "Reproducible research in computational harmonic analysis," *Computing in Science and Engineering*, vol. 11, no. 1, pp. 8–18, 2009.

[7] C. T. Silva and J. E. Tohline, "Computational provenance," *Computing in Science and Engineering*, vol. 10, no. 3, pp. 9–10, 2008.

[8] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego: Academic Press, 1998.

[9] D. Donoho and I. Johnstone, "Ideal spatial adaptation via wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, Dec 1994.

[10] T. Cai, "Adaptive wavelet estimation: a block thresholding and oracle inequality approach," *Ann. Statist*, vol. 27, no. 3, pp. 898–924, 1999.

[11] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, no. 1-4, pp. 259–268, 1992.

[12] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *SIAM Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[13] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Proc.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[14] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical Imaging and Vision*, vol. 20, pp. 89–97, 2004.

[15] S. S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[16] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. on Pure and Appl. Math.*, vol. 57, pp. 1413–1541, 2004.

[17] D. Donoho, "Compressed sensing," *IEEE Trans. Info. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[18] E. Candès, J. Romberg, and T. Tao, "Signal recovery from incomplete and inaccurate measurements," *Commun. on Pure and Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2005.

[19] P. Schröder and W. Sweldens, "Spherical Wavelets: Efficiently Representing Functions on the Sphere," in *Proc. of SIGGRAPH 95*, 1995, pp. 161–172.

[20] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[21] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, February 1997.
[22] J. Sethian, *Level Sets Methods and Fast Marching Methods*, 2nd ed. Cambridge University Press, 1999.