Master Thesis

Image Reconstruction by Inverse Methods and Application to Space Imaging

University of Liège

Faculty of Sciences

Master in Space Sciences - Professional Focus

Author:
Florian CANION

Supervisor:
Marc GEORGES

Reading committee members:
Denis GRODENT, Serge HABRAKEN, Murielle KIRKOVE

Academic Year 2023 - 2024

Acknowledgements

These few words aim to thank all the people whose support made possible the achievement of this master thesis.

First and foremost, I would like to deeply thank Dr. Marc Georges, my supervisor, firstly for allowing me to carry out the present work at the Centre Spatial de Liège and secondly for his help and advice during all the second year of my master's degree.

My thanks also to Laurent Jacques, professor at UCLouvain, whose knowledge and expertise in the field under study were very helpful. Additionally, thank you to the members of the reading committee and the people who have read or reread this document for their time.

I am also thankful to my friends and more generally to all the people I met during my university background and without which the last years would never have been the same. Thanks also to Clément and Émile with whom I worked during the realisation of this master thesis and whose assistance was primordial to the smooth running of this end-of-study project.

Last but not least, I am particularly grateful to my family and more especially to my parents for their daily support and for their guidance throughout these years of study. Without them, this academic journey would never have been possible. I will never thank them enough for always believing in me and in everything I undertake.

To my grandfather.

Abstract

Inverse methods are at the basis of the resolution of numerous applications, that notably take place in the context of signal processing. In particular, these techniques appear to be efficient for images-related problems, in which the analysed data are most of the time incomplete or at least perturbed by unwanted external contributions. These situations are commonly encountered when carrying out classical optical and remote sensing acquisitions, and it is therefore primordial to develop tools that will bring relevant solutions to these potential issues. On that purpose, series of numerical algorithms are currently tested and make use of both the so-called images sparsity property and the wavelet theory to recover the missing components of a broad variety of analysed images.

This master thesis focuses on the development of some of these numerical algorithms, that are optimised in the present work through a succession of tests. First, an overall state of the art is provided and covers all the concepts of interest, including images properties, the selected inpainting technique that performs images completions, thanks to adequate images sampling and threshold-based method that are described as well, and image deconvolution involving a point spread function. An entire section is then dedicated to all the digital image processing steps that mainly rely on images produced by a compressive sensing imager designed at the Centre Spatial de Liège. The correlated methodologies, followed to produce satisfactory images reconstructions, are finally applied to some damaged scenes produced either in laboratory or during an existing space mission. The final selection of the mission of interest has been inclined, thanks to a bibliographic study, towards the Landsat 7 mission, for which one of its component's malfunction led to the production of incomplete scientific data.

Keywords: image processing, inverse problem, inpainting, wavelet theory, iterative hard thresholding, image deconvolution, satellite data.

Table of contents

List of Figures

List of Tables

List of Acronyms

1D (2D)	One-Dimensional (Two-Dimensional)
BCM	Block-Circulant Matrix
CM	Circulant Matrix
CMOS	Complementary Metal Oxide Semiconductor
CS	Compressive Sensing
CSL	Centre Spatial de Liège
CWT	Continuous Wavelet Transform
DMD	Digital Micromirror Device
DCWT	Discretised Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
ETM+	Enhanced Thematic Mapper Plus
FR	Filling Ratio
FOV	Field Of View
FT	Fourier Transform
GK	Gaussian Kernel
GT	Ground Truth
HT	Hadamard Transform
IFOV	Instantaneous Field Of View
IHT	Iterative Hard Thresholding
MIR	Mid-Infrared
MSE	Mean Squared Error
NIR	Near Infrared
NOI / niter	Number Of Iterations
OTF	Optical Transfer Function
PSF	Point Spread Function

PSNR	Peak Signal-to-Noise Ratio
RGB	Red Green Blue
SLC	Scan Line Corrector
s.t.	subject to
STFT	Short-Time Fourier Transform
SWIR	Short-Wave Infrared
TIW	Translation Invariant Wavelet
USGS	United States Geological Survey
WT	Wavelet Transform
ZP	Zero Padding

Chapter 1

Introduction

Image processing constitutes a non negligible part of scientific data analysis. Indeed, images are omnipresent in lots of practical fields, including for example optical instruments calibration, photography, microscopy and many other domains of interest. One of these domains concerns space applications, for which interest is constantly growing and will certainly continue to evolve in this direction for many years. In this context, Earth monitoring plays a significant role and deserves to be studied in depth. However, the related optical devices embedded in observation satellites are generally submitted to a certain amount of constraints that limit, in some extent, the interpretation and the readability of the acquired data (in other terms, of the images). In addition, some unexpected events can also occur in the course of space missions, leading in this case to the production of erroneous or incomplete information that cannot be read properly without the use of appropriate tools. To overcome these potential issues, a series of possible techniques exist and include notably the resolution of inverse problems, the latter being considerably present in the context of image reconstruction.

Contrary to the traditional so-called direct problems, in which a set of possible consequences is deduced from a few initial conditions and causal effects, inverse problems work in reverse and determine causes based on observations. The difficulties therefore lie in the way in which the cause/consequence relationship is interpreted. Actually, an inverse problem usually gathers many solutions, and it is thus mandatory to apply some restrictions in order to find the result that fits the most with the specific studied context. On that purpose, various types of optimisation algorithms exist and allow to operate converging reconstruction processes. More precisely, the composition of these algorithms varies depending on the type of inverse problem that need to be solved. One can then distinguish linear inverse problems and non-linear ones. While linear problems involve usual functional analysis and linear algebra developments, non-linear problems are constructed on the basis of differential equations or partial derivatives and their resolution is consequently more complex [1].

In the framework of this thesis, the focus is on solving exclusively linear problems that are directly linked to image processing and that can be subdivided in two main types. The first one, on the one hand, is named inpainting and is applicable when the analysed images are composed of missing parts or holes. These lost elements can be retrieved thanks to the application of the wavelet theory, which consists of using primary oscillating functions that will probe the observed signal and determine its main structural configurations. The second one, on the other hand, concerns image deblurring and works on the basis of traditional deconvolution techniques.

Being focused on the numerical aspect that directly follows images acquisitions, this paper aims to explore the possibilities offered by the tested reconstruction algorithms and to propose certain improvements in connection with the applications in which they are involved. On that purpose, several parameters contained in the preliminary version of some existing codes are tested and optimised. Additional techniques are also implemented to overcome possible limitations encountered when solving the inverse problems of interest.

This master thesis is structured in this way. First, a state of the art related to the inverse methods presented in this document is provided. This includes notably the mathematical description of the different algorithms involved in the image reconstruction processes that are studied in the present work, namely the Iterative Hard Thresholding (IHT) one and the image deconvolution one. Then, all the methodology followed to realise the corresponding image processing is exposed, and is accompanied by the main results obtained during the different stages. The next part presents some applications linked to image inpainting, with emphasis on observations simulated by an optical system developed at the Centre Spatial de Liège (CSL) and actual measurements coming from the Landsat 7 space mission. Finally, a conclusion gathers the main ideas developed in this paper and provides prospects on possible results improvements.

The primary objective of this end-of-studies work is to bring relevant solutions for image processing that could be re-implemented in further space missions data analyses, especially in the case of satellite-based Earth observation and monitoring. Having been carried out at the CSL, this project is in close collaboration with the master thesis of Émile Ruwet, another student who designed the optical imaging device that generated part of the images analysed here. The contributions of both works are intended to enhance the thesis of Clément Thomas, a PhD student working at the CSL as well. His research explores notably the possibilities offered by Compressive Sensing (CS), an image reconstruction approach strongly connected to the one of inpainting, and which is also focused on Earth observation applications.

Chapter 2

State of the art

This chapter is focused on the theoretical concepts and mathematical descriptions that are behind the inverse methods applied in this master thesis. Before exploring the data reconstruction processes, a section dedicated to images and their characterisitics is provided in order to understand how they can be manipulated. This introductory section is directly followed by an overview of inpainting and of the related inverse problem that needs to be solved, and by a presentation of the concepts behind sparsity prior and atoms bases, significant elements that constitute the main properties and tools that are introduced in the reconstruction algorithms. These different algorithms, used either for image completion or image deblurring, are then explained in details in order to understand each principles they are based on, and how their converging optimisation process can lead to a solution. More precisely, the cases of Iterative Hard Thresholding (IHT) and image deconvolution using a Point Spread Function (PSF) are studied in depth.

In this part of the thesis, all that is designated as a dimension is a non-zero natural number, and a writing convention is established such that vectors and matrices are written in bold. Besides that, most of the information described in sections 2.2, 2.3, 2.4 and 2.5 come from references [2] and [3]. Extra explanations in sections 2.4.1 and 2.5.2 rely on [4] and [5], and any other development inspired from literature is clearly indicated in the text.

2.1 Images properties

Before operating some processing steps on images, it is first of all mandatory to have an idea of how they can be numerically interpreted. Each image as a specific format, and the one selected for all the images processed in this work is the bitmap one. A bitmap image, also called a raster graphic, is defined as a rectangular and regular grid of pixels having a N x M dimension. In this configuration, N refers to the number of columns and M to the number of lines constituting the bitmap [6]. Numerically speaking, images are therefore no less than matrices for which each of the components can be assimilated to one of the correlative image pixel having the same position and the same content. Images are thus submitted to the same kind of operations than the ones related to classical matrices, which appears to be considerably useful in terms of images manipulation.

Regarding the pixels content, each of the bitmap cells is filled with a specific value called the colour depth. This information is representative of the light intensity gathered in one single pixel. It directly depends on the number of coded bits, the latter determining the capacity storage of the pixels. All the images selected for the reconstruction processes explained further are monochromatic and have 8 coded bits, corresponding to 256 possible states. Light intensity can this way be subdivided into 256 different shades, going from 0 for the darkest shade to 255 for the brightest one, in the case of a grayscale image [6].

For colour images, however, the composition is a little bit more complex. Usually, colour images are the result of stacking several monochromatic images. This is notably true for Red Green Blue (RGB) images in which three different channels are stacked, each of them corresponding to a primary colour layer. Some numerical functions exist to split the original layers and convert them into grayscale intensity ones. The latter can then be combined in order to produce a global grayscale image that has the same patterns that the authentic one and whose pixels contain one of the 256 possible information as explained just before. Due to the segmentation into different channels, colour images contain obviously more data than the grayscale ones and the resulting number of coded bits that is associated with them is greater. Typically, the so-called true colour images have a colour depth of 24 bits, which allows to separate them in three monochromatic sub-images being each coded in 8 bits (for a RGB configuration for example) [7].

It is worth to notice that, depending on the application, some colour scientific images are not limited to only three different information layers as it is the case in the RGB format, and can harbour many components in accordance to the observed wavelength ranges. Understanding how the different channels are organised in a colour image is therefore primordial when manipulating multi-spectral images, which often happens when processing scientific data.

2.2 Inpainting

As mentioned in the introduction, inpainting consists of filling in the missing elements of a damaged image, or in other words to complete the holes corresponding to data losses. The typical inpainting problem can be deduced from the general form of an inverse problem, the latter being simply expressed as

$$\mathbf{y} = \mathbf{A}\ \mathbf{x} + \mathbf{q} \tag{2.1}$$

where $\mathbf{x}$ ($\in \mathbb{R}^N$) is the vector form of the original complete image, $\mathbf{A}$ ($\in \mathbb{R}^{M\times N}$) is a measurement matrix and $\mathbf{y}$ ($\in \mathbb{R}^M$) is designed as the observation, or equivalently, as the vector form of the incomplete image obtained on the basis of $\mathbf{x}$. With regard to $\mathbf{q}$, this quantity represents the image noise. For the sake of simplicity, all the images submitted to the reconstruction processes taking place in this work are assumed to be noiseless (i.e. $\mathbf{q} = 0$).

Two significant considerations can already be highlighted from equation 2.1. Firstly, it is mandatory to impose that the dimension of $\mathbf{y}$ must be strictly smaller than the dimension of $\mathbf{x}$ (in other words, the criterion $\mathrm{M} < \mathrm{N}$ must be respected). Indeed, $\mathbf{y}$ being the damaged measurement of $\mathbf{x}$, it must necessarily contain fewer components and less information than the initial image. Secondly, the measurement matrix $\mathbf{A}$ is the key element that allows to make the link between $\mathbf{y}$ and $\mathbf{x}$. The main objective when trying to solve problems as the one discussed here is thus to look for a matrix $\mathbf{A}$ for which the properties are consistent with the problem requirements and conditions.

An equivalent formulation of the initial problem, considering this time square images of dimension $\mathrm{n} \times \mathrm{n} = \mathrm{N}$ can be written as

$$\boldsymbol{f} = \boldsymbol{\Phi} \circ \boldsymbol{f_0} \tag{2.2}$$

with $\boldsymbol{f_0}$ ($\in \mathbb{R}^{n \times n}$) the original undamaged image and $\boldsymbol{f}$ ($\in \mathbb{R}^{n \times n}$) the observed damaged image. In this situation, $\boldsymbol{\Phi}$ ($\in \mathbb{R}^{n \times n}$) can be interpreted as a mask, the latter being applied on the initial image $\boldsymbol{f_0}$.

Here are some more details about the mask $\boldsymbol{\Phi}$. This object is in fact a binary matrix, being exclusively composed of 0 and 1 elements by definition. These elements are distributed in such a way that they form a given pattern, reproducing the one of the expected missing elements appearing in the observation $\boldsymbol{f}$. The application of $\boldsymbol{\Phi}$ on $\boldsymbol{f_0}$ is implemented in numerical codes as the same manner as a Hadamard product occurring between the two matrices. This specific product differs from the usual matrix product and is an operation between two matrices of same dimensions. Typically, the Hadamard product of a matrix $\mathbf{U} = (u_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ ($\in \mathbb{R}^{m \times n}$) with another matrix $\mathbf{V} = (v_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ ($\in \mathbb{R}^{m \times n}$) is defined as

$$(\boldsymbol{U} \circ \boldsymbol{V})_{ij} = u_{ij} v_{ij} \tag{2.3}$$

in which $\circ$ is the usual representation of the Hadamard product operator that also appears in equation 2.2.

In consequence, the above expression 2.3 can be interpreted in this way. The Hadamard product, also called element-wise product, is similar to a superposition of two matrix (or images), resulting in a scalar multiplication of each of the superposed cells (or pixels) [8]. This operation is the expected one when masking an image. This way, the masking operator $\boldsymbol{\Phi}$ is somehow applied to the complete image $\boldsymbol{f_0}$. More precisely, binary components of $\boldsymbol{\Phi}$ affect the elements of $\boldsymbol{f_0}$ in the following way. While 1 elements keep the corresponding $\boldsymbol{f_0}$ cells unchanged, 0 elements set them to zero, creating missing parts having the same shapes than the mask pattern. An illustration of this is provided at figure 2.1.

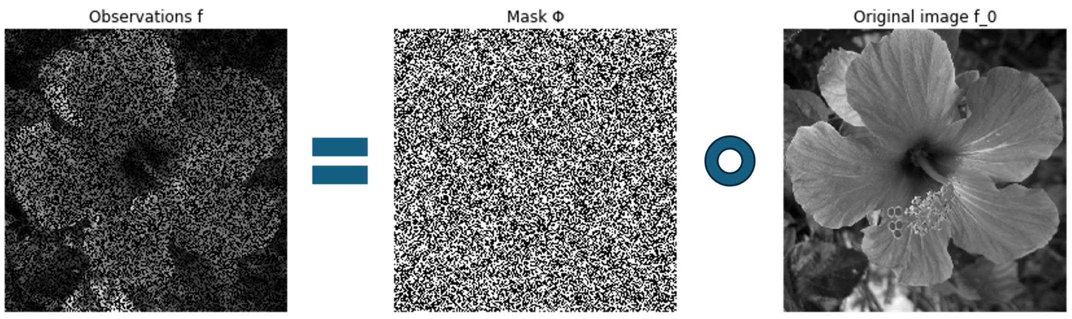


Figure 2.1: Visualisation of the image masking operation with a randomly generated mask.

A link between the two formulations of the initial problem, introduced in this section, can be made and establishes a possible definition of the classical inpainting problem. This new approach consists of preserving the vector form of equation 2.1 while keeping in mind the characteristics of the mask defined in formula 2.2. In this configuration, the measurement matrix **A** is built by taking into account two main assumptions. First, **A** is defined as a matrix of dimensions M × N, M being in this case the number of pixels of $\boldsymbol{\Phi}$ for which the value is 1. Then, **A** must contain a single element of value 1 per line and at most one element of value 1 per column. This second constrain imposed on **A** ensures that the **Ax** product ($\in \mathbb{R}^M$) only conserves the pixels stemming from the lines of **A**. If these criteria are respected when constructing **A**, then this matrix is suitable for the studied inpainting problem, which can therefore be reformulated in the same way as equation 2.1.

In reality, this overview of the inpainting problem only constitutes a minuscule part of all the possible researches linked to the CS mentioned in the introduction. CS is actually a method that makes use of two of the main signals properties, namely their ability to be compressible and sparse (the latter being described in the next section). Thanks to these two characteristics, the CS method is capable to recover signals by sampling them, with a number of measurements that is quite small compared to the initial amount of information gathered in the original analysed data. Its utility is therefore primordial for many scientific and technological applications, encompassing among others the resolution of mathematical, computer sciences and physics-related problems. Signals recoveries are then made possible by involving reconstruction algorithms, that can be classified into various types including notably the so-called thresholding-based methods [9]. In this work, the inpainting problem is resolved with one of these threshold-based algorithms, that is described in section 2.4.

2.3 Sparsity and atoms bases

2.3.1 Image sampling and decomposition

In the general context of signal processing, the vector **x** defined in the previous section is obtained by sampling N times a continuous initial signal. A discretisation is this way operated, so that **x** contains N elements that are representative of the whole analysed data. As a result, a reorganisation of all the information included in the primary signal is carried out, and is at the basis of the formation of the new vector-like object $\mathbf{x}$, which is quite more manageable compared to the original continuous set of information. In order to determine if the non-discretised signal can be precisely retrieved from $\mathbf{x}$, it is mandatory to refer to a criterion stated by the Nyquist (or Nyquist-Shannon) theorem. According to this theorem, the initial signal can be totally recovered if the sampling frequency f_s (namely the number of created samples per unit of time) is at least twice the Nyquist frequency f_N (i.e. the maximum possible frequency that is established in the signal). Nyquist theorem is then simply formulated as

$$f_s \geq 2f_N. \tag{2.4}$$

For images, however, the discretisation step is immediately produced by the pixels segmentation originating from the related optical or numerical image acquisition system. As a consequence, the entire image content (or its vector equivalent $\mathbf{x}$) can be adequately recovered with a number of elements K being much lower than the total amount of pixels N that initially forms the image. This affirmation can be established thanks to the knowledge of an important property of images called the sparsity prior. In reality, this image attribute offers the possibility to decompose the vector representation of the image as a series of K fundamental functions called atoms. These elementary functions are gathered in a specific basis and the quantity of available atoms involved in the image recuperation must verify the condition K $\ll$ N. The expected decomposition is then expressed as

$$\mathbf{x} = \sum_{k=1}^{K} \alpha_k \psi_k = \boldsymbol{\Psi}\boldsymbol{\alpha} \tag{2.5}$$

where $\boldsymbol{\alpha}$ is a vector of dimension N containing the α_k coefficients and having K $\ll$ N non-zero elements. $\boldsymbol{\Psi}$ is a basis of atoms ψ_k.

As much of the coefficients composing the $\boldsymbol{\alpha}$ vector presented in the previous relation are equal to zero, $\boldsymbol{\alpha}$ is said to be sparse. In opposition to so-called dense arrays, for which most of the elements are non-zero, sparse matrices or vectors are mathematical objects that have most of their elements equal to zero. In the present case, $\boldsymbol{\alpha}$ only has K non-zero α_k coefficients and is for that reason called a K-sparse vector. This sparsity aspect of $\boldsymbol{\alpha}$ is only permitted in the case the $\boldsymbol{\Psi}$ basis and its relative collection of fundamental functions ψ_k are chosen such that they are suitable for the linear combination of $\mathbf{x}$, in agreement with the α_k coefficients. A small note regarding the expression 2.5 is that $\boldsymbol{\Psi}$ can be generalised to the case of a dictionary which may harbour functions coming from multiple bases. The related atoms allow then to approximate an image as illustrated in figure 2.2. As one can expect, the greater the number of atoms, the more the approximation is similar to the initial image $\mathbf{f}$.

Many types of atoms bases actually exist, the most known ones being probably those associated to the Fourier Transforms (FTs), Hadamard Transforms (HTs) and Wavelet Transforms (WTs). However, the problems involving these particular bases have a high complexity of resolution that is caused by the huge variety of existing functions, from

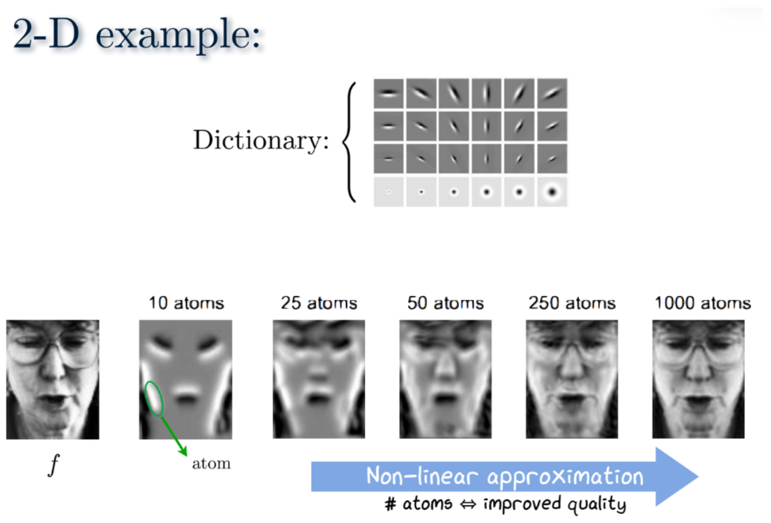


Figure 2.2: Image sparse approximation using a predefined dictionary that contains two-dimensional atoms. Above: example of dictionary composed of fundamental functions. Below: image approximation performed by considering the atoms present in the dictionary; From [2].

which numerous solutions can be determined. Bases that are discussed here are therefore overcomplete in some way, and must be restricted to the functions that will serve to perform images sparse approximations. This justifies the need of using optimisation algorithms when manipulating such mathematical tools.

2.3.2 Wavelet Transforms

WTs are known to be efficient processing tools for data restoration and compression. Particularly, their involvement appears to be primordial in the resolution of traditional inpainting and CS inverse problems. This especially comes from their ability to transcribe and represent signals on the basis of a number of parameters that is smaller than the amount of samples derived from the analysed data.

The reconstruction algorithms developed in sections 2.4 and 2.5 make use of these WTs to accomplish a regularisation method that is described later for IHT and image deconvolution applications. Before explaining in details the mathematical backgrounds of the studied algorithms, it is essential to explore the principal characteristics of WTs to have a better comprehension of their utility. On that purpose, a general definition of WTs and their main forms is first exposed to clarify on which theoretical considerations they are built. Then, the specific case of images decomposition is introduced through the notion of Discrete Wavelet Transforms (DWTs).

2.3.2.1 General definition

Even if they constitute a separate class of transforms, WTs are strongly correlated with usual FTs. Widely used in physics-related topics, FTs are indispensable in the field of signal processing and make the link between frequency and time domains of the signal of interest. This connection is established by the following function which is an extension of classical

Fourier series

$$F(\omega) = \int_{-\infty}^{+\infty} \mathbf{f(t)} e^{-i\omega t} dt \tag{2.6}$$

where ω is a frequency, $\mathbf{f(t)}$ is the probed signal that can be either a 1D one or a multidimensional one (for instance, an image is perceived as a 2D signal), and t is the time component. Expressed thanks to a complex exponential, the frequency information is thus directly deduced from the temporal evolution of the signal. Depending on the needs, it may however be interesting to probe both the frequency and temporal signal contents. Purposely, a more complete version of the expression 2.6, designated as the Short-Time Fourier Transform (STFT), is written

$$STFT\{\mathbf{f(t)}\}(\tau, \omega) = \int_{-\infty}^{+\infty} \mathbf{f(t)} w(t-\tau) e^{-i\omega t} dt. \tag{2.7}$$

This time, $\mathbf{f(t)}$ is constrained by a window function w(t) that delimits the temporal length of the signal around a time τ. Signal's frequency and time information is this way accessible for predetermined time intervals. Another option for the writing of equation 2.7, that takes into account some basis functions $k_{\tau,\omega}(t)$, is

$$STFT\{\mathbf{f(t)}\}(\tau, \omega) = \int_{-\infty}^{+\infty} \mathbf{f(t)} k_{\tau,\omega}(t) dt. \tag{2.8}$$

By making appropriate changes of variables, in which a is a scale factor and b another representation of τ for the time shift operated by the window function w,

$$\begin{cases} \omega = \frac{1}{a} \\ \tau = b \end{cases} \tag{2.9}$$

$k_{\tau,\omega}(t)$ can be expressed hereunder with a more adequate shape

$$k_{b,a}(t) = \frac{1}{\sqrt{a}} \gamma \left(\frac{t-b}{a} \right) \tag{2.10}$$

in which $\gamma(\cdot)$ is called a wavelet, an oscillating function that brings together both time and frequency dependencies. Furthermore, the typical evolution of a wavelet is marked by a rapid decay. This evolving trend can be assimilated to the one of the exponential decay present in FTs formulations, and, considering this resemblance, an expression similar to the relation 2.6 can be found. This way, another type of transform, the WT, is formulated and can be generally defined by the Continuous Wavelet Transform (CWT)

$$CWT\{\mathbf{f(t)}\}(b, a) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} \mathbf{f(t)} \gamma \left(\frac{t-b}{a} \right) dt. \tag{2.11}$$

A discretised version of the CWT exists and is naturally called a Discretised Continuous Wavelet Transform (DCWT). Contrarily to the CWT, the DCWT does not have the same aspect than a FT but is rather viewed as the equivalent to a Fourier series in which the notion of wavelet is included. The discretisation, indexed by an integer number m, takes place at the level of the time variable. The signal is then sampled over time, not continuously as before but rather with a precise amount of different time values t_m. Based on this new temporal segmentation, a rewriting of equation 2.11 leads to

$$DCWT\{\boldsymbol{f(t_m)}\}(b,a) = \frac{1}{a}\sum_m \boldsymbol{f(t_m)}\gamma\left(\frac{t_m - b}{a}\right). \tag{2.12}$$

2.3.2.2 Discrete Wavelet Transforms

In order to carry out images reconstructions, a series of operations is needed and implies knowing the so-called wavelet coefficients, the latter being obtained by decomposing images in a very specific way. To do so, a new sort of WT, the Discrete Wavelet Transform (DWT), not to be confused with the DCWT, is introduced and is inspired by the general wavelet theory exposed in the previous section. The ways an image decomposition by a DWT is performed and the wavelet coefficients are determined are described hereafter.

Here are the initial conditions of the image decomposition problem. A square image $\mathbf{f(x)}$, filled with N pixels and having $\sqrt{N} \times \sqrt{N}$ dimensions, is spatially distributed in a 2D frame. This frame is characterised by a set of directions $\mathbf{x} \equiv (x,y)$ $(\in \mathbb{R}^2)$ and by the image pixels locations $\mathbf{n} \equiv (n_x, n_y)$ $(\in \mathbb{Z}^2)$. Apart from that, the image will be submitted to the action of four restructuring functions, that will influence the image spatially speaking. Indeed, image processing does not take into consideration any time evolution when manipulating a single image and all the time variables, such as the ones mentioned in section 2.3.2.1, must therefore be replaced by spatial elements. The first structure-related function, $\mathit{\Phi}(\mathbf{x})$, is a scaling one that will affect the image in a bi-dimensional manner thanks to two 1D sub-functions, $\phi(x)$ and $\phi(y)$, that will act singly in the corresponding preferential direction

$$\mathit{\Phi}(\mathbf{x}) = \mathit{\Phi}(x,y) = \phi(x)\phi(y). \tag{2.13}$$

The three other functions $\psi(\mathbf{x})$ are mother wavelet functions that will have both a shrinkage and a translation effect. These modifications can be of different natures, the possibilities being either vertical (V), horizontal (H) or diagonal (D). The three configurations results from combinations of the primary functions ϕ and ψ and represent vertical, horizontal and diagonal frequencies contained into the image

$$\begin{cases} \psi^V(\mathbf{x}) = \psi(x)\phi(y) \\ \psi^H(\mathbf{x}) = \phi(x)\psi(y) \\ \psi^D(\mathbf{x}) = \psi(x)\psi(y) \end{cases}. \tag{2.14}$$

These primary functions can be developed as

$$\begin{cases} \phi_{j,\mathbf{n}}(\mathbf{x}) \equiv \frac{1}{2^j}\phi\left(\frac{\mathbf{x}-2^j\mathbf{n}}{2^j}\right) \\ \psi^r_{j,\mathbf{n}}(\mathbf{x}) \equiv \frac{1}{2^j}\psi^r\left(\frac{\mathbf{x}-2^j\mathbf{n}}{2^j}\right) \end{cases} \tag{2.15}$$

where j $\in [J_0; J-1]$ is a resolution of one of the image decomposition steps described later, and r $\in \{"H","V","D"\}$ transcribes the directional behaviour of the function. One can notice than the function arguments in expression 2.15 have the same appearance than the ones in $\gamma\left(\frac{t-b}{a}\right)$ introduced at equation 2.10, with the exception that the time t, as well as b and a coefficients, are respectively replaced by a spatial reference $(\mathbf{x})$, $2^j\mathbf{n}$ and 2^j. While $2^j\mathbf{n}$ indicates a position (translation effect of $\psi(\mathbf{x})$), 2^j corresponds to a scale factor (shrinkage effect of $\psi(\mathbf{x})$), which justifies the appellation of a in section 2.3.2.1.

Based on all these definitions, the sought DWT is designated as the following operation, that establishes a transition between the original image $\mathbf{f(x)}$ and a matrix $\mathbf{W}$

$$DWT: \qquad \mathbf{f(x)} \rightarrow \mathbf{W} = \begin{cases} a_{J_0,\mathbf{n}} = \langle \mathbf{f}, \phi_{J_0,\mathbf{n}} \rangle \\ d^r_{j,\mathbf{n}} = \langle \mathbf{f}, \psi^r_{j,\mathbf{n}} \rangle \end{cases} . \tag{2.16}$$

The created object $\mathbf{W}$ aims to collect all the wavelet coefficients and to store them in distinct areas. Two types of coefficients actually exist and one can distinguish the approximation coefficients $a_{J_0,\mathbf{n}}$, being the result of an inner product between $\mathbf{f}$ and ϕ functions, and the details coefficients, $d^r_{j,\mathbf{n}}$, deduced in the same way from the inner product between $\mathbf{f}$ and ψ functions. The distinction between the two types of coefficients is manifested by considering the frequency content of the signal. While approximation coefficients are assimilated to low frequencies and provide a smoothed representation of the original signal, details coefficients are by opposition related to high frequencies and symbolise the signal's details and edges, which justifies their denomination.

The steps leading to the distribution of a and d coefficients in $\mathbf{W}$ are illustrated at figure 2.3. A first two-dimensional decomposition is operated on the primary image that is composed of the primordial approximation coefficients having a resolution J. In order, a vertical 1D WT decomposition and a horizontal one allow to distribute the coefficients in four main quadrants. Approximation coefficients are gathered in the upper left quadrant while details coefficients are distributed in the three remaining quadrants. A second decomposition can be obtained by realising the same type of operations in the upper left corner of the first decomposition. It is possible to repeat the different steps m times, and it corresponds then to a decomposition of order m. Thus, each decomposition level is computed from the approximation coefficients of the precedent level. As the decompositions progress, the resolution J linked to the coefficients decreases until reaching a threshold value J_0.

Regarding $\mathbf{W}$, the final product of the image decomposition, this matrix has dimensions ($\sqrt{N}$ x $\sqrt{N}$) that are equal to the ones of the original image $\mathbf{f}$. In terms of elements, $\mathbf{W}$ contains N coefficients, compared to $\mathbf{f}$ that is composed of N pixels as a reminder. A schematic view of the coefficient matrix is provided at figure 2.4.

Another interpretation of $\mathbf{W}$ can be made with regard to the resolution segmentation occurring in the image decomposition process. Indeed, this matrix is a multi-resolution frame

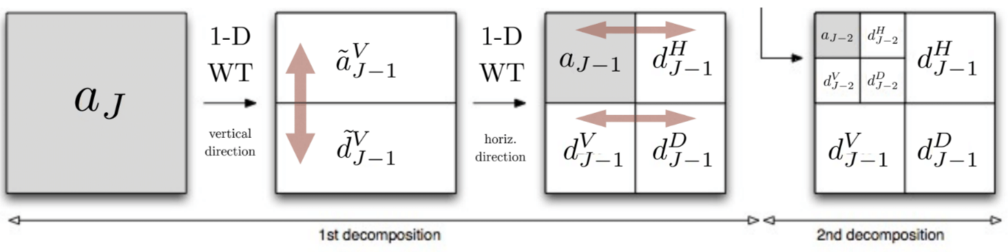


Figure 2.3: Storage of the wavelet coefficients in sub-zones produced by the DWT. From [3].

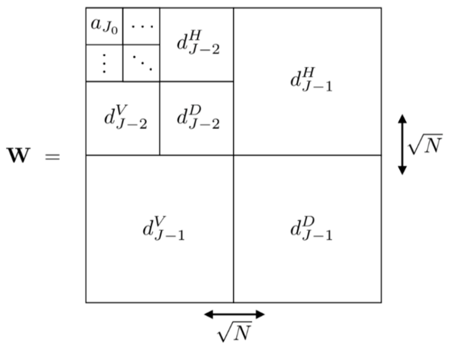


Figure 2.4: Matrix **W** obtained after storing the wavelet coefficients of an image. From [3].

gathering sub-images formed by the wavelet coefficients and the DWT-related computation. This visual description of the image components is proposed at the figure 2.5. In this case, the index m refers to the decomposition order as mentioned before, for which a specific image resolution is associated to each value of its value. The most upper left corner of the new representation contains the approximation image while other parts are composed of image details. In summary, the further the sub-images are from the upper left corner, the finer the resolution.

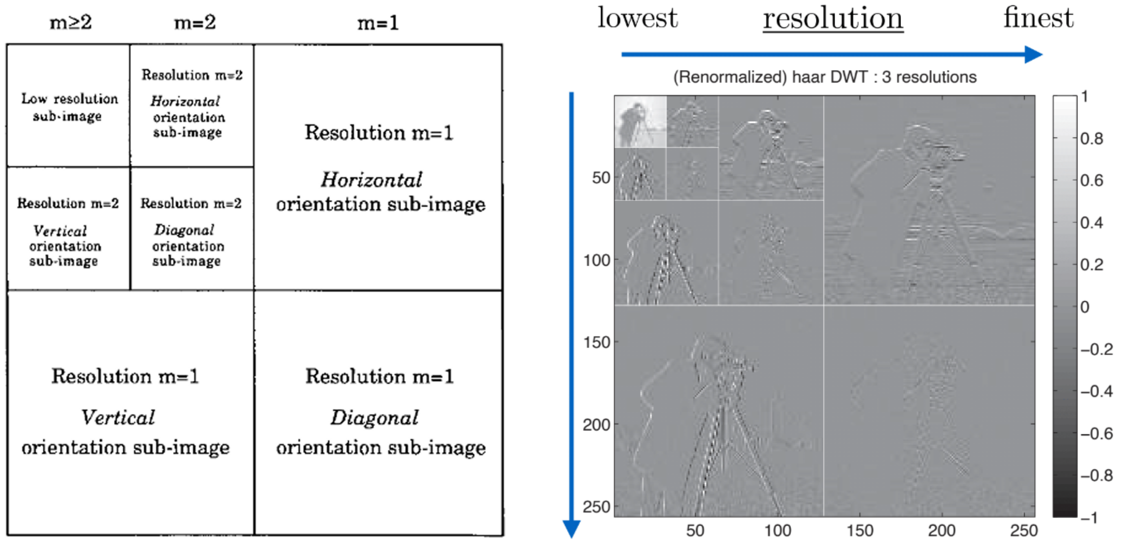

m≥2
m=2
m=1
Low resolution sub-image
Resolution m=2 Horizontal orientation sub-image
Resolution m=1 Horizontal orientation sub-image
Resolution m=2 Vertical orientation sub-image
Resolution m=2 Diagonal orientation sub-image
Resolution m=1 Vertical orientation sub-image
Resolution m=1 Diagonal orientation sub-image
lowest
resolution
finest
(Renormalized) haar DWT : 3 resolutions

Figure 2.5: Multi-resolution representation inferred when performing a DWT. On the left: equivalent repartition than the ones presented in figures 2.3 and 2.4 complemented by resolution values. From [10]. On the right: visualisation of the image decomposition. From [3].

2.4 IHT reconstruction algorithm - Mathematical description

The classical image inpainting problem can be solved by the IHT technique, the latter involving in the present situation, among other existing possibilities, the wavelet theory concepts summarised in the previous section. By choosing an appropriate wavelet basis, a set of wavelet coefficients, determined in connection with the analysed damaged image, is integrated into the reconstruction process and submitted to the action of a Hard Thresholding operator. The threshold that defines this operator is then responsible for the classification of the different wavelet coefficients. As a result, a selection is carried out between the coefficients that are removed from the algorithm, because of their low values that are assimilated to the holes and discontinuities found in the damaged images, and the ones that are kept thanks to their relatively high values. These high value coefficients are the ones that will therefore contribute to filling the missing parts that initially compose the incomplete images. After sorting the wavelet coefficients, an iterative and converging process, implemented in the algorithm responsible of the image reconstruction, is finally performed in order to find a solution.

This part of the state of the art chapter aims to go through the steps followed by the IHT reconstruction algorithm. A first theoretical view inspired by [11] exposes the concept of regularisation and is complemented by a second more practical approach describing the way the reconstruction process is numerically interpreted.

2.4.1 Theoretical approach

A rewriting of the inpainting problem, encompassing both elements established in equations 2.1 and 2.2, is

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{x} + \mathbf{q} \tag{2.17}$$

with $\mathbf{y}$ the observation, $\boldsymbol{\Phi}$ a measurement matrix that is comparable to the mathematical object $\mathbf{A}$ described in section 2.2 and that is constructed on the basis of the mask applied to the image, $\mathbf{x}$ the complete image and $\mathbf{q}$ an extra noise that is still chosen as being equal to zero. The solution of this additional formulation is inferred from a regularisation method, that minimises the norm of a signal approximating the one of interest. This regularisation implies the use of a basis $\boldsymbol{\Psi} = (\psi_m)_m$ (ψ_m are atoms as a reminder) that creates a sparse set of coefficients (s^*_m) and solves

$$\boldsymbol{s^*} \in \underset{\mathbf{s}}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{y} - \boldsymbol{\Phi\Psi}\mathbf{s}||^2 + \lambda \mathbf{J}(\mathbf{s}) \tag{2.18}$$

where $\boldsymbol{s^*}$ is a wavelet coefficient vector conserving the atoms that are selected in the regularisation process, $\boldsymbol{\Psi}\mathbf{s}$ is the signal that must be reconstructed for which the decomposition is equivalent to equation 2.5 and λ is a threshold embedded in hard thresholding calculations (see section 2.4.2).

In the present case, $\boldsymbol{\Psi}$ is chosen as a basis containing Translation Invariant Wavelets (TIWs) that seem to constitute a relevant choice for sparse regularisation. Indeed, the translational invariance aspect of this particular type of base implies that the processed signal is, by some means, independent of the time component which is generally associated with it. Moreover, working in a such invariant frame favours the reduction of artifacts usually found in the context of image restoration. The reason is that the thresholding operator responsible for the image reconstruction (that will be exposed in the next section), also becomes translation invariant [12].

Other possibilities actually exist for bases choices, such as orthogonal bases holding orthogonal wavelets for instance. However, orthogonality properties offer poorer regularisation due to their absence of translation invariance, which justifies the selection of TIWs.

Regarding the quantity $\mathbf{J(s)}$, it is defined with the following reasoning. The objective is here to find the sparsest vector $\mathbf{x}$ possible that verifies $\mathbf{y} = \boldsymbol{\Phi}\mathbf{x}$ (noiseless version of equation 2.17). As $\mathbf{q} = \mathbf{0}$, the current problem can be compared to a l_0 minimisation problem, the latter taking the form

$$min||\mathbf{z}||_0 \quad s.t. \quad \mathbf{y} = \boldsymbol{\Phi}\mathbf{z} \tag{2.19}$$

where $\mathbf{z}$ is a K-sparse approximation that most closely matches x. However, there is no single algorithm that is capable to find a solution to this problem for any combinations of $\mathbf{y}$ and $\boldsymbol{\Phi}$. In order to work around this issue, the $l1$ minimisation replaces the l_0 one and a solution is now coming from

$$min||\mathbf{z}||_1 \quad s.t. \quad \mathbf{y} = \boldsymbol{\Phi}\mathbf{z}. \tag{2.20}$$

$\mathbf{J(s)}$, called the l_1 sparsity prior, is then inspired by this l_1 minimisation and normalises the wavelet coefficients extracted from $\mathbf{s}$, a K-sparse vector of dimension N $\gg$ K. The usual definition of $\mathbf{J(s)}$ is

$$\mathbf{J(s)} = ||s_m||_1 = \sum_m |s_m|. \tag{2.21}$$

Once the regularisation problem is solved, the inverse of the corresponding DWT defined in section 2.3.2.2 ($DWT^{-1} \rightarrow \mathbf{W} : \mathbf{f}$), is applied to $\boldsymbol{s}^*$ and the reconstructed image is thus obtained.

2.4.2 Practical approach

In practice, the reconstruction algorithm successively performs a hard thresholding, with a decaying value of the associated threshold, and a gradient descent step. The applied hard thresholding function $h_T(x)$ is defined by

$$h_T(x) = \begin{cases} x & \text{if} \quad |x| > T \\ 0 & \text{if} \quad |x| < T \end{cases} \tag{2.22}$$

where T is the threshold value. The schematic representation of this function is provided at figure 2.6.

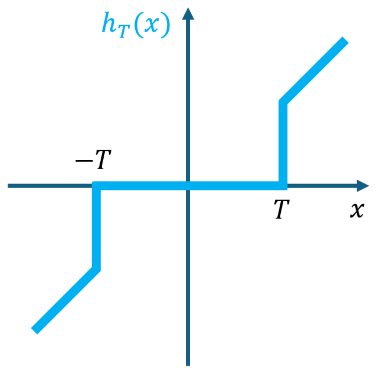


Figure 2.6: Hard thresholding function. Created on the basis of information found in [4].

An operator version of $h_T(x)$ is

$$H_T(\mathbf{f}) = h_T(\boldsymbol{s_f}) \tag{2.23}$$

where $\boldsymbol{s_f}$ is the vector gathering the wavelet coefficients that are linked to the decomposition of the processed image $\mathbf{f}$ obtained from a DWT operation. In a wavelet basis $\boldsymbol{\Psi}$ (the one made of TIWs as mentioned earlier), the hard thresholding operator can be re-expressed as

$$H_T^{\boldsymbol{\Psi}}(\mathbf{f}) = DWT^{-1} \circ H_T \circ DWT \tag{2.24}$$

which justifies the last affirmation provided in the precedent section. A generation of a series of images $\mathbf{f}_l$ (sparse signals in the TIW basis) is made possible by proceeding to

$$\mathbf{f_{l+1}} = H_{\lambda_l}^{\boldsymbol{\Psi}} \left(\mathbf{f_l} + \rho \boldsymbol{\Phi^T} (\mathbf{y} - \boldsymbol{\Phi f_l}) \right) \tag{2.25}$$

where an image $\mathbf{f_{l+1}}$ is formed thanks to the one that directly precedes it $\mathbf{f_l}$. In this formulation, the argument of $H_{\lambda_l}^{\boldsymbol{\Psi}}$ corresponds to the gradient descent step. A change of variable is realised such that λ_l replaces the T variable defined above. As for the ρ factor, this component aims to weight, and somehow adjust, the difference between the observation and the signal that needs to be restored (that is to say, $(\mathbf{y} - \boldsymbol{\Phi f_l})$), in order to find which combination is more likely to match the reconstruction.

Based on previous considerations, the parameter λ_l that is present in equation 2.25 is the threshold connected to the hard thresholding operator. As one wants the signal $\mathbf{s}$ to be as sparse as possible (i.e. a K-sparse signal, in the same manner as in equation 2.5),

λ_l can be viewed as the Kth largest value in expression 2.25, at each iteration. Another possibility is that the value of λ_l can be arbitrarily fixed, and this option will also be tested in the images reconstructions exposed later in this work. Regarding the iterative process of the algorithm, the initial threshold value λ_0, corresponding to the largest threshold value used during the different iterations, will make it possible to threshold the initial image $\mathbf{f_0}$. Finally, by using decreasing threshold values (λ_l), for a determined number of iterations (NOI), the algorithm converges towards a solution.

2.5 PSF implementation

Later in this work, reconstruction methods explained before will be applied on images directly produced by an imager designed in laboratory. Such instrument always implies some images imperfections, that can be caused by the noises found in the experimental surroundings or by the instrumental components themselves (optical aberrations). In order to correct the generated images as much as possible, different techniques exist and one of them consists of carrying out an image deconvolution, on which this section is focused.

More accurately, the idea here is to study the deblurring of an image by knowing the PSF of the optical instrument through which it was acquired. Unwanted noise coming from the instrumental environment is this way subtracted from the image, increasing then the data readability. The deconvolution algorithm involved in the image processing that is discussed here is actually built on similar methods than the ones covered in the IHT, and it notably uses sparse properties of images as previously.

In this section, an overview of what is a PSF and what are its properties is first introduced and is based on information found in [13]. A theoretical approach of the deconvolution problem is then developed and leads to a third part that justifies how the theory is adapted to implement in practice the deconvolution method in numerical codes.

2.5.1 PSF basics

The PSF is a 2D distribution of light representing the way an optical system perceives the transmission of a luminous point source. This distribution varies from one instrument to another and depends on a series of factors that characterises the experimental environment in which images are acquired. Indeed, the light propagating from the object plane (i.e. the observed scene) to the image plane (also called the focal plane, i.e. the detector area) is perturbed when passing through optical elements (lenses, mirrors,...) due to reflection and refraction mainly.

Every optical device is submitted to a certain amount of aberrations that notably manifest themselves in the PSF. Being of various natures ((mono)chromatic, spherical, distorsion,...), aberrations effects actually produce some repercussions on the propagation of light, and their combination induces in particular a deviation from the paraxial ray trace defined by the optics. Related variations are then more or less important according to several parameters, the latter being for instance the optics shape and geometry, the wavelength of light interacting with the surrounding medium, or even the aperture size and the focal length of the optics.

As a result, the image of an observed point-shaped object is distorted, with a specific pattern that is representative of the modifications of light encountered through the optical elements that form the image. The PSF, which is a spatial arrangement of the light intensity, is then also impacted by these changes. However, even if the PSF pattern can slightly differ with the optics configuration, its usual shape fits with the classical Airy pattern. The typical PSF spatial distribution is illustrated at figure 2.7.

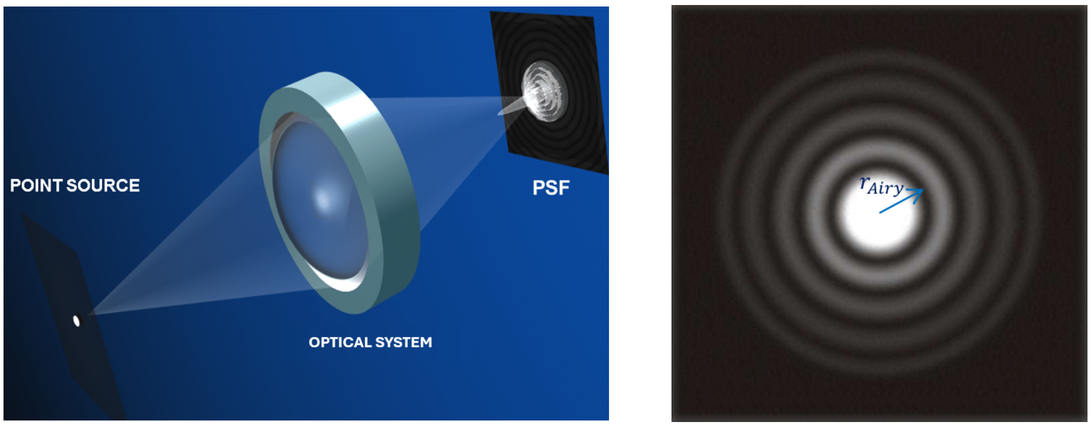


Figure 2.7: Spatial distribution of a PSF. On the left: optical configuration. On the right: Airy pattern and the corresponding Airy radius. From [13].

The Airy pattern has a diffractive configuration that comprises two main structures: a bright disk and a set of external concentric rings whose centre is the one of the disk. These patterns of light intensity maxima are all separated by intermediate dark rings, the latter constituting by opposition regions of light intensity minima. Most of the light intensity is in fact contained in the portion of space delimited by the central disk, having a radius

$$r_{Airy} = 1.22\,\frac{f\lambda}{D} = 1.22\,F_{\#}\lambda \tag{2.26}$$

where f is the focal length and D the pupil diameter of the optical system, λ the wavelength of the incoming light and $F_{\#} = \frac{f}{D}$ is named the F-number, a quantity describing the diffraction effect (Airy pattern) perceived on the focal plane.

The determination of the light intensity distribution deduced from the diffracting aspect of the Airy pattern leads to the definition of the Rayleigh criterion, a useful indicator for both spatial and angular resolutions. While the spatial resolution is expressed by 2.26 ($\mathit{\Delta} l \approx r_{Airy}$), the angular resolution θ is known as

$$sin\theta \approx \theta \approx 1.22\frac{\lambda}{D}. \tag{2.27}$$

In this context, the Rayleigh criterion states that two points present in an image, Q and Q', are said to be resolved if the predominant maximum in the light intensity distribution of Q (respectively Q') is not closer than the first minimum of Q' (respectively Q). On the contrary, the two points are unresolved if the condition is not respected. The minimum distance at which Q and Q' are just resolved is denoted σ. Figure 2.8 schematises the different situations.

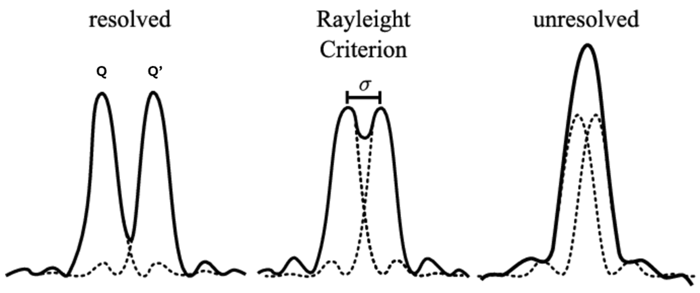


Figure 2.8: Rayleigh criterion. Adapted from [14].

As previously mentioned, another interest of knowing the PSF lies in the resolution of the deconvolution problem. This problem takes place when the image obtained through an optical device is not clear due to the convolution between the light intensity distribution of the observed object and the associated PSF. A visual representation of the convolution is provided at figure 2.9. The way in which image deconvolution is carried out in this work is developed in the two next sections.

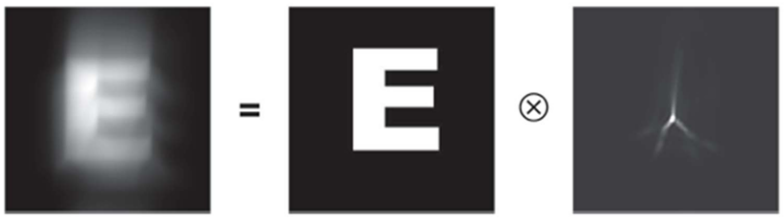


Figure 2.9: Blurred image resulting from the convolution of an original scene with a PSF. From [13].

A last little note concerning the PSF is that it is generally accompanied by the notions of Optical Transfer Function (OTF), a complex function designated as the FT of the PSF, and the Modulation Transfer Function (MTF), which is the modulus of the OTF. This OTF will be particularly useful in the numerical implementation of the deconvolution operation.

2.5.2 Theoretical approach

The configuration displayed at figure 2.9 can be mathematically interpreted as

$$\mathbf{f} = \mathbf{h} * \mathbf{f}_{\text{pure}} = \mathbf{H}\mathbf{f}_{\text{pure}} \tag{2.28}$$

with $\mathbf{f}$ the undeconvolved image, $\mathbf{h}$ the impulse response of the image acquisition system (that is to say, the PSF), $*$ the convolution operator, $\mathbf{f}_{\text{pure}}$ the pure deconvolved image that must be retrieved and $\mathbf{H}$ a matrix symbolising the application of $\mathbf{h}$ on $\mathbf{f}_{\text{pure}}$.

Solving the equation 2.28 relies thus on the determination of the above matrix $\mathbf{H}$, which is in reality a circulant matrix (CM). This uncommon type of matrix is defined as a square matrix whose transition from one line to another is done by circular permutation of its elements. As an example, if one considers a vector $\mathbf{b}$ that holds m components, the resulting CM produced with these components has the same general form as the one of equation 2.30

$$\mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \tag{2.29}$$

$$\mathbf{H} = \begin{pmatrix} b_1 & b_m & \cdots & b_2 \\ b_2 & b_1 & \cdots & b_3 \\ \vdots & \ddots & \ddots & \vdots \\ b_m & \cdots & b_2 & b_1 \end{pmatrix}. \tag{2.30}$$

With the intention of visualising the conventional form of a CM, the figure 2.10 exemplifies how an original random matrix can be turned into a CM one. In this example, each of the initial matrix components was chosen to be unique compared to the other ones with a view of differentiating them easily. An existing and online available Python function allows to construct a CM based on a determined array [15]. The function selects first the (1,1) matrix element in the upper right corner, and then goes in the opposite direction to that of usual reading, from right to left starting from the last line to the first one as illustrated. The first line of the CM is then composed of the signal's elements in the precise order they have been previously classified in a 1D array. The following lines are just circular permutations of those which directly precede them. The final matrix has $p^2 \times p^2$ dimensions if the original one presents p lines and p columns.

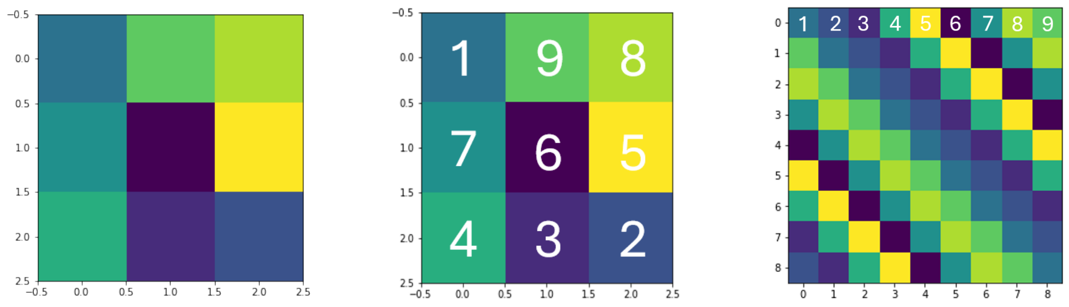


Figure 2.10: CM generation based on a 2D signal rearranged into a vector. On the left: visualisation of the original matrix (of dimensions 3 x 3 in this case) corresponding to the initial 2D signal. In the centre: order in which the matrix elements are selected and stored in a 1D vector by the generating numerical function in order to form the expected CM. On the right: resulting CM of dimensions 9 x 9 obtained from the original matrix.

2.5.3 Practical approach

In the previous section, $\mathbf{H}$ is constructed on the basis of a 1D signal, in which are stored matrix elements that are reorganised in the produced CM. However, this approach is not the one expected when processing 2D signals such as images, and the concept of CM must be extended to the one of block-circulant matrix (BCM). To do this, the vector $\mathbf{b}$ of expression 2.29 is replaced by a new vector $\mathbf{r}$ that is also composed of m elements, at the exception that each of these elements (r_i ; $1 \leq i \leq m$) are now of length n, compared to the previous ones (b_i ; $1 \leq i \leq m$) that were of length 1 (scalar matrix components). $\mathbf{r}$ is thus a so-called block vector, that gathers m vectors of length n. Thanks to that, all pixels contents included in an image having m lines and n columns can consequently be stored in $\mathbf{r}$. In practice, the $m \times n$ image is the PSF acquired from the optical device that aims to observe the object of interest. Each components of $\mathbf{r}$ (r_i) are this way interpreted as the content linked to a precise PSF pixels line

$$\mathbf{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix}. \tag{2.31}$$

Thanks to this new formulation, the matrix $\mathbf{H}$ obtained hereafter (equation 2.32) is a BCM. All of its components *H(*r_i*)* are in consequence blocks which directly depends on the elements found in $\mathbf{r}$. As one can see, this is the arrangement of the different blocks that is circulant, not the blocks themselves. Permutations of blocks follows the same rules as the ones exposed in the preceding section

$$\mathbf{H} = \begin{pmatrix} H(r_1) & H(r_m) & \cdots & H(r_2) \\ H(r_2) & H(r_1) & \cdots & H(r_3) \\ \vdots & \ddots & \ddots & \vdots \\ H(r_m) & \cdots & H(r_2) & H(r_1) \end{pmatrix}. \tag{2.32}$$

The BCM presented here is, as a matter of fact, diagonalisable. In other terms, one can write $\mathbf{H} = \mathbf{F}^*\mathbf{D}\mathbf{F}$, where $\mathbf{F}$ is a Fourier matrix ($\mathbf{F}^*$ is its complex congugate form) and $\mathbf{D}$ is a matrix that is circulant in diagonal blocks. They are respectively defined by

$$\begin{cases} \mathbf{F} = \mathbf{F}_m\mathbf{F}_n \\ \mathbf{D} = diag(\mathbf{F}\mathbf{r}) \end{cases} \tag{2.33}$$

where $\mathbf{F}_m$ and $\mathbf{F}_n$ are Fourier matrices of order m and n. These objects are expressed through the use of complex numbers $\omega_n = e^{2i\pi/n}$ and are written

$$\mathbf{F}_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 1 & \omega_n & \cdots & \omega_n^{n-1} \\ \vdots & \ddots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)^2} \end{pmatrix}. \tag{2.34}$$

The diagonalisation of the BCM is then expanded as indicated in equation 2.35. This is this last shape of $\mathbf{H}$ that will be implemented in the numerical reconstruction codes.

$$\mathbf{H} = \begin{pmatrix} H(r_1) & \cdots & H(r_2) \\ \vdots & \ddots & \vdots \\ H(r_m) & \cdots & H(r_1) \end{pmatrix} = \begin{pmatrix} \mathbf{F}_n^*\mathbf{D}_1\mathbf{F}_n & \cdots & \mathbf{F}_n^*\mathbf{D}_2\mathbf{F}_n \\ \vdots & \ddots & \vdots \\ \mathbf{F}_n^*\mathbf{D}_m\mathbf{F}_n & \cdots & \mathbf{F}_n^*\mathbf{D}_1\mathbf{F}_n \end{pmatrix} = \mathbf{F}^*\mathbf{D}\mathbf{F} \tag{2.35}$$

The above development is inspired by explanations coming from [16] and leads to the determination of the predicted shape of the BCM $\mathbf{H}$. Nevertheless, a last problem persists regarding the resolution of equation 2.28. Indeed, $\mathbf{H}$ is usually badly conditioned and/or not invertible (i.e. $\mathbf{H}\mathbf{H}^{-1} \neq \mathbb{I}$, with $\mathbb{I}$ the identity matrix) and the relation 2.28 cannot be directly inverted with classical matrix algebra. To overcome this difficulty, one must perform the same kind of regularisation as the one presented in 2.4.1.

In this context, the image $\mathbf{f}_{pure}$ (equation 2.28) is sparse in a wavelet basis $\boldsymbol{\Psi}$ such as

$$\mathbf{f}_{\text{pure}} = \boldsymbol{\Psi}\mathbf{s} \tag{2.36}$$

in which $\boldsymbol{\Psi}$ is a TIW basis as the one mentioned in section 2.4.1 and $\mathbf{s}$ is a vector containing wavelet coefficients of the analysed image. $\boldsymbol{\Psi}\mathbf{s}$ is then the signal that must be reconstructed. Hence, a possible deconvolution program is to take, among all the images $\mathbf{g} = \boldsymbol{\Psi}\mathbf{s}$ such as $\mathbf{Hg}$ is close to $\mathbf{f}$ (equation 2.28), the one which has a small sparsity. By taking a positive threshold λ (i.e. $\lambda > 0$), an expression highly similar to equation 2.18 is determined

$$\boldsymbol{s}^* \in \underset{\mathbf{s}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{f} - \mathbf{H}\boldsymbol{\Psi}\mathbf{s}||^2 + \lambda\mathbf{J}(\mathbf{s}), \tag{2.37}$$

at the exception that the observation $\mathbf{y}$ is replaced by the undeconvolved image $\mathbf{f}$, and that the mask $\boldsymbol{\Phi}$ is substituted by the BCM $\mathbf{H}$. An analogous reasoning that the one described in 2.4, involving a decaying value of λ, allows the algorithm to converge towards a solution.

Chapter 3

Digital image processing

3.1 Introduction

Starting from scratch, with no knowledge of Python code, the very first challenging objective was to learn how this programming language works, and particularly in terms of image processing. On that purpose, a series of notebooks available online were consulted. These notebooks are included in the Numerical tours, which consist of visual applications (image manipulation, plots,..) of various images-related problems. The most used one was the *Inpainting using Sparse Regularization* notebook [4], dedicated to the learning of image reconstruction using the IHT algorithm exposed in section 2.4. A second one whose content is originally coded in Matlab, the *Image Deconvolution using Sparse Regularization* notebook [5], was adapted and re-written in Python in the framework of this master thesis to perform images deconvolution for which the theoretical concepts are presented in section 2.5.

This chapter focuses on all the methodology and techniques that are followed and involved in the digital image processing covered in this work. Results are then presented and discussed for each of the correlated methodology sections. First, the contribution of a technique called zero padding (ZP) is studied and aims to allow the reconstruction of images of any dimensions. Some images restorations of a scene, acquired by an imager that is described beforehand in this same chapter, are then performed to introduce some useful information that will be used throughout the rest of the work, notably in the following section for which the objective is to test and optimise the different parameters of the IHT algorithm. How certain adjustments can be applied to the obtained results is also covered and the chapter is then completed with an approach related to image deconvolution.

3.2 Laboratory-generated data

Most of the data processed in this work were obtained in the frame of the master thesis of Émile Ruwet, dedicated to the improvement of a CS imager designed and tested in laboratory. This section introduces how the different images that are needed for the reconstructions are produced, and is focused on two main aspects. First, a brief presentation of the imaging device is exposed in order to provide an overview of the optical setup and of its components.

A second part develops then the process linked to the generation of binary masks. This notion is then extended to the so-called eroded masks, the latter being formed thanks to a morphological erosion operation. Almost all the explanations gathered in this section rely on information derived from [17].

3.2.1 Description of the optical system

The imager that produces both the (un)damaged scenes reconstructed in this thesis and the masks applied on these (in)complete images is especially designed for inpainting applications and follows the same rules as classical CS optical systems. The structure of this imaging device comprises four elements that allow to obtain the expected data by following a succession of well-defined steps. First, a collecting lens gathers the light emanating from the scene in front of which it is placed. The scene is typically a previously chosen image that is printed on a piece of paper having adequate dimensions and which is maintained by an adapted support. The light is then focused and propagates until reaching a Digital Micromirror Device (DMD) that constitutes the second component.

A DMD is actually an array containing, as it name indicates, many micromirrors that can be tilted in two different configurations with a corresponding tilt angle of ± 12° (compared to the normal of the DMD surface). The first one, the ON configuration (tilt angle of 12°), corresponds to the positioning of a mirror that is set as such manner that the light reaching it is reflected towards the next optical elements. On the contrary, the OFF configuration (tilt angle of -12°) consists of orienting a mirror in a position that sends part of the overall perceived light outside the optical system and which will therefore not be taken into account during data acquisitions. As a consequence, these two mirrors arrangements offer the possibility to create either complete images of the observed scene (all micromirrors set in the ON configuration) or fragmented images in the context of damaged scenes (with this time part of the micromirrors set in the OFF configuration). In the present case, the DMD model embedded in the setup harbours 1024×768 micromirrors, each of them being remotely controlled by using an appropriate software generating the desired patterns. The patterns are implemented in the DMD thanks to an external electronic controller directly connected to DMD and that automatically switches the mirrors in ON or OFF directions, with a frequency of about tens of thousands of Hertz.

Once the incoming light has encountered the DMD surface, the remaining contribution reflected by the ON-state micromirrors is then directed towards a condensing lens, the third component of the imager. This lens brings afterwards the modulated image to a camera detector constituting the fourth and last element of the setup. The detector is a monochromatic CMOS (Complementary Metal Oxide Semiconductor) one, having a colour depth of 8, a total amount of 1280×1024 pixels and operating in the visible part of the electromagnetic spectrum. The particularity of CMOS devices is that each of their pixels is accompanied by its own transistors and readout amplifier, which considerably increases the cadence of data production and there is therefore no need to complement them with an additional shutter [6].

In order to have a better understanding of how the different optical elements are arranged in the final assembly, a visualisation of the latter is provided at figure 3.1. In reality, this representation is not exactly the final version of the instrument. Indeed, the entire setup, excepting the observed scene which is lit by an external lamp, was covered by using a black housing, with a view of decreasing the surrounding noises that could affect the data quality.

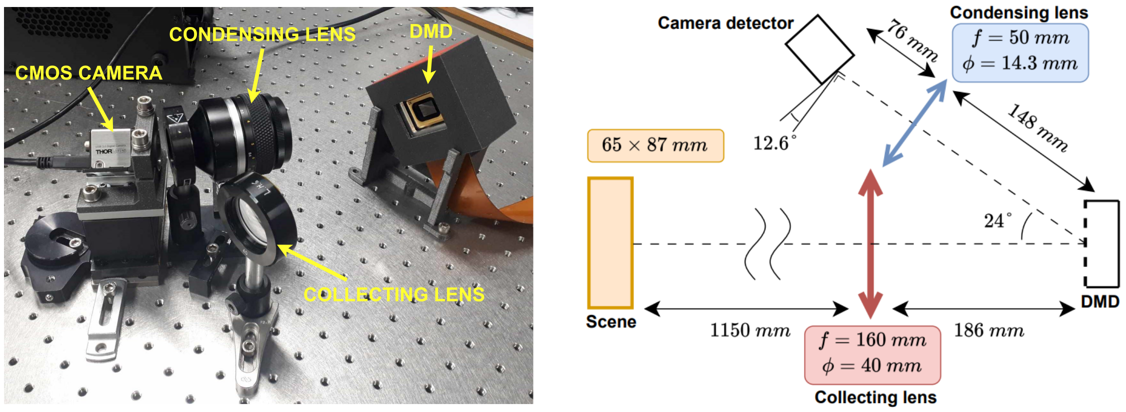


Figure 3.1: Representation of the optical setup. On the left: picture of the optical components. On the right: schematic view of the imager. Adapted from [17].

3.2.2 Production of binary and eroded masks

Thanks to the CS imager presented just before, series of mask patterns were created depending on the requirements imposed when testing images reconstructions by inpainting. The primary version of those masks, corresponding to the raw result directly seen through the optical setup owing to the use of the DMD, has grayscale variations, which must be corrected as much as possible. The reason behind this statement is that the mask structure must be formed on the basis of a binary pattern, in order to obtain something that is similar to what was illustrated at the figure 2.1.

A possible way to correct the non-uniformity of this initial mask version is to apply on it an Otsu's binarisation method. In a few words, this method probes the initial intensity distribution contained in the analysed image, in order to find an appropriate threshold. This threshold controls then each of the image's pixels content and apply on it the expected correction. Basically, the pixels values that are lower than the threshold one are set to 0 while larger values are set to 1, creating in this manner the binary mask of interest. An example of the effect caused by this Otsu's method is provided at figure 3.2.

Even if this binarisation operation considerably improves the quality of the produced masks, some adjustments can still be made to optimise the inpainting restorations results. Indeed, the binary mask constructed with the Otsu's method creates, when applying it on an observed damaged scene, gray edges that appear around the missing parts of the

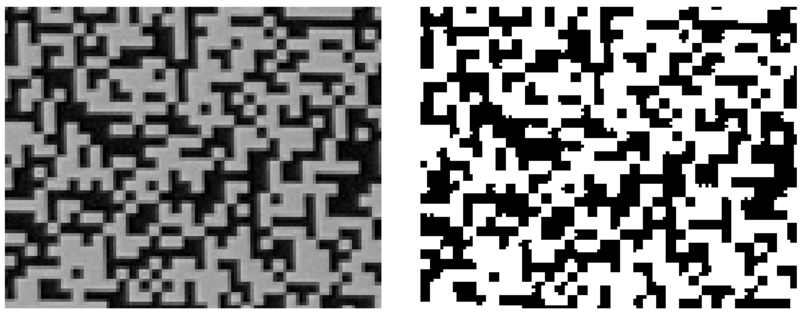


Figure 3.2: Comparison between a mask pattern seen from the optical setup (on the left) and the same area submitted to the Otsu's binarisation method. From [17].

incomplete image. The algorithm will therefore not reconstruct the gray parts as it is only focused on the mask's binary pattern, only composed of 0-value and 1-value pixels from which the gray edges are excluded. This will then inevitably lead to some issues during the reconstructions. To overcome this problem, the mask is submitted to a new technique called morphological erosion aiming to remove the gray edges in the reconstruction process. Morphological erosion involves the use of a matrix called a structuring element that will interact with all of the non-eroded image pixels. More precisely, the structuring element is here a 3×3 matrix as the gray edges a thickness of the order of one pixel on average. Due to these considerations, the principle of the morphological erosion in the present case is thus the following one. The centre of the structuring element goes through all of the mask pixels, and analyses the 8 pixels that surround the current pixel that is matching the structuring element centre at each iteration. The central pixel is then modified such that it corresponds to the minimum value found in its neighbours. As the mask is a binary one, the resulting value is therefore necessarily a 0 or a 1. The typical result of the morphological erosion of a binary mask is exemplified at the figure 3.3.

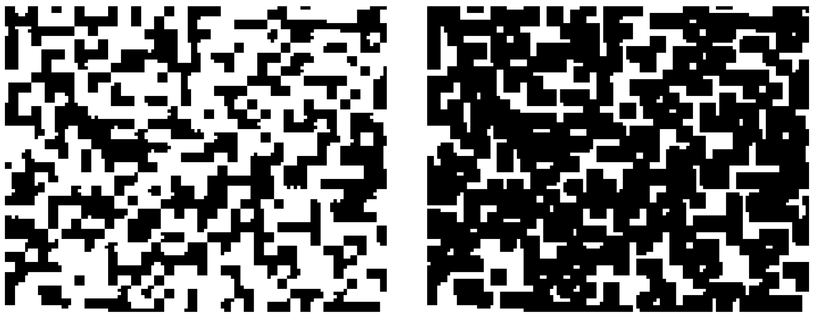


Figure 3.3: Comparison between non-eroded (on the left) and eroded (ont the right) patterns for a same mask portion. From [17].

One can observe that the erosion correction increases the mask's filling ratio (FR), the latter being defined by the percentage of white (1-value) and black (0-value) pixels filling the whole image of interest. In the following parts of this work, a convention is established such that all the FR associated to the masks involved in the different reconstructions are chosen as being equal to the percentage of black areas, and thus to the percentage of pixels having a zero intensity. As an example, the eroded mask illustrated in the previous figure

3.3 has a FR of 70% while the non-eroded mask has a FR of 50%. In order to estimate the impact of the morphological erosion and of the corresponding change of FR on the images reconstructions, a comparative study between non-eroded and eroded masks will be developed later in section 4.1.1.

3.3 Methodology

3.3.1 Zero padding

The very first idea when trying to optimise the inpainting numerical code proposed in [4] was to evaluate if the reconstructions methods were adapted for images having dimensions of any values. However, it appears that the original code is only suitable for square images and it only partially reconstructs rectangular images, which can cause some issues depending on the application. To overcome this potential blocking situation, a technique called zero padding (ZP) was implemented in the early stages of the images processing.

In reality, the ZP method consists of adding lines and/or columns of black pixels (i.e. complete an image with an complementary part for which the pixel intensity is equal to zero) at the edges of a pre-existing image, in order to correct its dimensions. Visually, it is thus equivalent to annex lateral black bands to the preliminary image. The advantage of this manipulation is that it does not affect in any way the content that is initially present in the image that needs to be processed.

As a broad variety of pictures are usually composed of $2^i \times 2^j$ pixels, where i and j are natural numbers, a first possibility is to add enough zero value pixels so that the final square image has its dimensions equal to 2^i if $i > j$ (respectively 2^j if $i < j$). The value of i (respectively j) is therefore adequately chosen to prevent 2^i (respectively 2^j) from being strictly smaller than the height and/or width of the initial rectangular image. The image completion is then performed accordingly. Nevertheless, even if this first approach leads to the production of square images as expected, the addition of pixels can become quite considerable, especially in the case none of the image dimensions is originally equal to a power of 2. A consequence of the ZP operation is then an increase of the image data size, the latter being quantified thanks to the following equation

$$Image\ size\ \textbf{[KB]} = \frac{M \times N \times B}{8 \times 1024}, \tag{3.1}$$

where M and N are the dimensions of the image ($M \times N$ is the corresponding total amount of pixels) and B is the colour depth, introduced in section 2.1, that is expressed in bits [6]. This data growth constitutes actually a drawback in terms of computation time. Indeed, it is quite obvious that the more voluminous the image content, the longer the computation time, as the image restoration algorithm has more information to process. In order to reduce as much as possible this computation time, a second version of the ZP method, hereafter referred as the improved ZP one, was created to minimise the amount of extra pixels. This time, the numerical code compares the image dimensions, and the smallest one is adapted to match the largest one. A new square image is then obtained, the length of one of its sides corresponding to the height or the width of the raw image, depending on whether the latter has a vertical or a horizontal shape respectively.

A visual representation of the initial and the corrected versions of the ZP technique is illustrated at figure 3.4, with the example of an original image of dimensions 297 $\times$ 684. As one can see, the image produced with the primary ZP method has 1024 $\times$ 1024 dimensions while the improved ZP one has 684 $\times$ 684 dimensions. In this representation, axes appearing to the left and bottom of the images directly provide an indication of the number of pixels making up their height and width. All these axes, that are attached to the images being either displayed in the text or in the appendix part of this work where most of the results are gathered, have the same meaning.

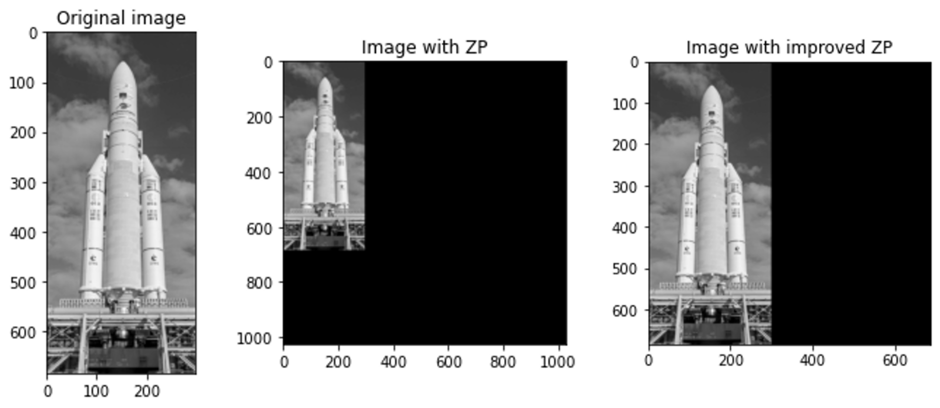


Figure 3.4: Comparison between original image, image with ZP and image with improved ZP.

Another possibility to save time when realising the images restoration would have been to rescale the images corrected by ZP, and limit in this manner the amount of processed data. An image rescaling is notably introduced at the beginning of the code extracted from the Numerical tours' notebook [4] to adjust the data that needs to be reconstructed. However, a rescaling operation inevitably leads to a loss of information, which should, of course, be avoided as much as possible. In fact, rescaling an image means that all the pixels content found in the unmodified image must be redistributed in a smaller grid of pixels, which makes the appearance of the image obtained by rescaling more pixelated. In a way, the rescaling step can thus be assimilated to a merging of pixels. The figure 3.5 shows how a rescaling can actually affect the image quality, with image dimensions that are divided by a factor 10 before carrying out the reverse operation to recover a image with the initial dimensions. A quick look clearly indicates that the final image is blurred compared to the primary one. Even if the scaling factor of 10 was here chosen to be large enough in order to reinforce the effect demonstrated in the example, a smaller scaling factor would also modify, to a lesser extent, the image content. The related loss of information justifies why this method cannot be kept for reconstructions and rescaling is therefore not implemented in the last version of the tested numerical algorithms for this reason.

With a view to test the two versions of the ZP technique and determine which one is

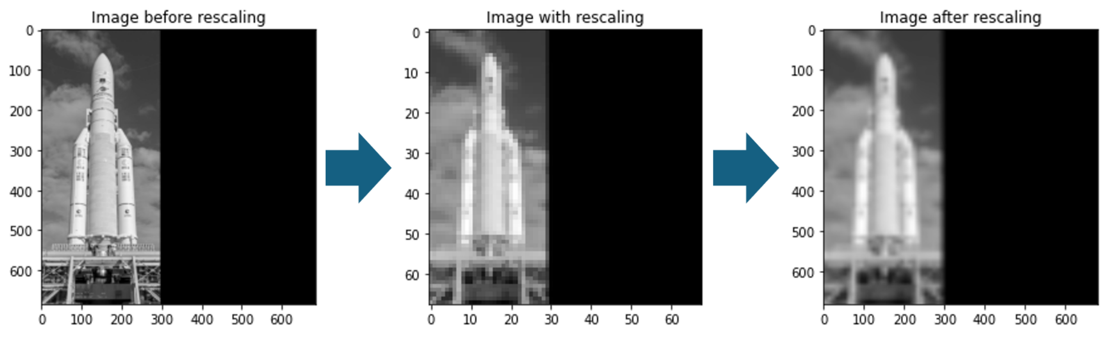


Figure 3.5: Effect of rescaling on an image with improved ZP.

the most suitable for the images reconstructions, different steps were followed in a precise order until obtaining pertinent results. This succession of manipulations is described in the figure 3.6. First, the image is completed by using either the initial or the improved type of ZP, and is after that masked by applying a randomly generated mask having the same FR in both cases and having the same dimensions as the image with ZP. The masked image is then submitted to the restoration algorithm and a corrected version of the ZP image is acquired. Finally, the obtained result is cropped to retrieve an image having the same size as the original one. A Peak Signal-to-Noise Ratio (PSNR) value can then be determined on the basis of the original image and the restored one.

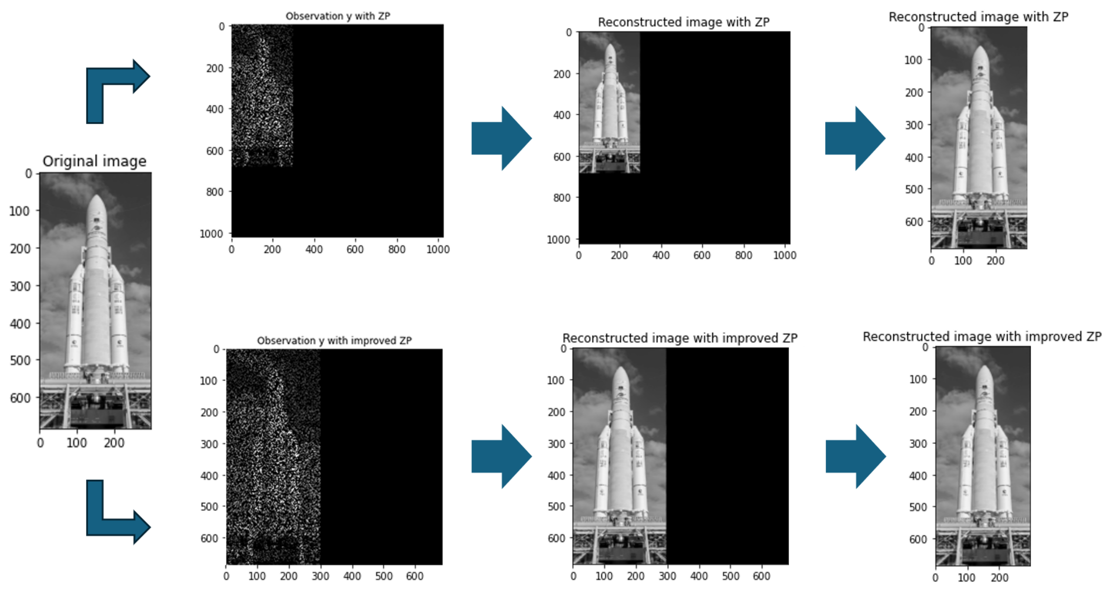

Original image
Observation y with ZP
Reconstructed image with ZP
Reconstructed image with ZP
Observation y with improved ZP
Reconstructed image with improved ZP
Reconstructed image with improved ZP

Figure 3.6: Main steps of images reconstructions involving the ZP technique, for both initial (above) and improved (below) cases.

PSNR is a quantity that represents the ratio between the maximum value that can be found in a signal and the noise that induces some perturbations on this same signal. As the dynamic range of a signal, defined as the ratio between its maximum and minimum values, can be broad, the PSNR is generally expressed through a logarithmic scale using decibel units. A general mathematical expression for PSNR is

$$PSNR = 20\,log_{10}\left(\frac{max(\mathbf{f})}{\sqrt{MSE}}\right), \tag{3.2}$$

where $max(\mathbf{f})$ is the maximum value exisiting in the original image (or signal) $\mathbf{f}$. The Mean Squared Error (MSE) is another quantity that symbolises the noise introduced in the PSNR definition and that can be developed as

$$MSE = \frac{1}{M}\frac{1}{N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}||f_{i,j} - g_{i,j}||^2. \tag{3.3}$$

The MSE can therefore be interpreted as follows. For two images $\mathbf{f}$ and $\mathbf{g}$ having similar dimensions, the MSE takes the square of the difference between every pixel f in $\mathbf{f}$ and the corresponding pixel (that is to say, the one having the same location (i, j)) g in $\mathbf{g}$, sums all the possible values and divides it by the product between the number of rows of pixels of the images M and the number of columns of pixels of the images N. Rows are indexed by i while columns are indexed by j. Here, $\mathbf{f}$ plays the role of the original image and $\mathbf{g}$ is the restored one [18].

In summary, the lower the MSE, the higher the PSNR value and so the better the image reconstruction. Indeed, a low MSE means that on the whole, the difference between two corresponding pixels is small too (it tends towards zero when the pixels contents are almost identical), and thus that there are only slight variations between the two images. As the PSNR value clearly indicates the relation between an original image and its reconstructed version, it will be used through all this work as a reference in order to estimate the quality of all the images restorations.

3.3.2 Reconstructions based on an observed scene

This section is focused on the very first reconstructions performed on an image acquired by the imaging device presented in section 3.2.1. In this preliminary series of tests, the observed image is initially complete, and masks are then numerically applied on it to hide some of the image zones, creating this way missing information that will be reconstructed by the IHT algorithm presented in the appendix section B.1. The idea is here to repeat this restoration process for varying values of the NOI and determine which range of PSNR values can be obtained by comparing the reconstructed images with the original image. In the algorithm, a decaying value of the threshold λ_l described earlier in equation 2.25 is selected. This threshold is hereafter called the λ parameter.

This primary approach of observed scene's reconstructions has two main objectives. Firstly, it aims to study the effect that the NOI value has on the quality of the image restorations. Secondly, a comparison is made between the reconstructions results obtained when masking the observed image either by a mask having a random pattern and being digitally generated, or by another mask having also a random pattern (different from the previous one) but that is this time created thanks to the micromirrors of the imager's

DMD. To avoid any confusion between the two masks in the following, the mask that is digitally obtained is hereafter referred as the random mask, while the second mask being experimentally obtained with the imaging device is hereafter called the lab mask. The figure 3.7 gathers the original image, sometimes also referred as the observed scene or the ground truth (GT), and the two tested masks.

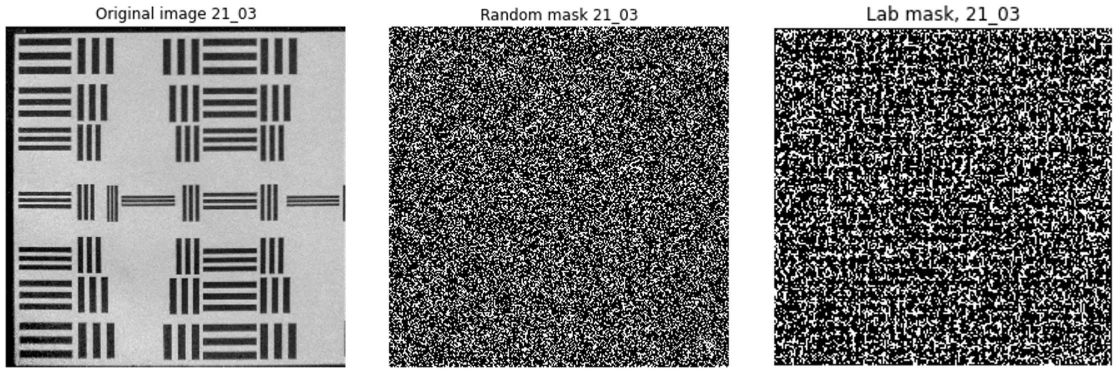


Figure 3.7: Original image of the well structured scene and its related random and lab masks.

As one can see, the observed scene was originally selected because of its regular patterns, from which deviations produced by the algorithm during the reconstructions are easily identifiable visually. This image was generated by the experimental setup on March 21 (21/03). This indication linked to the acquisition date will help to differentiate the scenes throughout the work, and notably in the appendices where all the reconstructed images are gathered. Regarding the masks, their FR were chosen to be equal (70%) so that they can be adequately compared. Nevertheless, the distribution of the two masks' patterns differ from each other. This difference actually influences the results of the image restoration, as it will be presented in section 3.4.2.

3.3.3 Optimisations

The primary objective of this section is to verify if it is possible to optimise the λ parameter by fixing its value, by opposition to the previous section in which the algorithm performed itself an iterative and decreasing resizing of the threshold. Here is the procedure allowing to achieve this goal. Similarly to what was already done before, a random mask is applied on the 21/03 scene. A series of reconstructions is then realised with a fixed value of λ selected arbitrarily for each of the restorations, in an interval going from 0 to 1. All of these operations are also carried out by using a fixed value of the NOI that does not vary between each of the reconstructions. This way, and by computing the PSNR between the initial undamaged image and each of the restored ones, a representation of the PSNR value as a function of the λ parameter can be obtained with a view to check if a precise value of λ maximises the PSNR and thus enhances the images processing results.

After that, this series of steps is also performed on a second scene that is less structured than the 21/03 one. This new GT (acquired on March 27 and therefore called the 27/03 scene) represents actually a satellite scene of the CSL that harbours many varying areas (in terms of their shapes), compared to the first image where there are only straight and redundant patterns. The interest is thus here to check if the regularity of a scene can influence the fixed value of the threshold λ for which the reconstructions are the best. A representation of the 27/03 GT and of the masking pattern involved in the image correction process, that is actually an eroded mask, is provided at the figure 3.8.

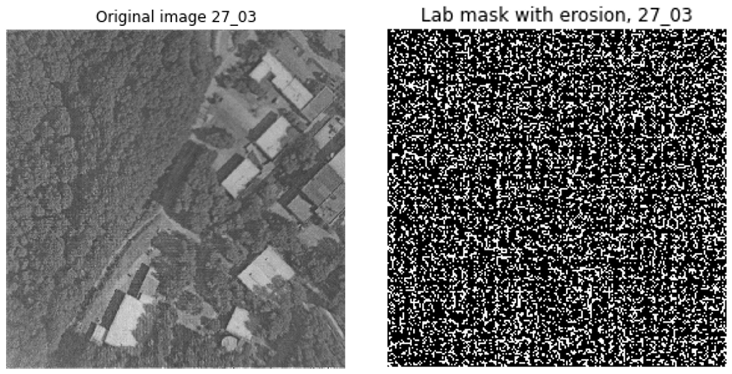


Figure 3.8: Original image of the uneven scene and its related eroded mask.

Besides the determination of an optimised λ parameter, some adjustments and improvements were also implemented in the IHT numerical code, as it was already the case for the ZP technique described above. The first modification consisted in creating a loop that will generate, in an automated way, a series of reconstructions based on the masked image of interest and on various NOI values that are encoded beforehand by the user. Two arrays are also introduced in this same loop, one aiming to store all the reconstructed images in order to facilitate their manipulation in subsequent operations, and the other one gathering all the generated PSNR values so that they can be directly integrated into a graph afterwards. The IHT Python code is then completed by adding a part dedicated to the display of the reconstructions results with their grayscale colorbars. This last improvement is explained in more details in the next section.

3.3.4 Grayscale adjustment

Due to the numerous operations taking place inside the IHT algorithm, the restored images obtained after the reconstructions usually appear clearer than the initial corresponding GT. This can be understood by the fact that the manipulations of images that are carried out by the WTs affect somehow the pixels intensities that are originally contained in the GT. These initial intensities are, knowing that all the images processed in this work have a colour depth equal to 8 bits, distributed according to 256 distinct values, in an interval going from 0 to 255 (as it was already explained in section 2.1). However, the operations realised by the WTs are not restricted to this particular interval and the resulting pixels can thus have intensities that exceed the usual range. As a consequence, the global contrast of the reconstructed image is modified, according to the following expression:

$$C = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \times 100\ \%, \tag{3.4}$$

where C is the image contrast, and I_{max} and I_{min} are respectively the maximum and the minimum value of all the pixels intensities present in the image [13].

Some possibilities exist to correct this change of contrast, so that the reconstructed images can be compared easily to their respective GT. The method followed here is composed of two distinct steps. The first one consists of restricting all the pixels intensities values of the reconstructed image in the expected range. In order to avoid having non-integer values, the second one rounds all the obtained values to the nearest integer number. These two corrections are directly performed by using two classical Python functions.

For the purpose of visualising to which of the grayscale level the images content corresponds, a display added at the end of the IHT code, available at the appendix section B.1, allows to accompany any type of 8-bits image with its grayscale in the form of a colorbar. Every time it is needed, all the following images presented in this work, and especially the ones gathered in the appendices, are complemented with their respective colorbar.

3.3.5 Image deconvolution using PSF

As it was already mentioned in the state of the art chapter, image deconvolution is a process aiming to remove from an image undesired contributions caused by the imperfections of the optics, to make it sharper and smoother. This part of the master thesis actually explores the methodology proposed by the numerical code found in [5] (from which an adapted version is available in the appendix section B.2). The content of this part of the work is thus quite different from what was already covered earlier as it does not include inpainting and IHT reconstructions, and is focused on another kind of inverse problem. The goal is now to carry out an image denoising (a denomination found in [5] that will be reused in the current section) and deconvolution, by applying on an observed scene the FT of predefined kernels, and by making use of the wavelet coefficients of the image of interest as it was presented in section 2.5. Here, two types of initial kernels were tested, namely a gaussian kernel (GK), and a PSF measured in laboratory demonstrating the possibilities offered by real measurements. A third option would have been to test, in a more theoretical manner, the effects caused by a classical Airy pattern, but this case is however not covered in the present work.

The figure 3.9 below illustrates all the components that will play a significant role in the image denoising and deconvolution. At the beginning of the process, one finds two different types of kernels. The first one, that is digitally generated and is initially proposed in [5], is a GK similar to a 2D Gaussian function and for which the spatial extension and general shape were selected in order to mimic as much as possible the ones of measurable PSFs. These PSFs constitute the second sort of analysed kernels, and one of them was especially acquired by the optical setup of section 3.2.1 for the purposes of these image deblurring tests. The production of this PSF was performed by observing a uniformly white scene through the imager with only one of the DMD micromirrors set in the ON configuration, representing in this manner the original point source on the basis of which a PSF is defined. From these two different objects, it is then possible to compute their respective FT, that will in a certain manner replace the Fourier matrices developed in 2.5.3, and will then lead to the production of an operator having the same resulting effect as the one of the BCM **H** previously described. However, it is important to note that the FTs exposed in the figure 3.9 are actually the real parts of the initial FTs, and can therefore be assimilated to the OTF cited in section 2.5.1. These real parts are in fact taken in order to visualise the FTs in a more conventional way, as these mathematical objects are usually made of complex values. A last consideration about these FTs is that they are initially not shifted. In other words, most of their high frequency components are all gathered in the corners of the corresponding images. A shift operation is therefore needed and applied on the initial FTs thanks to a dedicated function, so that all the frequency information is now centred. This corresponds to the final form of the FTs that will be implemented in the denoising operator.

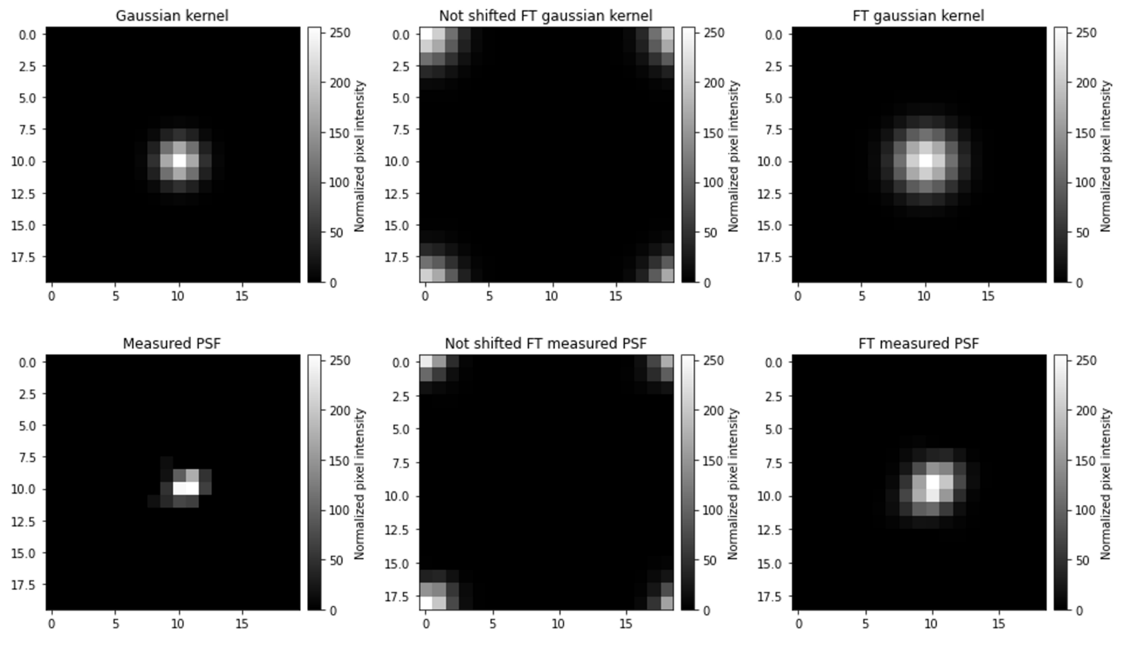


Figure 3.9: Main objects used in the image deconvolution process, in the cases of a GK and of a measured PSF. On the left: kernels. In the middle: not shifted FTs. On the right: FTs.

The original scene submitted to the denoising and deconvolution operations is the one of figure 3.10. This version of the observed scene is slightly inclined due to the structural configuration of the imager and notably to the tilt angle of the DMD's micromirrors, as it is schematised in the figure 3.1. Possible corrections of the visualisation of the scene, leading to images similar to the one exposed at the figure 3.8, are obtained thanks to post-processing techniques. After that, some preliminary tests are first realised with a view to denoise the GT. To do so, the final versions of the FTs are completed with the same ZP

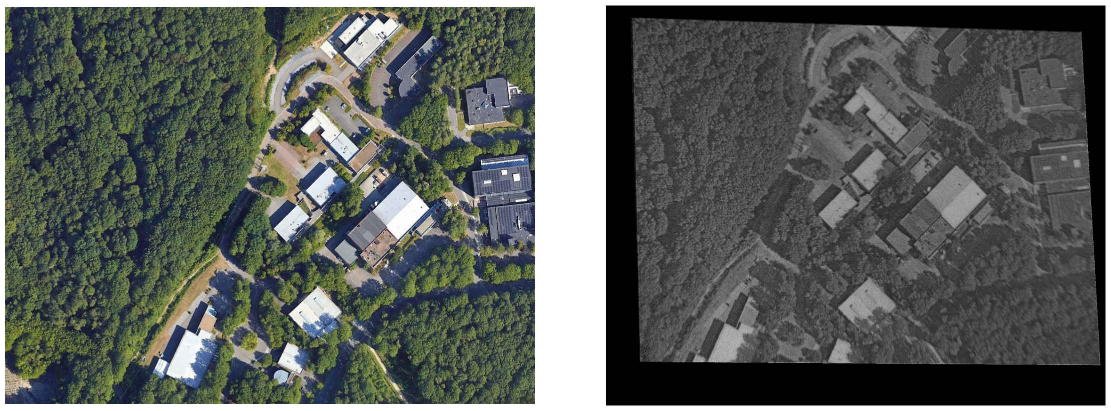


Figure 3.10: Satellite view of the CSL serving as a reference scene (on the left) and its equivalent image (GT) produced by the optical setup with all the DMD mirrors set to the ON-configuration (on the right). Adapted from [17].

technique than the one of the IHT code, so that their dimensions match the ones of the GT, while keeping high frequency information at the centre of the image. This technique is applicable in this case because the pixels intensities found at the edges of the FTs images are negligible (their values tend towards zero), as it can be seen once again in figure 3.9. Hence, adding extra 0-value pixels in each sides of the FTs does not affect the operations significantly.

The results obtained for the GT denoising when applying on it the FTs are available at the figure 3.11. The application of the FT related to the GK seems to properly denoise the entire image. However, the outcome is clearly different for the measured PSF, even if the followed procedure is exactly the same. A potential hypothesis for this unexpected situation is that the FT of the measured PSF contains in reality two intensity peaks, that are not resolved in the figure 3.9 due to optical limitations. This could explain why the FT of the measured PSF seems a little bit stretched and why the resulting denoised image looks like the superposition of two identical images, offset from each other.

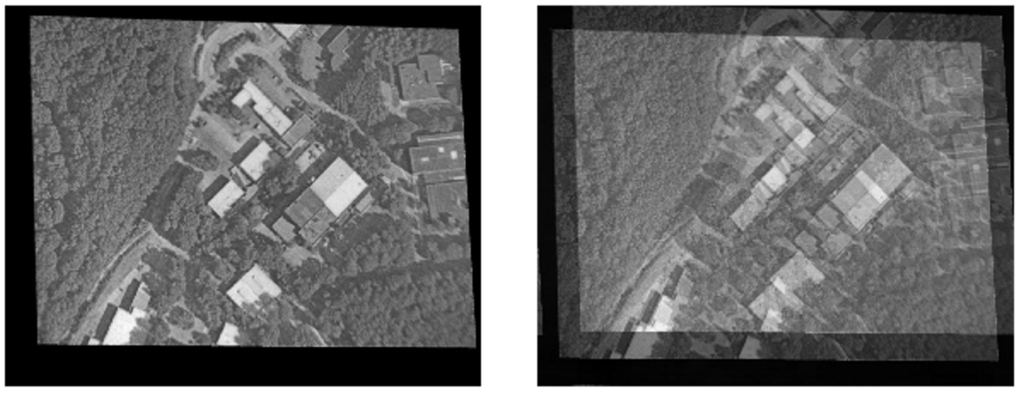


Figure 3.11: Effect of the denoising operation on the GT, from a GK (on the left) and a measured PSF (on the right).

Here is the methodology followed to solve this problem. First, the GT is cropped with arbitrarily chosen dimensions that are smaller than the ones of the complete GT (here, square images were selected). In parallel, the FT of the measured PSF is completed by ZP so it can be applied efficiently on the cropped GT. The position of this cropped GT, compared to the original complete GT, is then adapted by cropping another image area with the same dimensions as before, until the shifted images of figure 3.11 are superposed. This step is therefore repeated several times and corresponds to a trial and error method. The expected superposition is thus obtained by conserving the same FT completed by ZP all along, while modifying the cropped GT. After that, a regularisation of the image's wavelet coefficients is performed, similarly to what was developed at the end of the section 2.5.3. As this last step corresponds to the final image deconvolution and makes use of the sparsity property of images, it is called a sparsity deconvolution. The figure 3.12 summarises all the procedure that is discussed here.

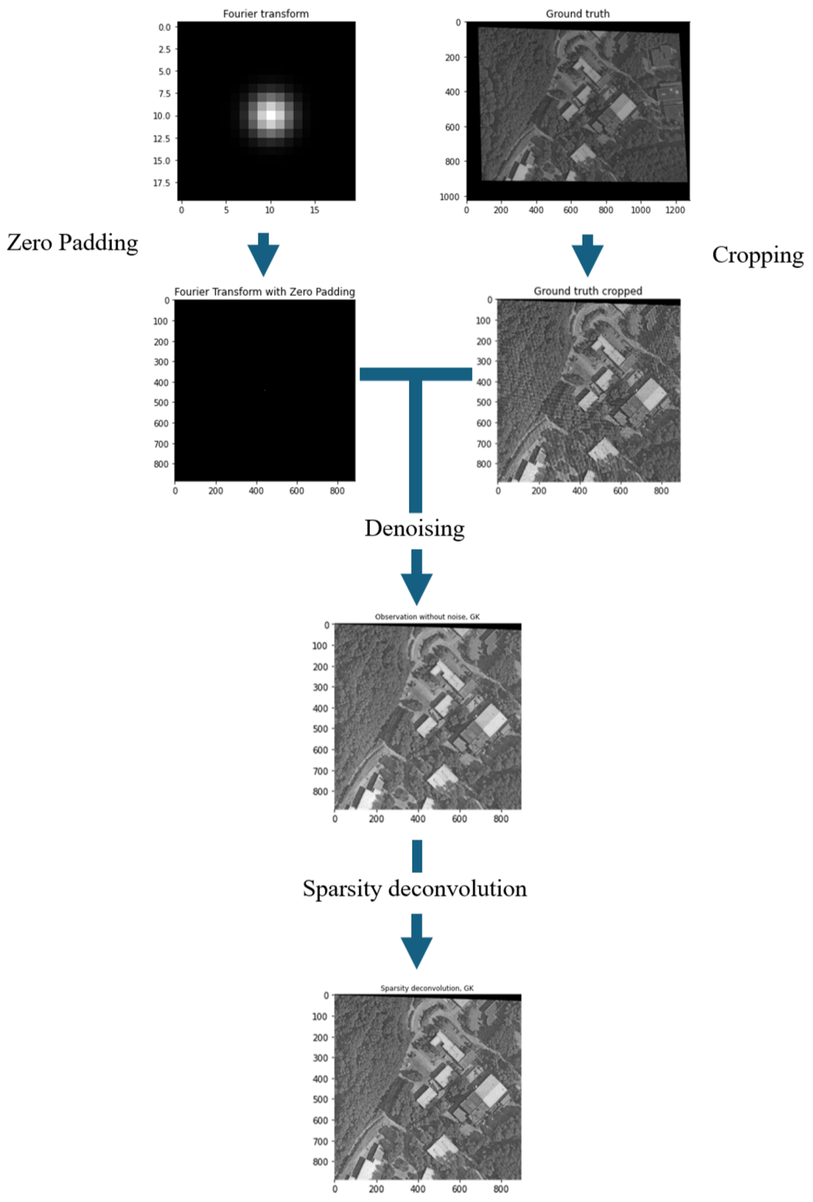


Figure 3.12: Overview of the image sparsity deconvolution process.

3.4 Results and discussion

3.4.1 Zero padding

In total, a series of four rectangular images were submitted to the methodology presented in figure 3.6 in order to establish which of the two different ZP techniques is the most advantageous and determine if some trends emerge from the main results. The table 3.1 gathers the PSNR values obtained for the different reconstructions, and all the corresponding images are displayed in the appendix section A.1.

Images	**PSNR [dB]**			
	ZP		**Improved ZP**	
	Before cropping	After cropping	Before cropping	After cropping
Ariane	40.56	28.32	34.21	28.25
CSL	39.28	31.40	34.98	31.42
PLATO	45.44	43.39	45.78	44.18
Spacesuit	37.41	30.50	33.22	30.59

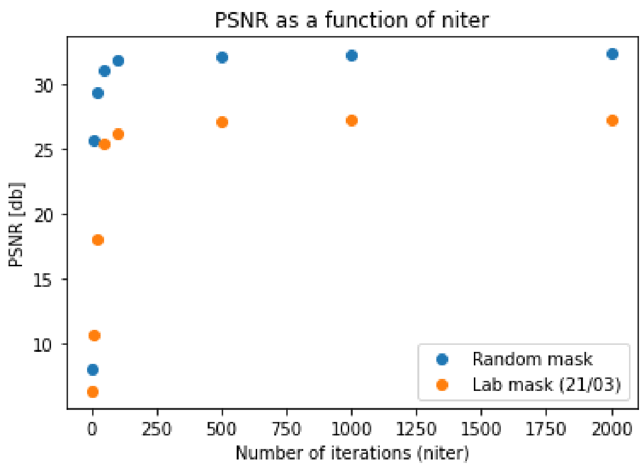
Table 3.1: PSNR values obtained for reconstructions with ZP.

Two main information can be deduced from this table. Firstly, the values acquired before cropping the corrected images are usually higher for the initial technique than for the improved one. This is simply explained by the fact that the images modified by the first version of ZP contain in general more extra zero value pixels added by completion. These pixels conserve their value after the reconstructions, and consequently the MSE computed between the images before and after the reconstructions decreases globally, leading to an increase of the PSNR. Secondly, PSNR values are of the same order of magnitude for the two methods once the reconstructed images are cropped. The slight variations come from the randomly generated mask that is applied on the images with ZP. The mask was in reality regenerated for each reconstruction and its distribution differs thus every time, even if the FR remains always the same. As only one part of the image with ZP corresponds to the initial image (the other part being the added data information coming from the ZP itself), this part is, as a result, more or less masked depending on the mask pattern in this precise area.

As the primary and the improved ZP techniques provide similar results when the reconstructed cropped images are compared to the initial ones, the improved ZP was kept in the final version of the IHT code as it can be seen in section B.1. The justification of this choice is the gain of computation time offered by this second approach of the ZP technique, as it was already mentioned before in section 3.3.1. Even if most of the images tested in this work are originally square images and do not require any completion by ZP a priori, the ZP technique is not removed from the main code as it can serve as a verification step to check the images dimensions, and hence avoid size errors in further steps.

3.4.2 Reconstructions based on an observed scene

The figure 3.13 and table 3.2 below present the results obtained on the basis of the methodology described in section 3.3.2. All the correlated restored images are available in the appendix section A.2.



NOI	PSNR [dB]	
	Random mask	Lab mask (21/03)
2	8.12	6.39
10	25.71	10.69
20	29.28	18.04
50	31.11	25.41
100	31.85	26.19
500	32.12	27.10
1000	32.28	27.26
2000	32.33	27.31

Figure 3.13 & Table 3.2: NOI and PSNR values for the 21/03 reconstructions. Comparison between random and lab masks results with a decaying value of λ.

Some interesting information can be deduced from these results. First, the quality of the reconstructions has the same behaviour for the two types of mask, as the two plots displayed in the figure 3.13 have the same trends with an increasing value of the NOI. More precisely, a sharp increase of the PSNR values occurs for small NOI values and is followed by a nearly constant evolution starting at about 500 iterations. This kind of plateau zone somehow symbolises the limitations of the algorithm that cannot reach larger PSNR values with the selected parameters. This particular evolution for large NOI values will be observed later in similar types of graphs presented in figures 4.1, 4.2, 4.3, 4.4 and 4.5.

Moreover, it clearly appears that the random mask leads to the production of better reconstructions results as the associated PSNR values are higher than the ones obtained when using the lab mask. This can be explained by comparing the distribution of black and white areas in the two masks. Indeed, a quick look at the figure 3.7 allows to directly see the difference between the two masks. On average, the random mask contains thinner patterns than the lab mask, due to a distribution of 0-value and 1-value pixels that is a little bit more homogeneous (even if the FR is the same in both situations as a reminder). As a consequence, the inpainting technique seems thus to be quite more efficient for images having thin damaged zones.

3.4.3 Optimisations

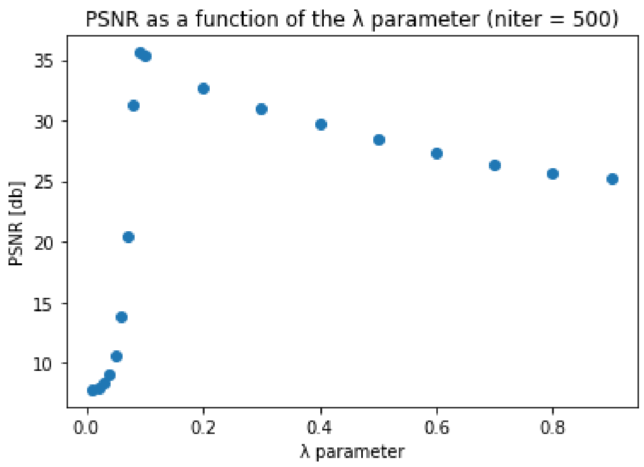
This section exposes the results that are related to the optimisation of the arbitrarily fixed threshold value playing a significant role in the IHT reconstruction algorithm. The succession of steps listed in section 3.3.3 were first performed by using 10 different values for the fixed λ parameter, going from 0 to 1 in steps of 0.1. However, the corresponding plots of the PSNR as a function of the λ parameter showed decreasing PSNR values for increasing values of λ, without containing a maximum value as it was expected. To determine if this peak value was located at a lower range of λ parameter values, and in order to refine the determination of the threshold leading to the best reconstructions results, additional images restorations were performed between $\lambda = 0.01$ and $\lambda = 0.1$, this time in steps of 0.01.

All the results of this optimisation process are gathered in figures 3.14, 3.15 and in tables 3.4, 3.4. They were all obtained by choosing a NOI of 500. The justification of this choice relies on the results mentioned in the previous section, where a NOI value of 500 iterations seemed to constitute a good compromise between satisfactory reconstructions and a relatively low computation time. Obviously, the higher the NOI, the longer the computation time.

Two possible interpretations can be developed regarding the optimisation results. The first one is that the same value of $\lambda = 0.09$ allows to create the best reconstructions (maximal PSNR values), whether in the case of a well structured scene (21/03 GT) or an uneven one (27/03 GT). Even if it is impossible to affirm if this value would be applicable to any type of observed scenes, one can already confirm that a suitable fixed threshold value is neither too small (i.e. not too close to 0) nor to high (i.e. not too close to 1). To understand why an intermediate value is obtained in both cases, a link can be made with what was already mentioned in the introductory paragraph of section 2.4. As the threshold of the Hard Thresholding operator is responsible for the classification of the image's wavelet coefficients, its fixed value directly impacts the quality of the reconstructions.

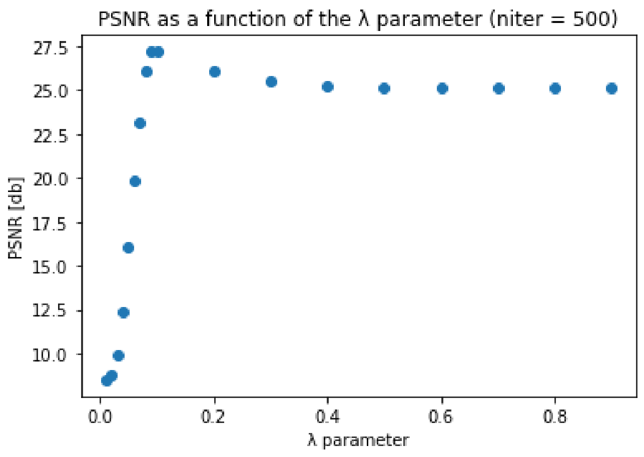
More exactly, a too low threshold value means that most of the wavelet coefficients are kept during the reconstruction process. As the algorithm cannot select properly the right coefficients, due to the fact that a too high diversity of possibilities exist, this situation consequently leads to partial, or at least poor, reconstructions. This explains why PSNR values drop for small λ parameter values, as it is illustrated in figures 3.14 and 3.15. On the contrary, a too high threshold value means that not enough wavelet coefficients are conserved in the IHT algorithm (most of them are set to zero). An image restoration is therefore possible but is however suboptimal. This justifies why large λ parameter values provide PSNR values that are relatively high, but that are not the maximal one though, as it can be seen once again in figures 3.14 and 3.15.

A second interpretation concerns then the comparison of PSNR values acquired for the well structured and uneven scenes. The maximum value derived from the first situation reaches 35.6 dB, compared to 27.2 dB for the second situation, which is a quite large variation taking into account that the PSNR is expressed using a logarithmic scale as a reminder. As it was already justified in the last section, this difference may be explained by the fact that the two scenes were masked by two distinct types of mask for which the patterns vary from each other. Also, results might slightly differ because the two original undamaged images, and so their respective content, are intrinsically different at the start.



λ **parameter**	**PSNR [dB]** with random mask
0.01	7.8
0.02	7.9
0.03	8.3
0.04	9.0
0.05	10.6
0.06	13.9
0.07	20.5
0.08	31.3
0.09	35.6
0.1	35.4
0.2	32.7
0.3	31.0
0.4	29.7
0.5	28.4
0.6	27.4
0.7	26.4
0.8	25.7
0.9	25.2

Figure 3.14 & Table 3.3: Optimisation of the λ parameter value for a well structured scene reconstruction with niter = 500 fixed.



λ **parameter**	**PSNR [dB]** with eroded mask
0.01	8.5
0.02	8.8
0.03	9.9
0.04	12.4
0.05	16.1
0.06	19.8
0.07	23.2
0.08	26.1
0.09	27.2
0.1	27.1
0.2	26.1
0.3	25.5
0.4	25.2
0.5	25.1
0.6	25.1
0.7	25.1
0.8	25.1
0.9	25.1

Figure 3.15 & Table 3.4: Optimisation of the λ parameter value for an uneven scene reconstruction with niter = 500 fixed.

3.4.4 Grayscale adjustment

On the basis of what was described in the section 3.3.4, the figure 3.16 below compares an original image, one of its possible reconstructions and the correction of the latter.

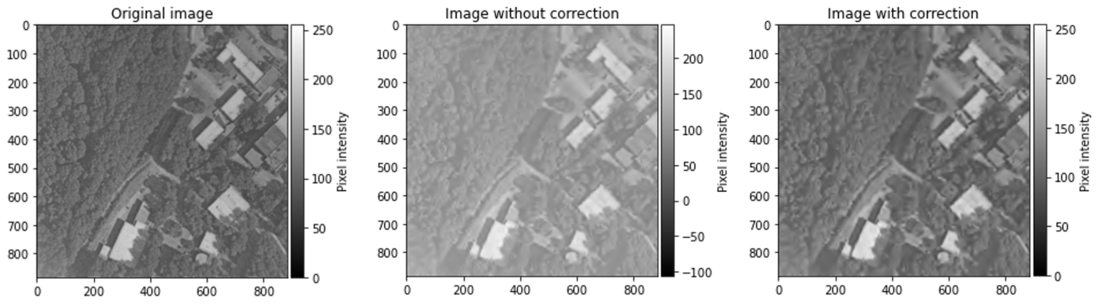


Figure 3.16: Grayscale adjustment results.

It can be easily observed that the uncorrected image is much clearer than the GT, and its colorbar represents an unrealistic range of pixel intensities including negative values. The corrected image, however, is accompanied by a classical colorbar gathering the 256 expected values of gray. In addition, its global appearance is much more similar to the one of the observed scene.

3.4.5 Image deconvolution using PSF

The main results that are linked to the image deconvolution process presented in section 3.3.5 are illustrated in this last part of the digital image processing chapter. First of all, an unprocessed version of the cropped GT is provided at the figure 3.17 so that it can be easily compared to the denoising and sparsity deconvolution results. The upper black band located at the top of this cropped scene was kept on purpose and served, during preliminary tests of the image deconvolution technique, as a reference for the image positioning, as it was necessary to slightly modify the image cropping when carrying out the image denoising step by a measured PSF.

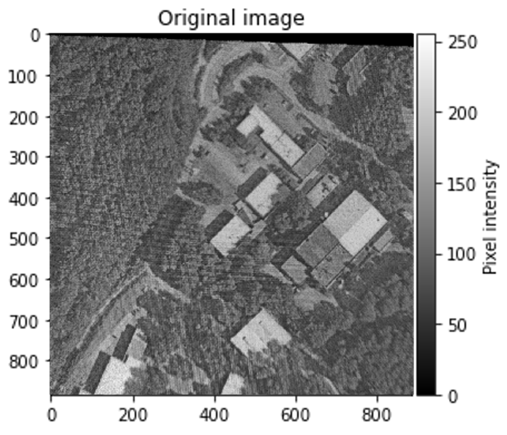


Figure 3.17: Original cropped scene containing noise.

The figure 3.18 shows which kind of typical denoised images can be obtained with the methodology introduced earlier. As one can see, the obtained images appear to be much smoother than the GT for both the GK and the measured PSF cases, which is the modification that was expected when performing a denoising of the GT. In addition, the two corrected scenes are a little bit clearer than the original image, which seems to indicate that the global contribution of the noise that is initially present in the image content tends to darken it. Besides that, the figure 3.19 gathers the sparsity deconvolution results showing corrected images too.

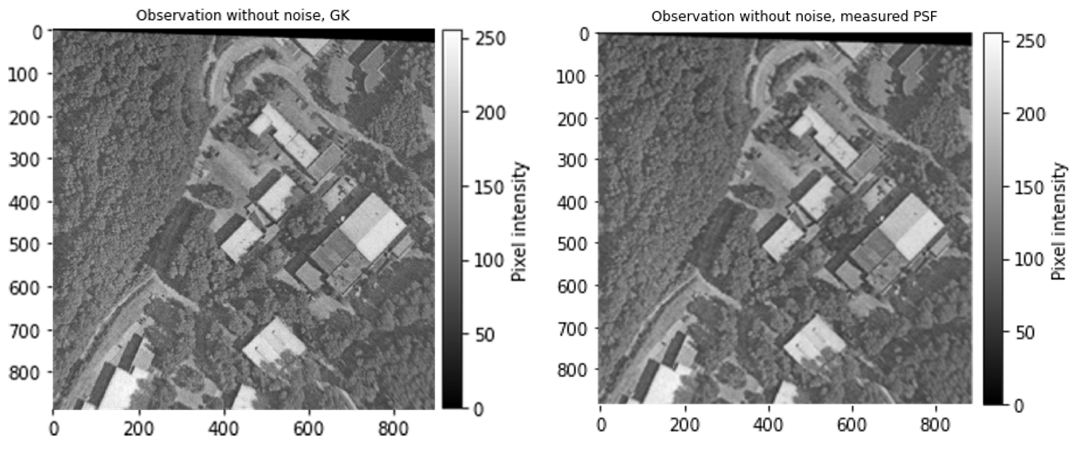

Observation without noise, GK
Observation without noise, measured PSF
Pixel intensity
Pixel intensity

Figure 3.18: Denoised cropped scene obtained by applying the FT of a GK (on the left) and of a measured PSF (on the right) on the noisy cropped GT.

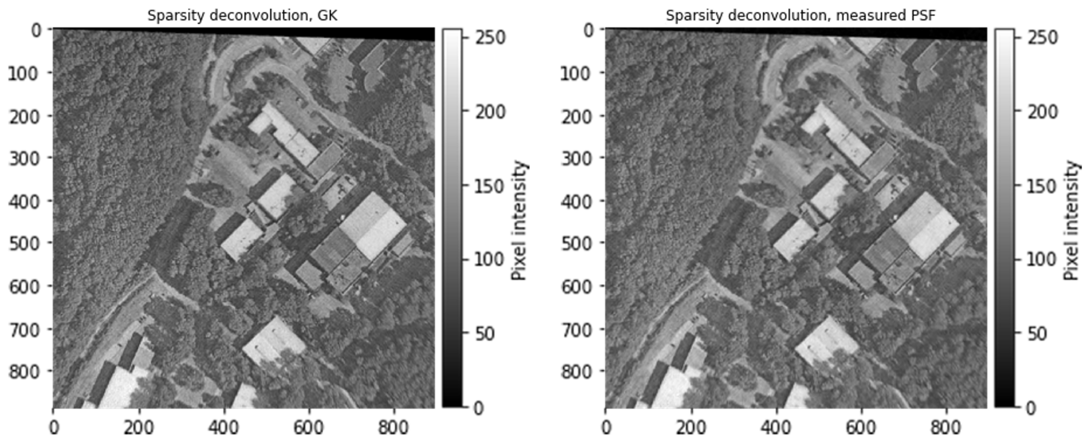

Sparsity deconvolution, GK
Sparsity deconvolution, measured PSF
Pixel intensity
Pixel intensity

Figure 3.19: Results obtained for the sparsity deconvolution involving a GK (on the left) and a measured PSF (on the right).

However, it is impossible, at this stage, to perform PSNR computations in order to compare the quality of the image rectifications, and the explanations provided in this section are thus purely qualitative. Indeed, the denoised / deconvolved images do not correspond anymore to the GT, and determining PSNR values between corrected images and the observed scene is therefore not relevant. Moreover, the original image that does not contain any imperfection is not available as it is observed by the optical setup from which the perturbations come from. As a consequence, PSNR values comparing the unperturbed image and the deconvolution results cannot be obtained too. Nevertheless, it is still possible to affirm that the techniques involved in this section improve the quality of the GT, as they remove the kind of grainy aspect that can be easily observed at the figure 3.17 and that is not present in the initial image. Visual variations between the different results are however barely perceptible.

It is also essential to remember that in the case of the measured PSF, certain approximations could affect the obtained results. Firstly, the trial and error method tested for the image denoising correction is certainly not optimal and it would be interesting to find a more automated version. Secondly, the acquisition of the PSF, as it was already mentioned before, relies on one micromirror of the DMD that is set in its ON configuration. Nevertheless, the definition of the PSF is, as a reminder, based on the observation of a point source having an infinitely small spatial distribution, which is of course not the case of the DMD components. Improving as much as possible these two considerations could probably lead to results that are even better than those obtained here.

Chapter 4

Applications

In consequence of everything that has been described before, it is possible to involve the procedures followed through all this document in more practical situations, in order to test their potential limitations and evaluate their efficiency regarding new types of simulations. On that purpose, this chapter is dedicated to the description of two applications of the inpainting reconstruction process, with emphasis on an adaptation of the methods previously presented in this work. Firstly, observations of some damaged scenes will provide the possibility of testing the quality of the images restorations, with a comparison of the effects caused by non-eroded and eroded masks on the one hand, and an extension of these masks' FR on the other hand. Secondly, a more pragmatic problem, that is in direct connection with corrupted data derived from the Landsat 7 space mission, is introduced and accompanied by a description of the general related context. All the results obtained in the framework of this master thesis for these two series of tests are presented in the following sections.

4.1 Observations of damaged scenes

Contrary to what was already covered in the digital image processing chapter, where the images generated by the CS imager designed in laboratory were completely observed and were after that numerically masked thanks to a predefined binary pattern, this section focuses on another type of reconstructions. Here, the objective is to carry out identical images completions, at the difference that the observed scenes are already damaged when observing them through the optical setup (presented in section 3.2.1 as a reminder), in which the micromirrors of the DMD are adequately set to produce the missing parts of the GT of interest. Purposely, different masks were also obtained using the same imaging device so that their patterns exactly correspond to the lost areas of the images that need to be corrected. These masks show variations when comparing them, and these changes are intentionally implemented to see how these modifications actually affect, in a positive or negative manner, the reconstructions results. Hereafter, an evaluation of the quality of the images restorations, based on the masks properties, is provided.

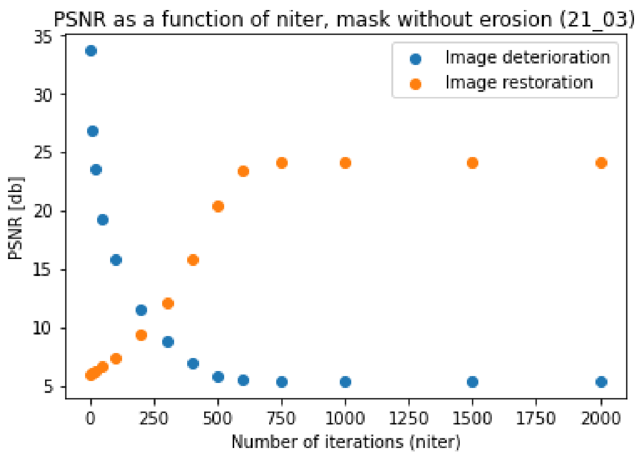
4.1.1 Comparison between non-eroded and eroded masks

As it was discussed in the section 3.2.2, the production of eroded masks by the morphological erosion method is realised as a mean to rectify the reconstructions imperfections caused when employing simple binary masks. As a result, it is therefore expected that the eroded version of masks leads to better results than the non-eroded one.

Here is the methodology followed to determine if the last affirmation is effectively correct or not. First, a series of reconstructions are performed for a damaged scene on which a binary mask without erosion is applied. This series of acquisitions is realised by taking different NOI values. Afterwards, another set of images reconstructions is carried out for the same GT, involving this time an eroded mask and conserving identical values of the NOI than in the previous case. PSNR values are then extracted to characterise and compare the quality of the results.

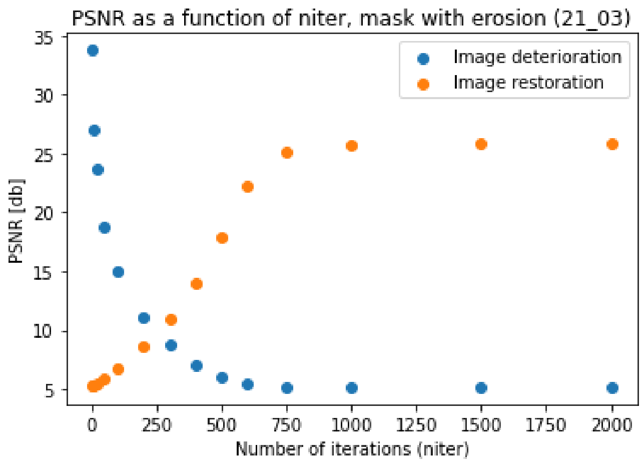
Nevertheless, it is important to mention that the computation of PSNR values is, throughout this entire section 4.1, performed in two distinct ways for each of the analysed situations. The first one compares the reconstructions results with the damaged GT, which causes a decreasing evolution of the PSNR as the NOI increases, due to the fact that the image is more and more restored, and that pixels of damaged zones are progressively refilled. Consequently, this type of PSNR values, denoted in this paragraph as PSNR1, is hereafter the one that is related to the so-called image deterioration, the latter being high when there are still most of the image's missing parts after the reconstruction (small NOI value), and low otherwise. All the reconstructed images of this current section 4.1 gathered in the appendices are accompanied by this specific PSNR1 value. In contrast, the second kind of PSNR, called PSNR2, is the traditional one obtained when comparing a complete scene with the corresponding restored images. This is made possible by acquiring a second GT that represents the same image portion than the one delimited by the damaged GT but with all of the micromirrors of the imager's DMD tilted in the ON configuration. This PSNR2 value is the same as the one presented in all the chapter 3 and will for this reason be linked to the image restoration, for which the PSNR values increase as the NOI increase too, as the image is less and less damaged. In summary, it is now possible to produce simultaneously two series of PSNR values (PSNR1 and PSNR2) for a same set of reconstructions, that represent either the image deterioration or the image restoration, these last two systematically presenting an evolution that is opposed to the one of the other.

All the results associated to the comparison of the effects of non-eroded and eroded masks when they are introduced in the IHT process are illustrated in figures 4.1 and 4.2, and listed in tables 4.1 and 4.2. They were obtained by selecting a fixed value of the IHT threshold (i.e. of the λ parameter) equal to 0.09, as this value was determined as being the optimised one for the scene of interest, which is in the present case the one already exposed before at the figure 3.7. All the images allowing to visualise the results are available at the appendix sections A.3.1 and A.3.2. As expected, the image deterioration is accompanied by a decreasing evolution while the image restoration shows an increasing evolution as the NOI increases, for both the reconstructions involving a mask with and without erosion. More precisely, PSNR values of the deterioration first decrease sharply, before stabilising at high NOIs. Besides that, the increase of the PSNR values related to the image restoration is more moderate at the beginning (low NOI values) until reaching a similar plateau zone corresponding to the limitations of the reconstructions.



NOI	**PSNR [dB]** mask without erosion	
	Image deterioration	Image restoration
2	33.63	5.99
10	26.75	6.10
20	23.57	6.24
50	19.25	6.65
100	15.74	7.42
200	11.53	9.35
300	8.72	12.06
400	6.90	15.78
500	5.85	20.33
600	5.46	23.30
750	5.38	24.09
1000	5.37	24.15
1500	5.37	24.15
2000	5.37	24.15

Figure 4.1 & Table 4.1: NOI and PSNR values for the 21/03 reconstructions. Mask without erosion, $\lambda = 0.09$.



NOI	**PSNR [dB]** mask with erosion	
	Image deterioration	Image restoration
2	33.76	5.25
10	27.07	5.37
20	23.61	5.51
50	18.83	5.94
100	15.04	6.73
200	11.15	8.61
300	8.76	10.96
400	7.10	13.98
500	5.99	17.95
600	5.42	22.28
750	5.21	25.19
1000	5.18	25.71
1500	5.18	25.80
2000	5.18	25.80

Figure 4.2 & Table 4.2: NOI and PSNR values for the 21/03 reconstructions. Mask with erosion, $\lambda = 0.09$.

In order to visualise in a more efficient manner the effects of the two types of tested masks, the figure 4.3 displays the image restoration results of both figures 4.1 and 4.2. For NOI values smaller than 750, the PSNR values for reconstructions involving an eroded mask are lower than the ones of the non-eroded mask. However, which is interesting to notice here is that the eroded mask's PSNR values become greater than the non-eroded ones from a NOI equal to 750. Even if the FR of the eroded mask is enlarged (which means that more GT content is initially lost) compared to the one of the mask without erosion (as mentioned in section 3.2.2), and produces therefore poorer restorations at small NOI values, the mask that contains a pattern submitted to the erosion correction provides better reconstructions results at high NOI values than the mask without erosion, which is in agreement with the aforementioned expectations.

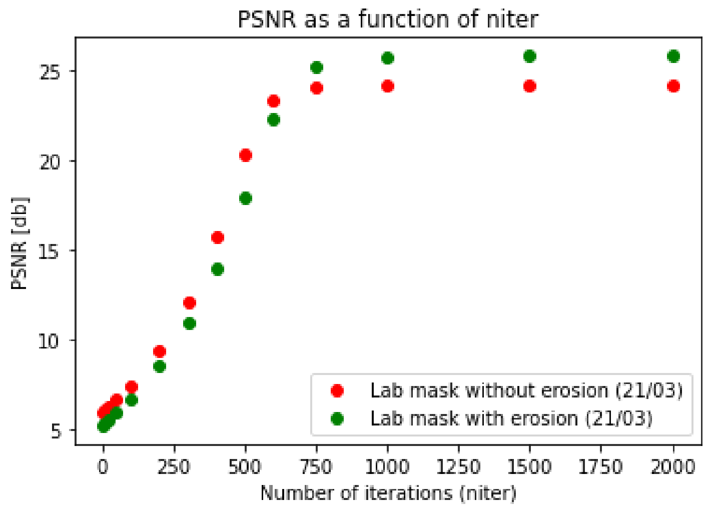


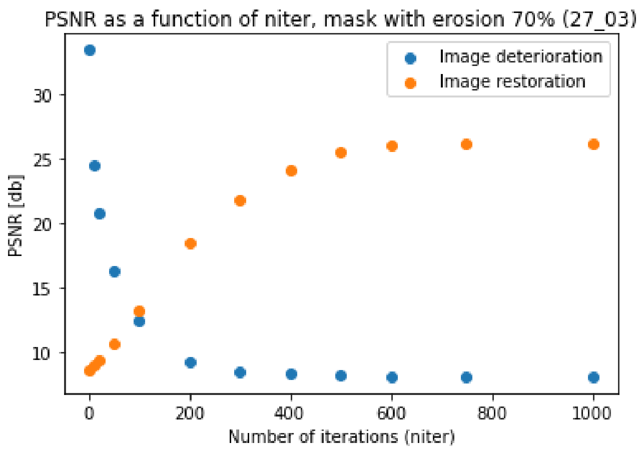
Figure 4.3: Comparison of the mask erosion effect on the PSNR values obtained for the 21/03 reconstructions, with $\lambda = 0.09$.

4.1.2 Extension of the mask filling ratio

This new application is focused on a methodology that is exactly the same than the one described in the previous section 4.1.1. However, the main objective differs and consists, in the present case, of comparing the effects produced by two masks having a different FR. The motivation of this new series of tests is to evaluate how this FR can impact the quality of the images reconstructions and to see which kind of limitations the IHT algorithm can encounter for a considerable loss of information regarding the content of the damaged scene.

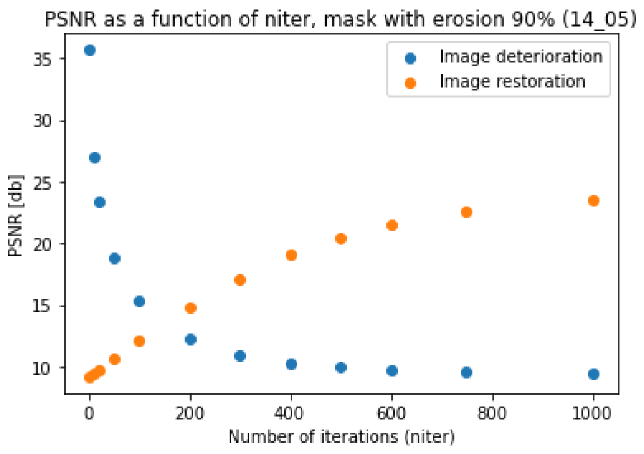
To meet the new reconstructions requirements, the observations of two eroded masks and two scenes were carried out by using the optical setup described above, on two different dates. The acquisitions performed on March 27 (27/03) correspond to a mask having a FR equal to 70 %, while data obtained on May 14 (14/05) are linked to a mask having a FR of 90%. The two observed scenes are derived from the same initial image and are thus comparable.

Figures 4.4 and 4.5, as well as tables 4.3 and 4.4, collect the results obtained for the two series of PSNR computation, for the masks having a different FR. All the correlated restored images are presented in appendix sections A.4.1 and A.4.2. Now, the NOI vary between 2 and 1000, as this maximal value showed satisfactory results in previous simulations. An additional reconstruction was performed in the situation where the GT is masked at 90%, with a NOI of 2000, with the aim to visually improve the image restoration. The associated image is displayed in the appendix section A.4.2 but the related PSNR value is not displayed in the following graphs and tables. As for the λ parameter, it is kept equal to 0.09 as the undamaged version of the observed scenes is initially the same as the one of figure 3.8.



NOI	PSNR [dB] mask with erosion 70 %	
	Image deterioration	Image restoration
2	33.40	8.56
10	24.44	8.89
20	20.77	9.31
50	16.20	10.65
100	12.43	13.21
200	9.22	18.48
300	8.47	21.81
400	8.24	24.02
500	8.13	25.44
600	8.07	26.01
750	8.05	26.14
1000	8.05	26.14

Figure 4.4 & Table 4.3: NOI and PSNR values for the 27/03 reconstructions. Mask with erosion, FR = 70 %, λ = 0.09.



NOI	PSNR [dB] mask with erosion 90 %	
	Image deterioration	Image restoration
2	35.65	9.19
10	26.96	9.45
20	23.34	9.76
50	18.81	10.67
100	15.31	12.12
200	12.29	14.79
300	10.99	17.13
400	10.31	19.02
500	9.95	20.41
600	9.76	21.43
750	9.59	22.53
1000	9.46	23.46

Figure 4.5 & Table 4.4: NOI and PSNR values for the 14/05 reconstructions. Mask with erosion, FR = 90 %, λ = 0.09.

At first glance, the four plots show similar evolution than the ones illustrated in figures 4.1 and 4.2, for both the image restoration and the image deterioration results, even if slight variations may appear due to the change of GT between the two sections. As one can expect, PSNR values are globally higher for the image restoration when damaging the GT at 70%, while PSNR computations provide higher results for the image deterioration when masking the scene at 90%. These two trends can be explained by the fact that more information is obviously corrupted with a mask having a FR of 90% than with another mask its FR equal to 70%. The algorithm is thus logically less efficient for images having a high rate of missing data than in the opposite case. As a consequence of all this, reconstructions images available in the appendix section A.4.2 are less sharp and contain less details than the ones exposed in A.4.1. The figure 4.6 shows the difference between the 27/03 and the 14/05 final reconstructions.

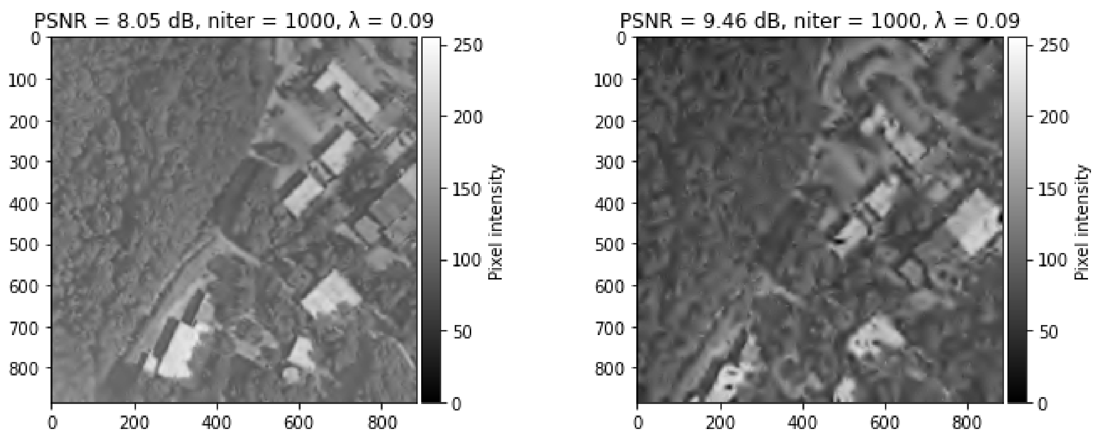

PSNR = 8.05 dB, niter = 1000, λ = 0.09
PSNR = 9.46 dB, niter = 1000, λ = 0.09
Pixel intensity
0
100
200
300
400
500
600
700
800
250
150
50

Figure 4.6: Final results of the 27/03 (on the left) and the 14/05 (on the right) reconstructions. PSNR values are the ones related to the images deterioration.

Despite that, most of the initial image areas are recognisable after the reconstructions involving an eroded mask for which the FR is equal to 90% (right part of the figure 4.6), which confirms that the inpainting algorithm is capable to provide acceptable results even if most of the initial data is missing at the start.

4.2 Reconstructions based on space mission measurements

The second half of this chapter is dedicated to another type of application, based on the past Landsat 7 space mission from which the data are managed by the United States Geological Survey (USGS). First, an overview of the mission and of the context in which it took place will be presented. After this presentation, a description of the production of scientific data by the Landsat 7 satellite is provided, and complemented by additional information concerning the available data types, with emphasis on which are the ones effectively processed in the framework of this master thesis. The corresponding procedure followed for the images reconstructions is then summarised and the quality of the results obtained through this method is finally reviewed.

4.2.1 Landsat 7 mission

Launched in California on April 15, 1999, the Landsat 7 mission is part of the NASA's Landsat missions, focused on satellite-based Earth observation and monitoring. The nominal phase of the mission, that ended on April 6, 2022, was configured such that the Landsat 7 spacecraft orbited the Earth in a classical sun-synchronous, near-polar orbit having an inclination of 98.2 degrees and an altitude of 705 kilometres. These parameters were chosen for two main reasons, namely to receive enough energy from the Sun, the spacecraft being partially powered by a Sun-tracking solar array, but also in order to cover all of the Earth's regions of interest (with a worldwide coverage accomplished in 16 days). Nevertheless, series of spacecraft manoeuvres, correcting its inclination, were needed to maintain the mean local time of the acquisitions (as it slightly changed over the years), with a view to conserve an adequate reference time for the data production. The last of these operations was carried out on February 7, 2017 and the orbit began to progressively degrade.

However, new manoeuvres were performed on April 6, 2022, in order to lower the initial orbit's altitude by 8 kilometres, thanks to a succession of spacecraft burns, that are illustrated at the figure 4.7. During these specific manoeuvres, the instruments embedded inside the spacecraft were switched to a stand-by mode. The new satellite's altitude allowed to refuel the Landsat 7 spacecraft in the framework of another servicing NASA mission. In this manner, the Landsat 7 acquisitions continued and the mission entered in its extended phase [19].

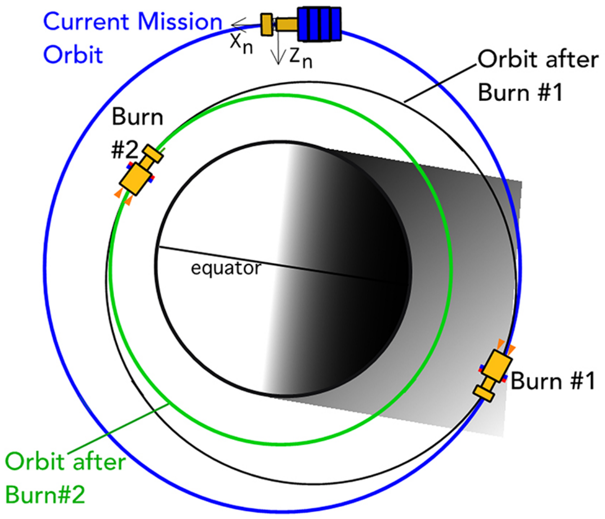


Figure 4.7: Orbital manoeuvres of the Landsat 7 spacecraft. From [19].

During this extended mission phase, that started on May 5, 2022, the Landsat 7 scientific devices produced no less than 450 multi-spectral images per day on average, for a total of about 175,000 observed scenes. The figure 4.8 below provides a global view of the different locations of these observed scenes, that are mainly gathered in continental areas. The whole mission finally ended on January 19, 2024, due to some problems encountered in terms of spacecraft's batteries deterioration [20].

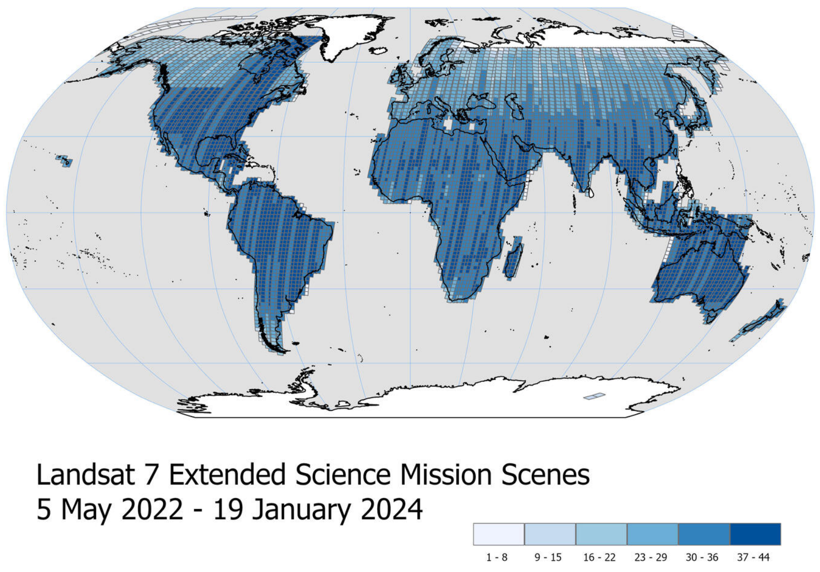

Landsat 7 Extended Science Mission Scenes
5 May 2022 - 19 January 2024
1 - 8
9 - 15
16 - 22
23 - 29
30 - 36
37 - 44

Figure 4.8: World map - areas covered by the Landsat 7 satellite during the extended mission, with the corresponding number of acquired scenes. From [20].

4.2.2 Data production, type and selection

All the data coming from the Landsat 7 mission were acquired by the Enhanced Thematic Mapper Plus (ETM+) instrument, a multi-spectral sensor corresponding to an enhanced version of previous scientific devices found aboard the spacecrafts of the preceding Landsat 4 and Landsat 5 missions. The ETM+ sensor allowed in fact to observe the regions of interest in 8 different spectral bands, varying in terms of their wavelength range and of their resolution, the latter being sometimes referred as the ground sampling interval. The three first bands correspond to the traditional RGB decomposition on the basis of which true colour images are formed. Three other bands are then included in the infrared domain and comprise a Near Infrared (NIR) band, a Mid-Infrared (MIR) one and a Short-Wave Infrared (SWIR) one. Two extra bands, composed of a thermal one and a panchromatic one, complete the observation range [19]. Panchromatic images, for which the spectral range is located in the visible / NIR domains, are less precise in terms of spectral resolution but they offer a better spatial resolution. All the bands and their features are listed in the table 4.5.

Data acquisitions were performed by using the whiskbroom satellite imaging technique, schematised at the figure 4.9. This acquisition method, largely developed in the field of remote sensing applications, consists of a rotating mirror scanning crosswise the satellite's path and reflecting the light towards the satellite's optics, until reaching the related detector, and creating each of the image pixels at a time. Thanks to this technique, the Landsat 7 ETM+ instrument observe elongated strips named swaths, that have a width of 185 kilometres in the case of the Landsat 7 mission and that correspond to the successive ground projections of the imager's angular FOV. This angular FOV is segmented into smaller

Band number	**Name**	**Wavelength range [μm]**	**Ground sampling interval (pixel size) [m]**
1	Blue	0.45 - 0.52	30
2	Green	0.52 - 0.60	30
3	Red	0.63 - 0.69	30
4	NIR	0.77 - 0.90	30
5	SWIR	1.55 - 1.75	30
6	Thermal	10.40 - 12.50	60
7	MIR	2.08 - 2.35	30
8	Panchromatic	0.52 - 0.90	15

Table 4.5: Spectral bands of the ETM+ instrument. Based on information found in [19]

parts, each of them being called the instantaneous (or incremental) field of view (iFOV), for which the projection is this time the ground area covered by a single image pixel. The observed zone associated to this second type of projection is sometimes referred as a ground resolution cell and has a width that is designated as the so-called ground sampling distance [13] [21].

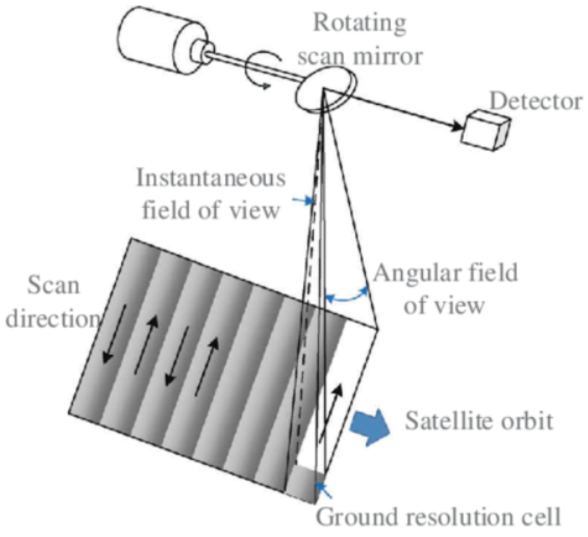


Figure 4.9: Whiskbroom satellite imaging technique. Adapted from [22]

One of the subsystems of the ETM+ instrument, the Scan Line Corrector (SLC), is a mechanism composed of two parallel mirrors designed to compensate for the satellite's forward motion. Thanks to this correction, the series of scans obtained by the whiskbroom method show a complete and rectilinear pattern, which is not the case otherwise. Scans performed without the SLC contain therefore a zigzag pattern, as it can be seen in the figure 4.10, comparing the two situations involving or not the use of the SLC mechanism. However, this device broke down from May 31, 2003, and all the attempts to repair it since this date were insufficient, making the failure permanent. All the produced images that followed were formed of non-rectilinear patterns separated by gaps and were thus generated in an incomplete way. This justifies why the Landsat 7 mission was selected to be studied in the frame of this work, in order to test if the inpainting technique covered in this master thesis can provide satisfactory results when correcting the damaged images.

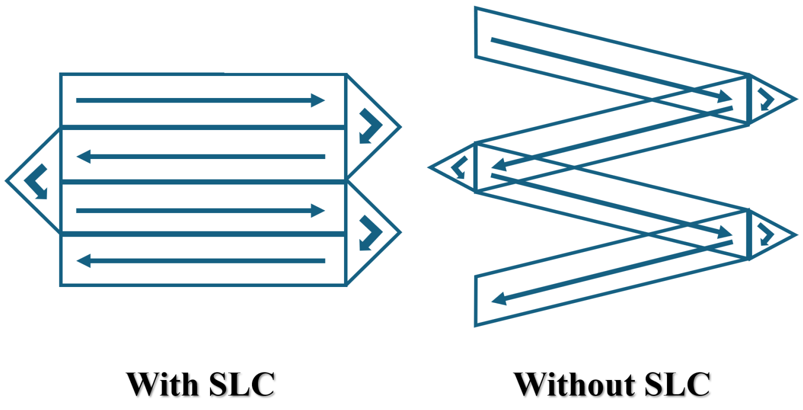


Figure 4.10: Impact of the SLC mechanism on the Landsat 7 images generation. In this configuration, the satellite's motion goes from top to bottom. Reconstructed on the basis of information found in [19] and [23].

Available online, all the data selected for the images reconstructions were downloaded and collected on the USGS database [24]. For the purpose of some pre-established requirements, two different GT were chosen, namely a non-damaged one, acquired before the SLC failure, and a second one showing the zigzag pattern produced after the SLC operating phase. In this manner, it is possible to follow the same type of methodology to simulate the reconstructions in the two situations and analyse this way the obtained results. The two satellite scenes of interest are displayed at the figure 4.11 below and are actually true colour images gathering the blue, green and red bands described in the table 4.5. However, due to the huge amount of pixels (8111 $\times$ 7431) in these two images, the portions of the scenes that were submitted to the reconstruction process were restricted to the areas delimited by the white squares, so that the computation time related to the restorations remains acceptable. The two portions were determined on the basis of their visible structures that will facilitate the visual interpretation when comparing the reconstructions results.

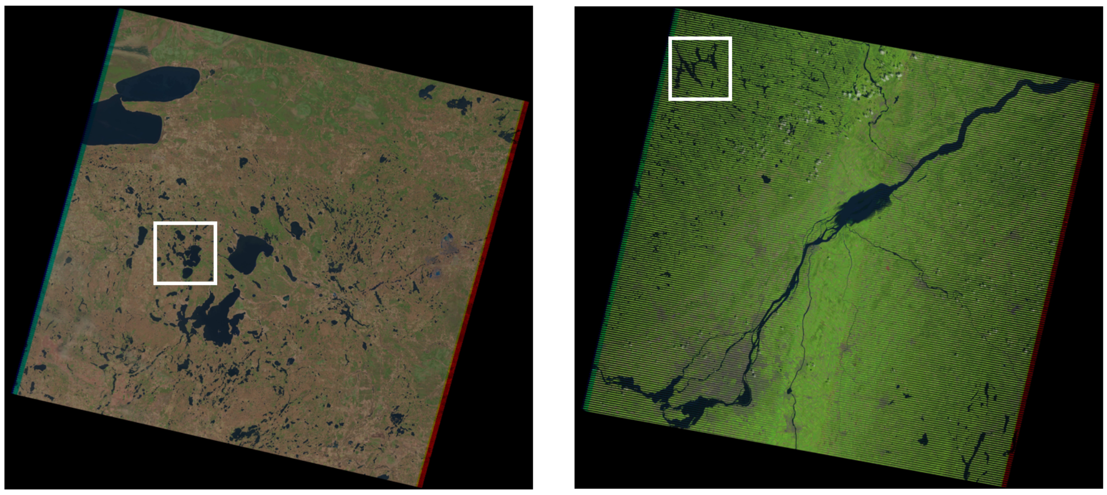


Figure 4.11: Initial undamaged (on the left) and damaged (on the right) satellite scenes, with the related regions of interest (white squares). Data downloaded on [24].

Only one element is still missing to perform the planned reconstructions. It is in reality the mask that will be applied on the two GT. The objective was therefore to find an identical pattern than the one of the damaged scene's missing information. On that purpose, a raw scan of the acquired thermal band, for the same observation than the one analysed here, was downloaded on the USGS website too. This particular image, having the same dimensions thand the GT, presents black and white patterns that exactly match the loss of data produced by the SLC dysfunction. As it is illustrated at the figure 4.12, the same image portion than the one of the true colour scene of figure 4.11 was conserved and is this time represented by a red square. Some preliminary operations were executed to check that this part of the mask pattern was effectively a binary one.

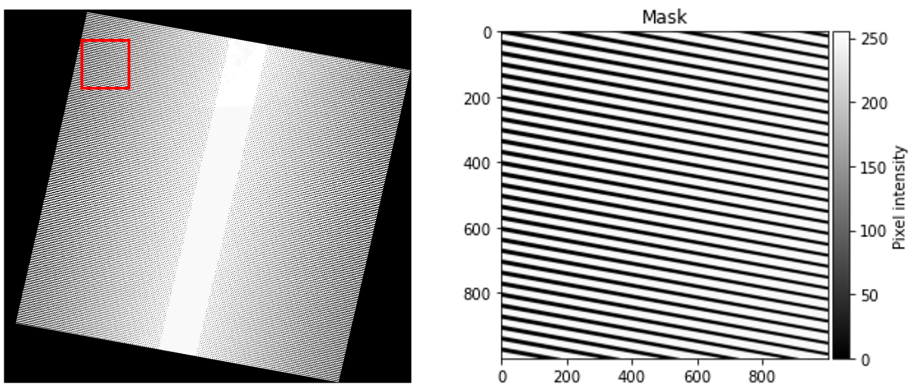


Figure 4.12: Raw scan linked to the thermal band acquisitions (on the left). The red square symbolises the mask's region of interest that is displayed besides (on the right). Data downloaded on [24].

4.2.3 Reconstruction procedure

The procedure followed to reconstruct the Landsat 7 images is quite similar to the different methodologies previously explored in this work. However, the true colour images processed in the current section are different from all the grayscale images processed so far. Consequently, the IHT code was partially adapted so that the new satellite scenes can be adequately manipulated. The changes implemented in the original numerical code are displayed in the appendix section B.3.

As it was already explained in the introductory section 2.1, true colour images have a colour depth of 24 bits, and can be split into three sub-images. The new images obtained with this decomposition are each associated to a precise colour (either red, green or blue) and have a colour depth being equal to the third of the one linked to the initial true colour representation (so, 8 bits in the present situation). Similarly, the three RGB images can be assimilated to three independent colour channels, the latter being schematised in the figure 4.13. As it can be seen, each of the colour channels is itself divided into three layers, namely a colour one and two black ones. Depending on the colour channel, the correlated colour layer is not distributed at the same position in the information content. Hence, the example proposed at the figure 4.13 shows that the red layer is positioned at the first stage, while green and blue layers are located at the second and third stages, respectively. Thanks to this configuration, the three RGB images can be recombined in order to reassemble the first true colour image. This recombination is part of the whole process described in this section, for which an overview is provided at the figure 4.14.

Figure 4.13: Illustration of the RGB colour channels and their layers. From [25]

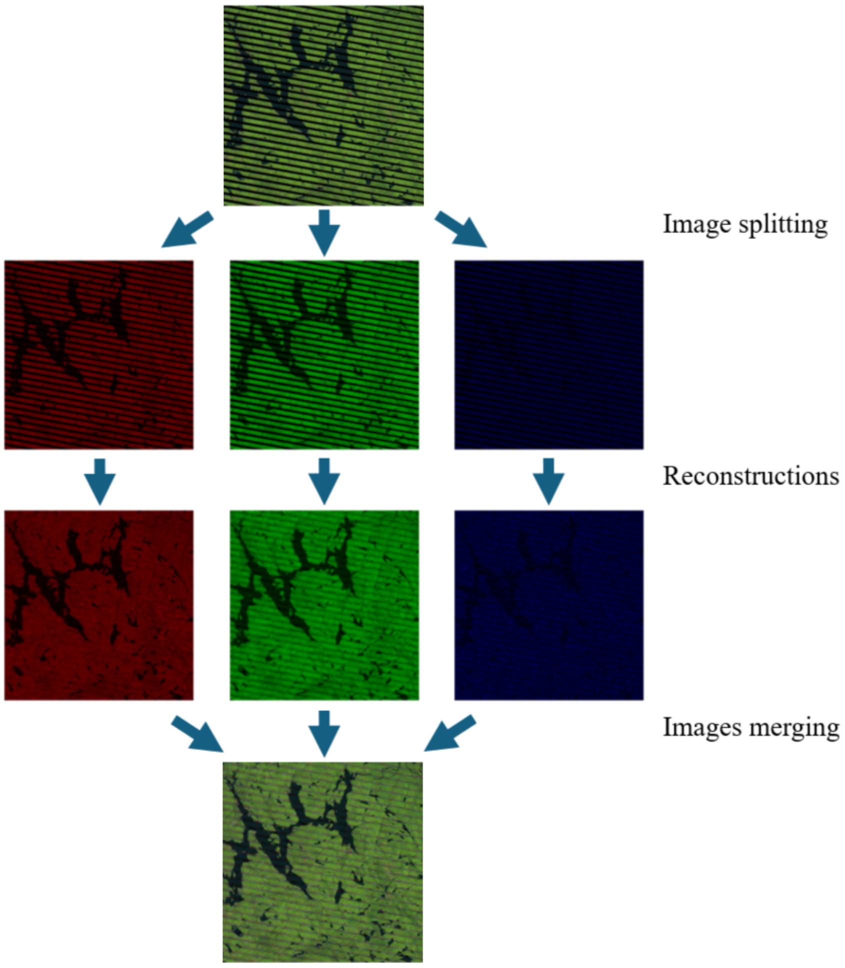


Figure 4.14: Overview of the true colour images reconstruction process.

The selected procedure aiming to reconstruct the incomplete Landsat 7 data is thus the following one. In order, the initial true colour GT is first split into its three RGB sub-images. From the latter are then extracted the corresponding colour layers that are implemented into the IHT reconstruction algorithm after having been masked by the pattern described in the previous section. The parameters chosen for each of the images restorations are a

NOI of 500 and a decreasing value of the λ parameter. Once the RGB layers are separately restored, they are reintroduced in their associated channel. The three channels are finally recombined to reconstitute the corrected true colour image. This series of steps is repeated for both the initially damaged and undamaged scenes, and a computation of the PSNR value between the original true colour image and its restoration completes the results in the case of the complete GT (which is of course impossible in the case of the incomplete GT as its undamaged version does not exist). More precisely, the different image manipulations are first carried out in the case of the undamaged GT to determine if the methodology is appropriate for such type of reconstructions. A similar process is then applied to the case of the damaged GT.

4.2.4 Results and discussion

In this last section, the results obtained for the reconstructions of the Landsat 7 satellite scenes are presented and interpreted. In order to visualise what kind of data is introduced in the reconstruction algorithm, the figure 4.15 is composed of the three masked layers of the scene that is initially undamaged. All the initial RGB layers, as well as their reconstructed form, are available in the appendix sections A.5.1 and A.5.2, in which they are displayed in their grayscale format. The figures 4.16 and 4.17 gather the original GTs and compare them to their true colour restored version.

Regarding the figure 4.16, it clearly appears that the technique described in the previous section is capable to provide satisfactory results for the Landsat 7 GT restorations. Some inevitable modifications between the two images are nevertheless observed. They simply come from the fact that some structures hidden by the mask pattern are smaller than the width of one single mask's black stripe, and the algorithm cannot reproduce them properly as they are totally removed by the masking operation. In some extent, this contribute to a certain loss of some of the image details that cannot be avoided. However, it seems that the image colour content is quite well preserved by the inpainting technique. The PSNR computed between the GT and its reconstruction has a value of 26.68 dB as it can be seen in the appendix section A.5.1.

As for the figure 4.17, the reconstruction technique looks a little bit less efficient than in the previous case in terms of the image colour content restoration, as zones harbouring the initial lines of missing data are still easily identifiable, showing the limitations of the process for scenes that are initially incomplete. Hypothetical improvements could be obtain thanks to a higher NOI value than the one selected here or by refining the λ parameter. The corresponding tests were however not covered in this work and it could be interesting to carry out further investigations about possible results enhancements. Besides all that, it is nevertheless interesting to notice that the algorithm allows to better distinguish the edges of the dark areas contained in the restored true colour image, these obscure regions undoubtedly corresponding to watering places.

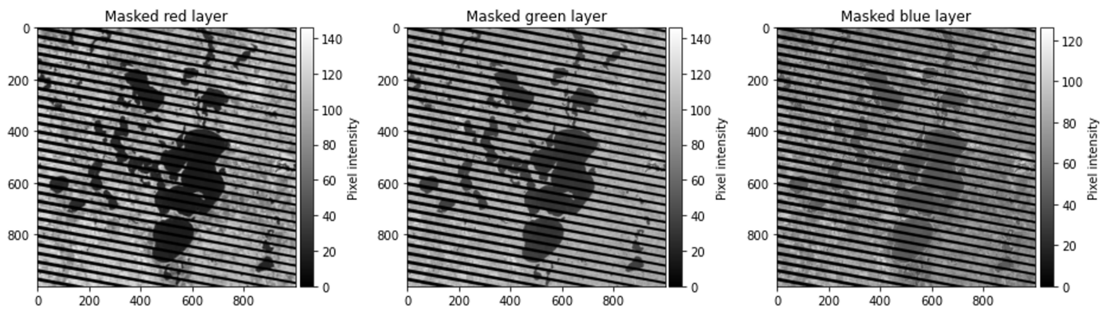


Figure 4.15: Masked RGB layers for an initially undamaged satellite scene.

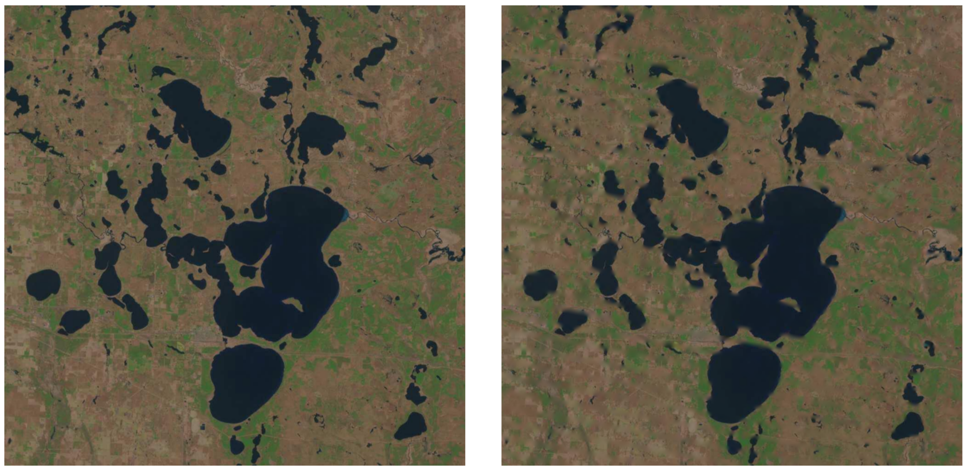


Figure 4.16: Comparison between the GT (on the left) and the associated reconstructed image (on the right) for an initially undamaged satellite scene.

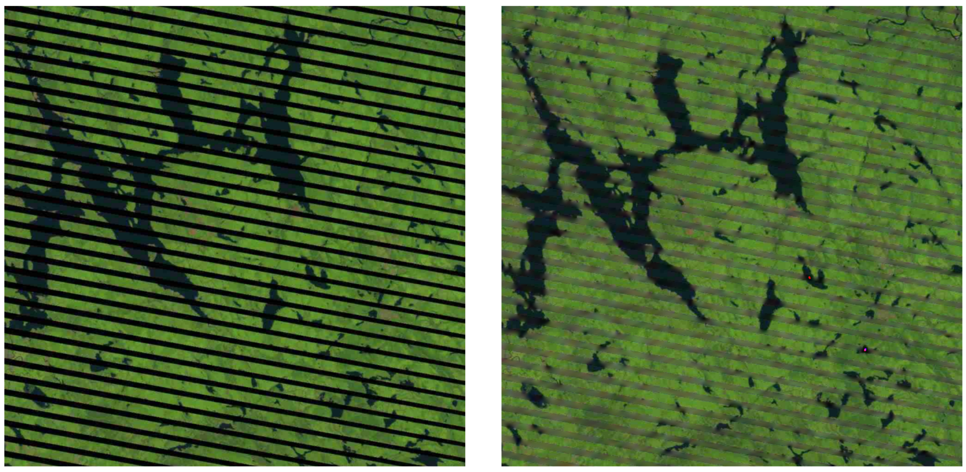


Figure 4.17: Comparison between the GT (on the left) and the associated reconstructed image (on the right) for an initially damaged satellite scene.

A little note concerning the image on the right of the figure 4.17 is that a close-up view of the reconstructed satellite scene shows the presence of two artifacts appearing in two distinct reconstructed areas. However, their origin cannot be determined on the basis of what was already discussed, and consequently it would be necessary to carry out additional systematic reviews in order to find where they effectively come from. Precise knowledge of their origin would then make it possible to remove them from the reconstructed image.

Chapter 5

Conclusion

5.1 Overall conclusion

This master thesis, focused on a digital approach of images reconstructions performed on the basis of classical inverse methods, had a primary objective. The latter was to develop relevant numerical algorithms, with a view of producing satisfactory results in the frame of Earth observation applications. This objective was accomplished, even if some inevitable limitations were encountered in terms of the selected algorithm's performances. Moreover, this document enters into the continuity of the thesis of Clément Thomas, one of the Centre Spatial de Liège's PhD students whose work is dedicated to the practical aspects of the compressive sensing method in the context of signals recoveries.

Directly following the introductory part of this work, the state of the art chapter described most of the theoretical concepts that necessarily need to be understood before carrying out any image manipulation. On that purpose, the main images properties were presented, and explanations of the typical inpainting inverse problem followed this presentation, with emphasis on the wavelet theory and the iterative hard thresholding algorithm on which the principal studied reconstruction method relies. The particular case of image deconvolution involving a point spread function was also covered.

Starting with a learning phase of the Python programming language, the digital image processing part of the thesis was devoted to the implementation of new techniques in the tested digital codes, including notably the zero padding method and the correction of the images' colorbars. This chapter also showed the importance of selecting the right parameters value within the tested algorithm, and particularly exposed the impacts of both the number of iterations and of the threshold λ values on the reconstructions results. The relevance of an appropriate selection of the mask hiding the original processed image was explained too and associated to the imager previously developed in the CSL's lab that produced most of the data processed in the different series of tests. A qualitative review of images deconvolution, performed by using either a gaussian distribution or a measured point spread function, was additionally provided.

Subsequently, in a chapter dedicated to applications, the algorithms developed in the current work have been applied in two different cases. The first one consisted of processing damaged scenes that were acquired beforehand with the CSL imaging device, the latter being devoted to compressive sensing studies thanks to its capability to adequately mask the observations with its DMD component. The importance of the mask's eroded pattern, as well as the influence of the NOI and FR parameters on the quality of the reconstructions results, were therefore highlighted. The second application case was derived from a bibliographic study that allowed to find real satellite data produced during the Landsat 7 space mission. Indeed, this NASA mission, focused on Earth observation, generated incomplete scientific data due to one of its satellite component's failure occurring in the nominal phase of the mission. The capability of the tested images reconstructions method to restore, to some extent, true colour images coming from remote sensing acquisitions, were finally demonstrated.

5.2 Perspectives

This very last section gathers some prospects that could constitute a possible starting point for the realisation of future works in the context of images processing-related problems, on the basis of what is covered in this master thesis.

First of all, additional modifications could be implemented into the IHT algorithm's digital code to complete its optimisation. As an example, it could be feasible to rewrite the whole code so that the images reconstructions would not be limited to the precise case of square images, avoiding this way to perform the zero padding technique developed here and therefore reduce the computation time. This computation time could also be shortened by inserting in the digital code the open source Python's Numba compiler [26], that aims to speed up the steps of coded functions or loops, and is especially designed for scientific computing. This would however imply a global reorganisation of the code's functions. Another option to improve the algorithm would be to automate the production of the images colorbars for each of the obtained results, so that additional manipulations would not be necessary after images reconstructions.

After that, the trial and error method selected for the image deconvolution using a PSF is clearly not optimised as well, and it would be appreciable to determine a more precise technique that would allow, in addition, to completely deconvolve the initial rectangular GT and not partially as it is the case in the present work. Besides that, it could be pertinent to check what kind of results could be obtained when deconvolving images produced by an optical setup before introducing them in the IHT reconstruction algorithm.

Finally, the study of the Landsat 7 mission's data is limited to true colour images reconstructions and it would therefore be interesting to test the followed procedure for other available spectral bands. This master thesis however prepare the ground for forthcoming works in the frame of space mission data analyses, notably in the domain of satellite-based Earth observation and monitoring.

Chapter 6

Bibliography

[1] C. Charles, “Introduction aux problèmes inverses.” University of Liège, Gembloux Agro-Bio Tech, 2014.

[2] L. Jacques and C. De Vleeschouwer, “Sparsity principles and applications: from orthonormal bases to redundant systems.” Lecture notes, UCLouvain.

[3] L. Jacques, “Wavelet theory & Multi-Resolution Analysis.” Lecture notes, UCLouvain.

[4] Jupyter Notebooks, “Inpainting using Sparse Regularization.” `https://nbviewer.org/github/gpeyre/numerical-tours/blob/master/python/inverse_5_inpainting_sparsity.ipynb`. [Online; last consultation on 14/08/2024].

[5] Jupyter Notebooks, “Image Deconvolution using Sparse Regularization.” `https://nbviewer.org/github/gpeyre/numerical-tours/blob/master/matlab/inverse_3_deconvolution_sparsity.ipynb`. [Online; last consultation on 14/08/2024].

[6] J. Loicq, “Space experiment development - Chapter 5: Detectors.” Lecture notes, University of Liège, 2024.

[7] Mathworks, “Display Separated Color Channels of RGB Image.” `https://nl.mathworks.com/help/images/display-separated-color-channels-of-rgb-image.html`. [Online; last consultation on 14/08/2024].

[8] R. A. Horn and C. R. Johnson, “Chapter 5 - The Hadamard product,” *Topics in Matrix Analysis*, p. 298, 1991.

[9] S. Foucart and H. Rauhut, “A Mathematical Introduction to Compressive Sensing,” *Springer Science+Business Media New York, Birkhäuser*, 2013.

[10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, “Image Coding Using Wavelet Transform,” *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 1, no. 2, 1992.

[11] C. Thomas, “Exploring Compressive Sensing for Earth Observation.” Master thesis, University of Liège, 2023 (Unpublished).

[12] L. Jacques, L. Duval, C. Chaux, and G. Peyré, "A Panorama on Multiscale Geometric Representations, Intertwining Spatial, Directional and Frequency Selectivity," *Signal Processing*, vol. 91, May 2018.

[13] J. Loicq, "Space optics - Chapter 1: Introduction & Optics prerequisite." Lecture notes, University of Liège, 2023.

[14] T. Dahmen, *Tomographic reconstruction of combined tilt- and focal series in scanning transmission electron microscopy.* PhD thesis, 01 2015. Universität des Saarlandes.

[15] SciPy, "SciPy API - Linear algebra - circulant." `https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.circulant.html`. [Online; last consultation on 14/08/2024].

[16] H. Khalil, "Matrices structurées et matrices de Toeplitz par blocs de Toeplitz en calcul numérique et formel." PhD thesis, Université Claude Bernard - Lyon, July 2008.

[17] E. Ruwet, "Improvement of the optical setup of a compressive sensing imager." Master thesis, University of Liège, 2024 (Unpublished).

[18] Emerson - National Instruments Corporation, "Peak Signal-to-Noise Ratio as an Image Quality Metric." `https://www.ni.com/en/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-vision-development-module/peak-signal-to-noise-ratio-as-an-image-quality-metric.html`. [Online; last consultation on 15/08/2024].

[19] USGS - Landsat missions, "Landsat 7." `https://www.usgs.gov/landsat-missions/landsat-7`. [Online; last consultation on 15/08/2024].

[20] USGS - Landsat missions, "Landsat 7 Extended Science Mission." `https://www.usgs.gov/landsat-missions/landsat-7-extended-science-mission`. [Online; last consultation on 15/08/2024].

[21] M. R. Radcliff - NASA Scientific Visualization Studio, "Landsat sensors: pushbroom vs whiskbroom." `https://svs.gsfc.nasa.gov/12754`. [Online; last consultation on 15/08/2024].

[22] K. Yifei, P. Li, S. Mingwei, L. Xinyi, and C. Qi, "Destriping high-resolution satellite imagery by improved moment matching," *INTERNATIONAL JOURNAL OF REMOTE SENSING*, vol. 38, p. 6346–6365, June 2017.

[23] K. Sayler, Department of the Interior U.S. Geological Survey, "Landsat 7 (L7) Data Users Handbook Version 3.0," *LSDS-1927*, May 2024.

[24] USGS, "Earth Explorer data portal." `https://earthexplorer.usgs.gov/`. [Online; last consultation on 15/08/2024].

[25] Escape Velocity Labs, "Digital art in Python: decomposing an image in its color channels." `https://medium.com/@evlabs/digital-art-in-python-decomposing`

-an-image-in-its-color-channels-1cc0da56a81e. [Online; last consultation on 15/08/2024].

[26] Anaconda navigator, “Numba makes Python code fast.” https://numba.pydata.org/. [Online; last consultation on 15/08/2024].

[27] Centre spatial guyanais, “Ariane 5.” https://centrespatialguyanais.cnes.fr/fr/centre-spatial-guyanais/lanceurs-et-installations/les-lanceurs/ariane-5. [Online; last consultation on 14/08/2024].

[28] ULiège, “Centre spatial de Liège.” https://www.csl.uliege.be/cms/c_10241774/en/csl. [Online; last consultation on 14/08/2024].

[29] Centre spatial de Liège, “PLATO mission gets green light for its next phase.” https://www.csl.uliege.be/cms/c_15530036/en/plato-mission-gets-green-light-for-its-next-phase. [Online; last consultation on 14/08/2024].

[30] F.G. Baptista, K. Berne and E. Conant, “How flight suits have evolved to keep astronauts safe in space - National Geographic magazine.” https://www.nationalgeographic.com/magazine/article/suiting-up. [Online; last consultation on 14/08/2024].

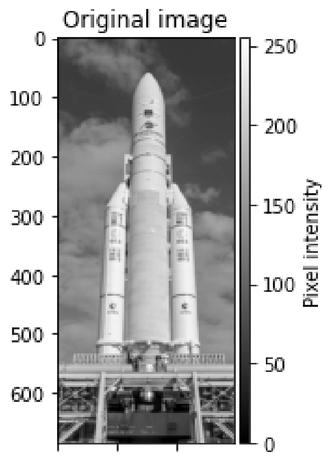
Appendix A

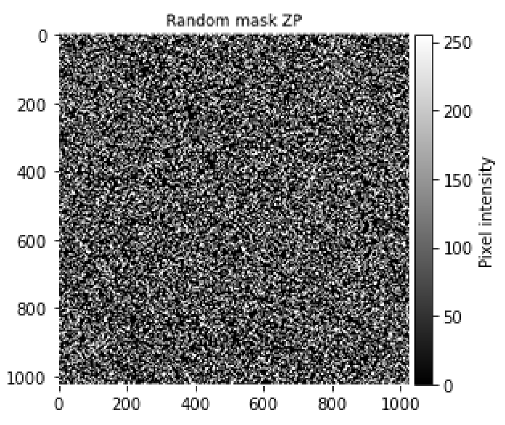
Reconstruction images

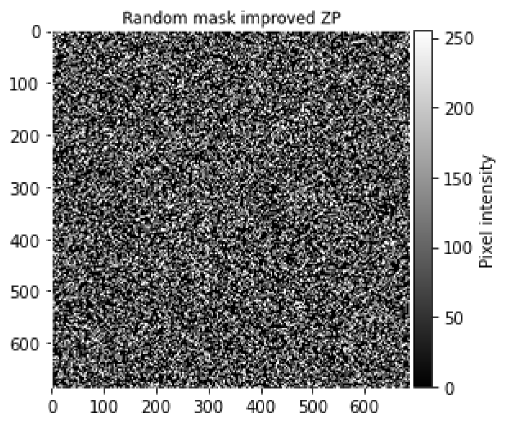
A.1 Zero padding

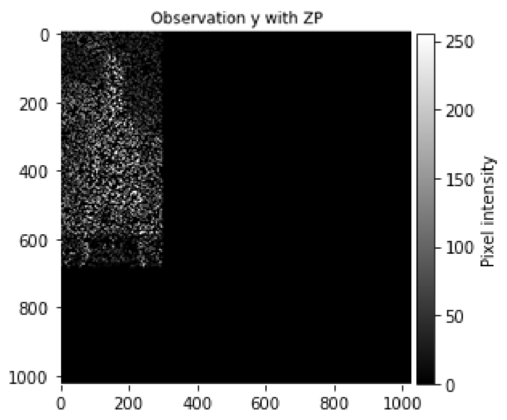
All the reconstructions displayed in this section were performed by using randomly generated masks (FR = 70%), a NOI of 500 and a decaying value of the λ parameter. Original images are taken and adapted from [27], [28], [29] and [30].

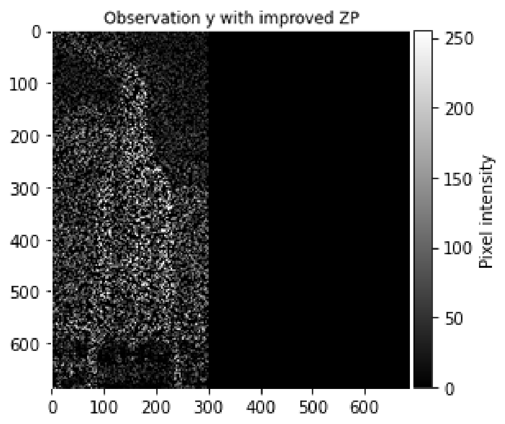
A.1.1 Ariane

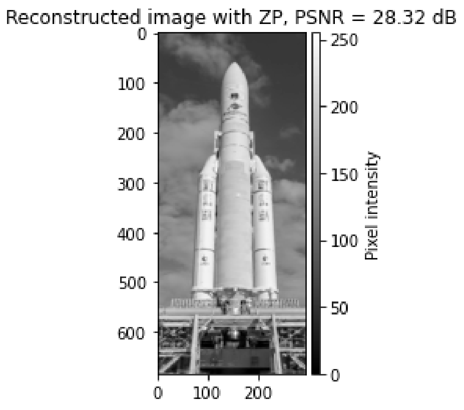


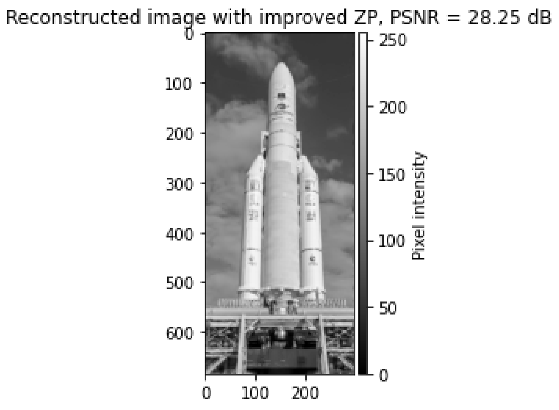




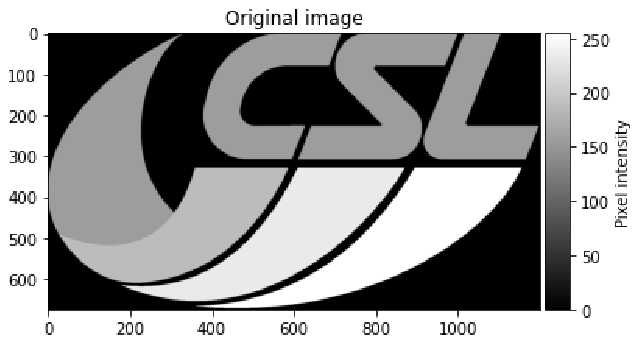

Observation y with ZP
Pixel intensity

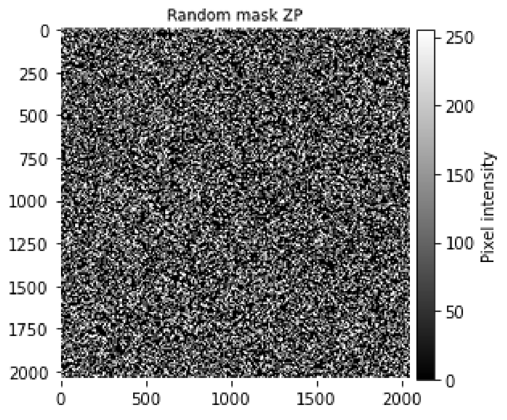

Observation y with improved ZP
Pixel intensity

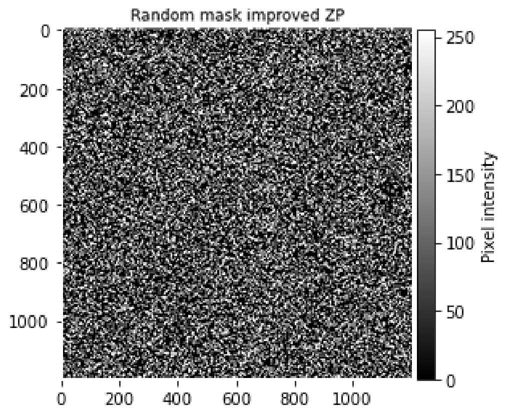

Reconstructed image with ZP, PSNR = 28.32 dB
Pixel intensity

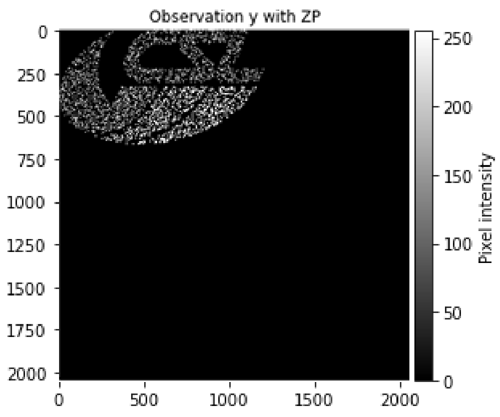

Reconstructed image with improved ZP, PSNR = 28.25 dB
Pixel intensity

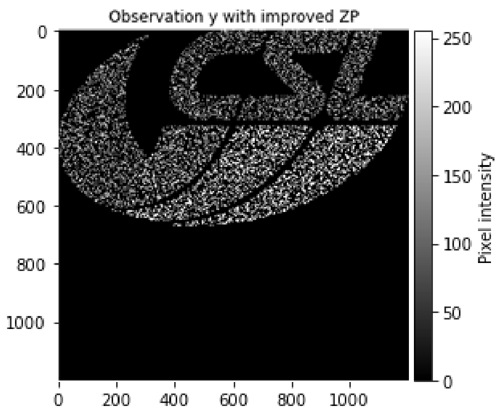
A.1.2 CSL

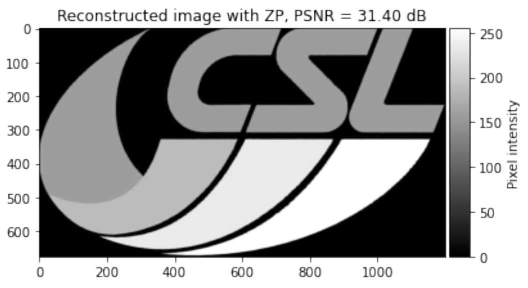

Original image
Pixel intensity

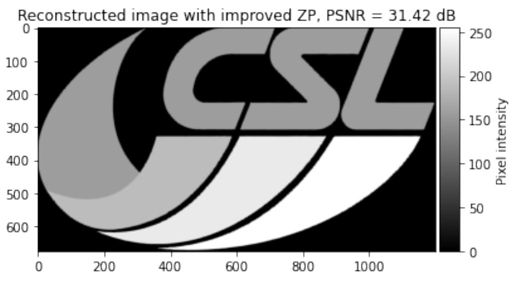

Random mask ZP
Pixel intensity


Random mask improved ZP
Pixel intensity

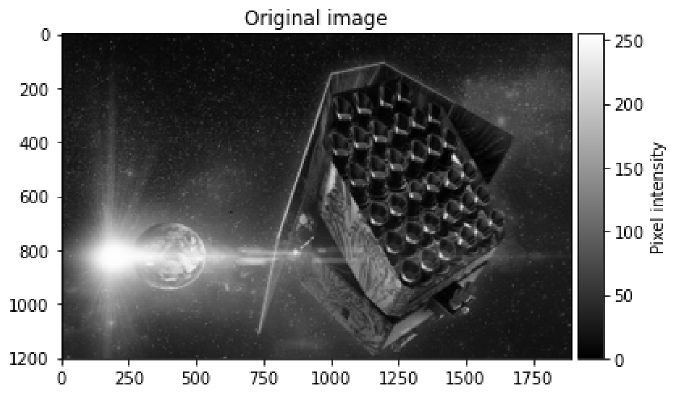

Observation y with ZP
Pixel intensity

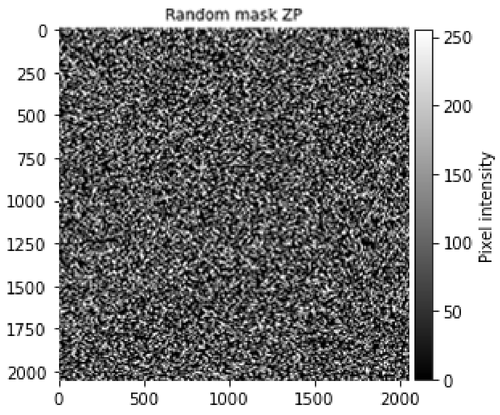

Observation y with improved ZP
Pixel intensity

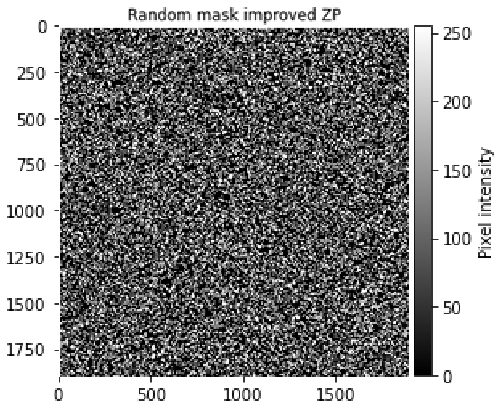

Reconstructed image with ZP, PSNR = 31.40 dB
Pixel intensity

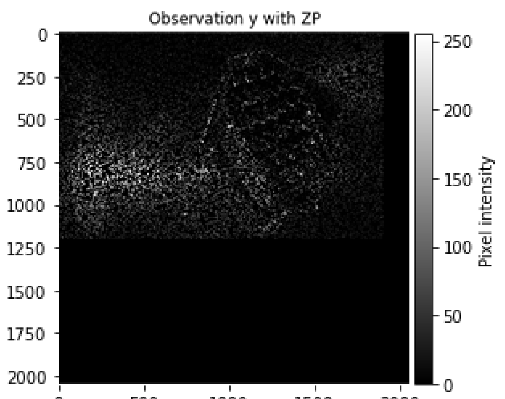

Reconstructed image with improved ZP, PSNR = 31.42 dB
Pixel intensity

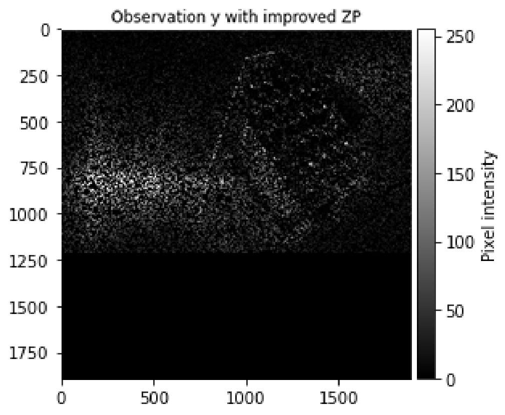
A.1.3 PLATO

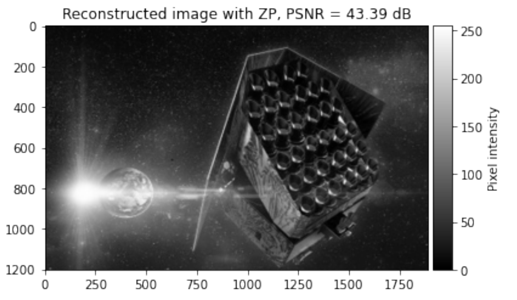

Original image
Pixel intensity

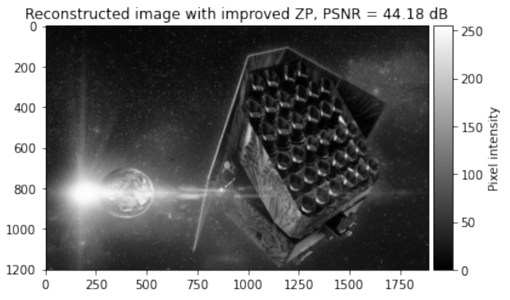

Random mask ZP
Pixel intensity


Random mask improved ZP
Pixel intensity

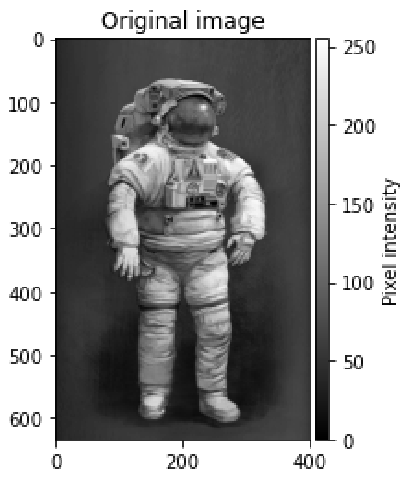


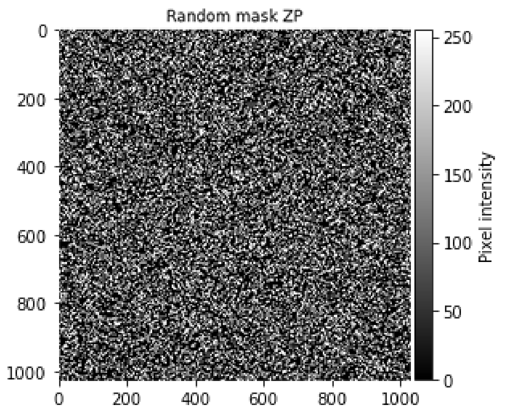


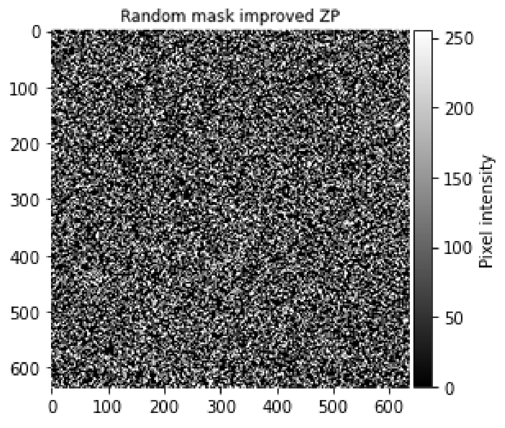


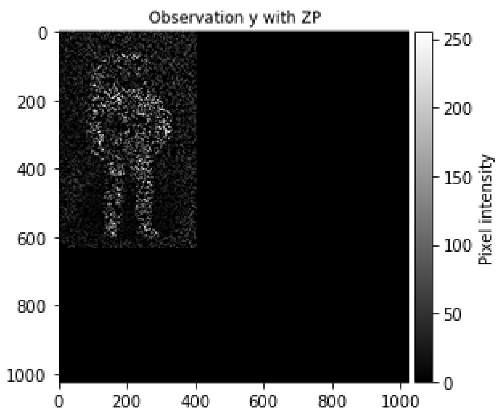


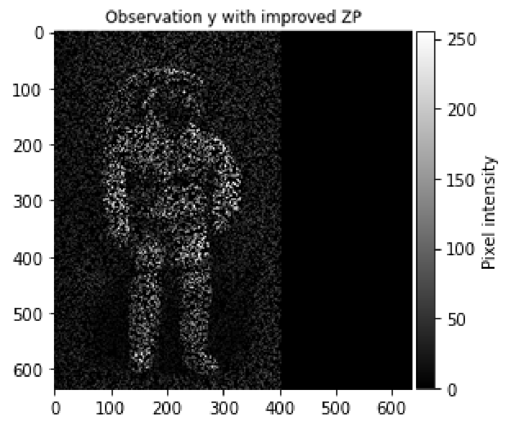
A.1.4 Spacesuit

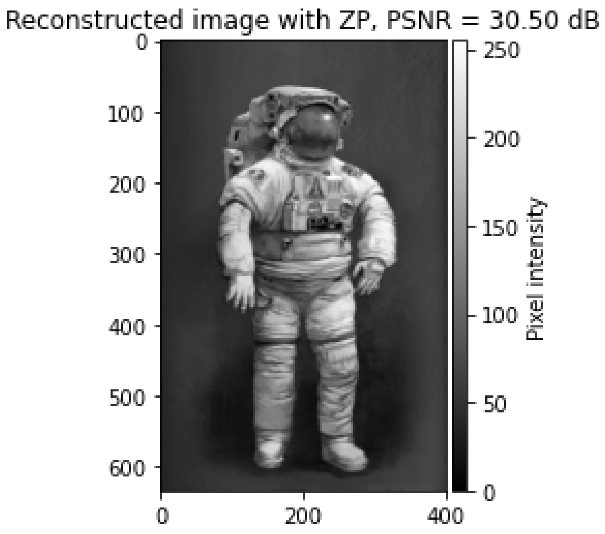


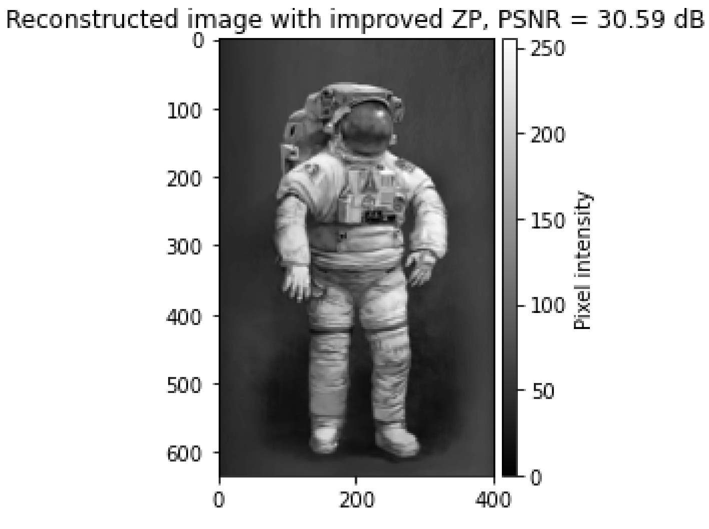





Observation y with ZP
0
200
400
600
800
1000
Pixel intensity
250
200
150
100
50

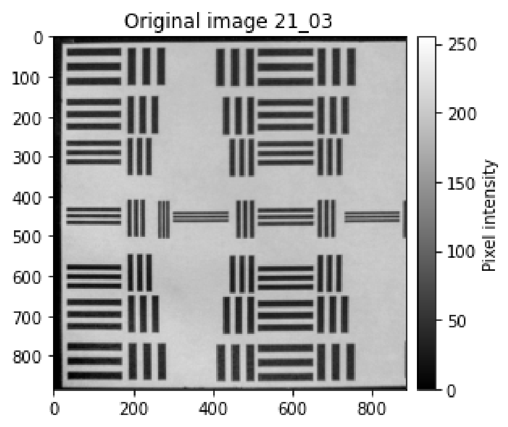

Observation y with improved ZP
0
100
200
300
400
500
600
Pixel intensity
250
200
150
100
50

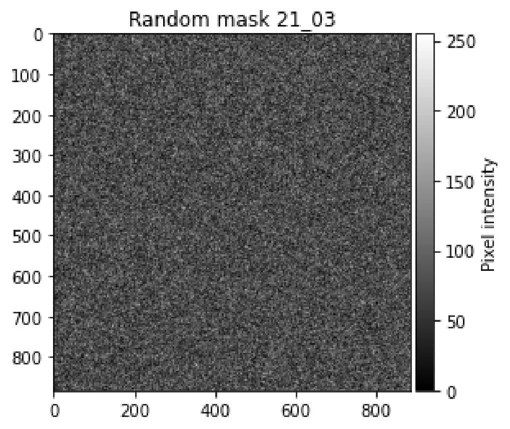

Reconstructed image with ZP, PSNR = 30.50 dB
0
100
200
300
400
500
600
Pixel intensity
250
200
150
100
50

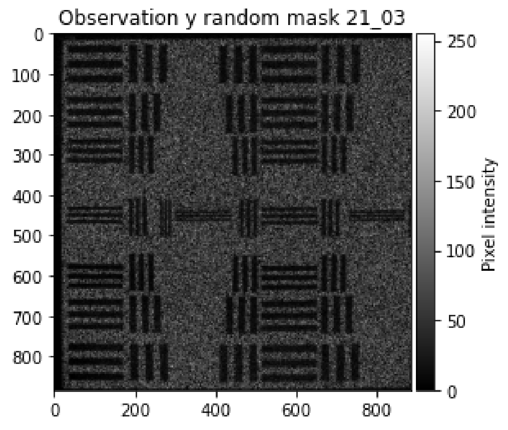

Reconstructed image with improved ZP, PSNR = 30.59 dB
0
100
200
300
400
500
600
Pixel intensity
250
200
150
100
50

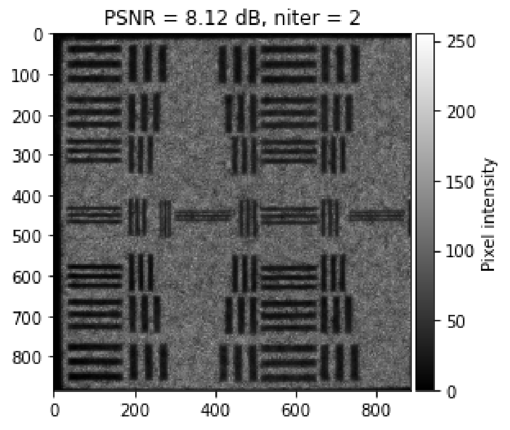
A.2 Reconstructions based on an observed scene

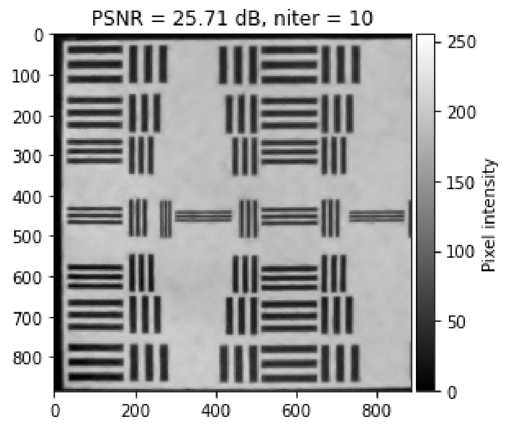
A.2.1 Random mask

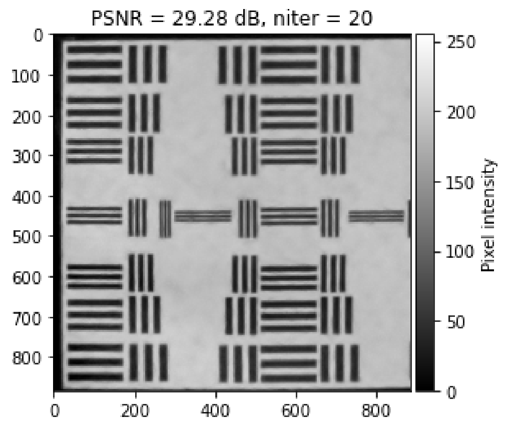


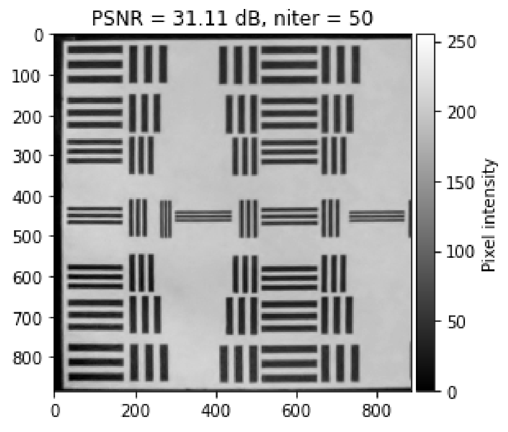


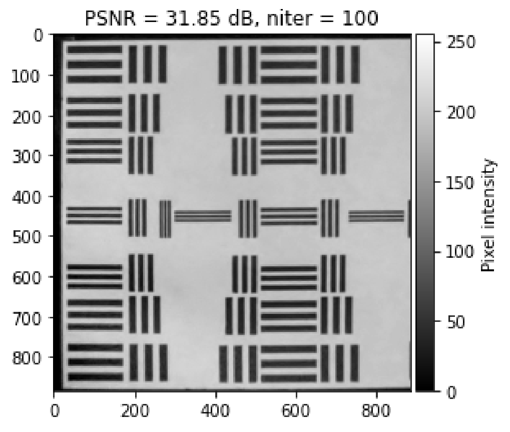


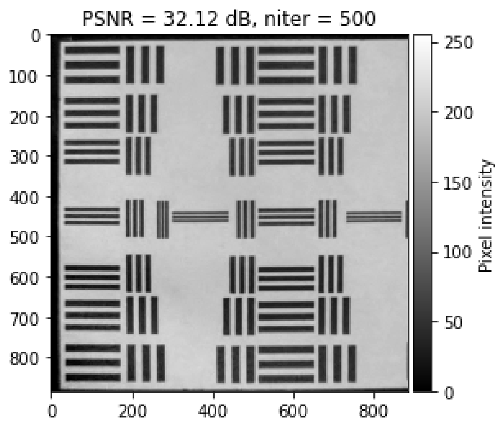


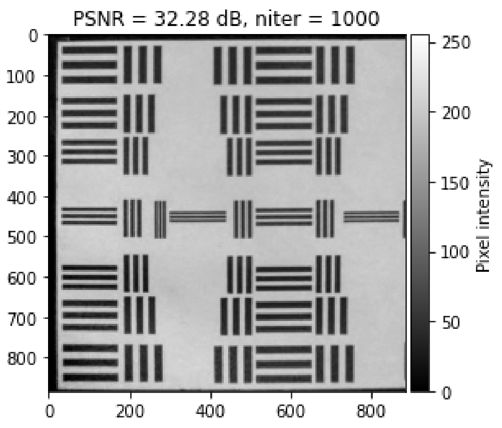


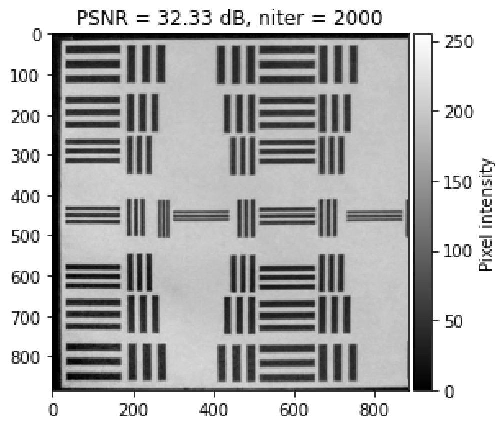




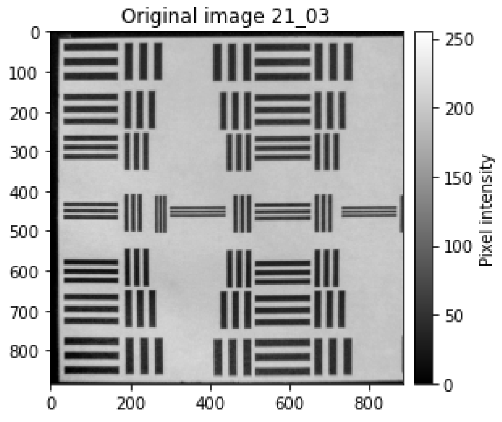


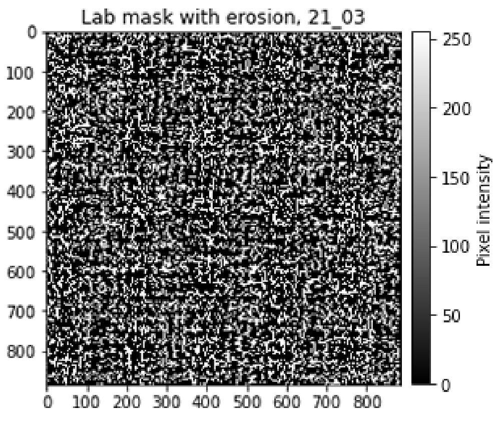


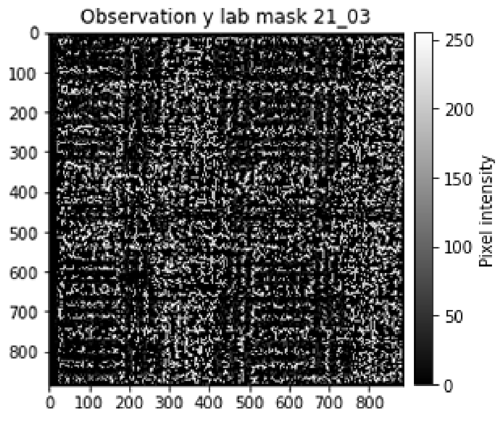


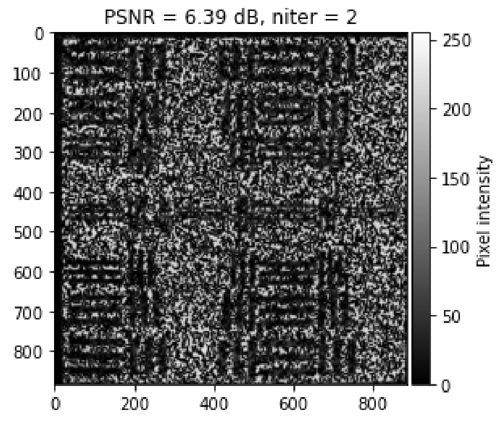


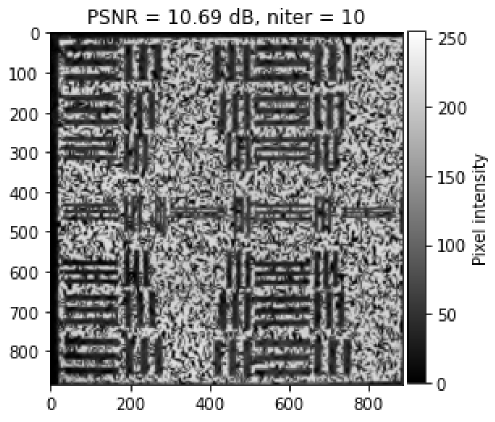
A.2.2 Lab mask 21/03

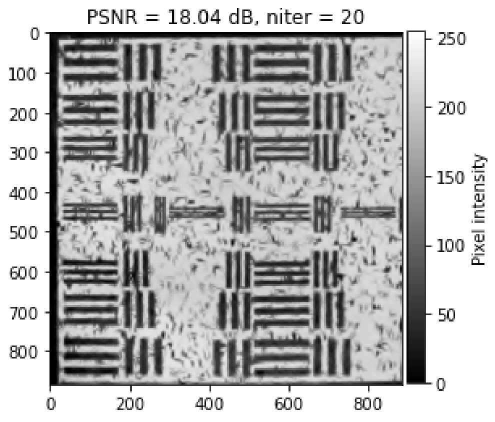


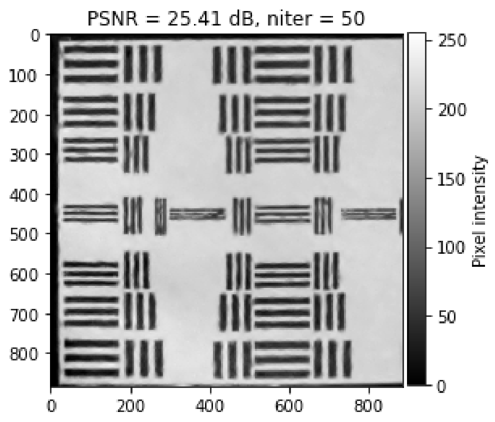


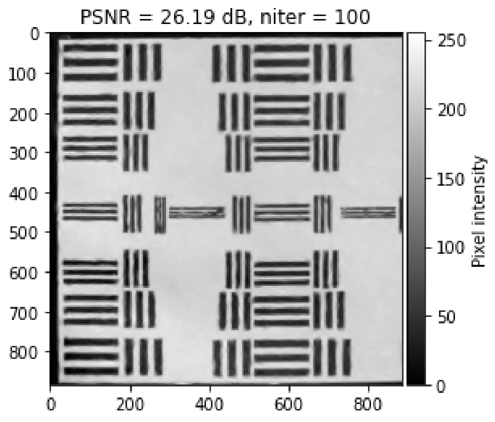


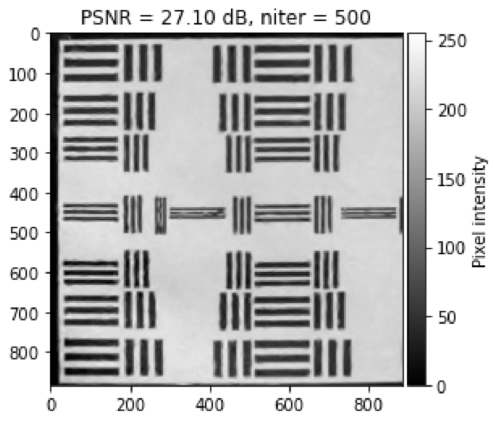


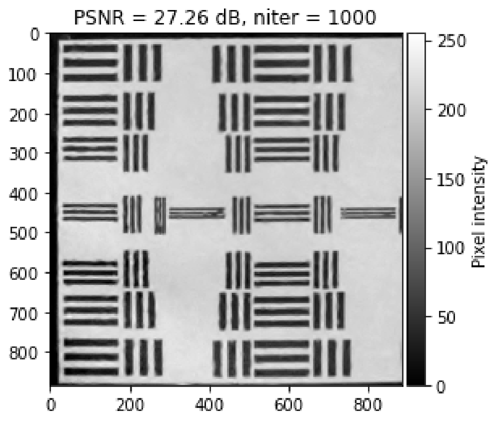

PSNR = 10.69 dB, niter = 10
Pixel intensity

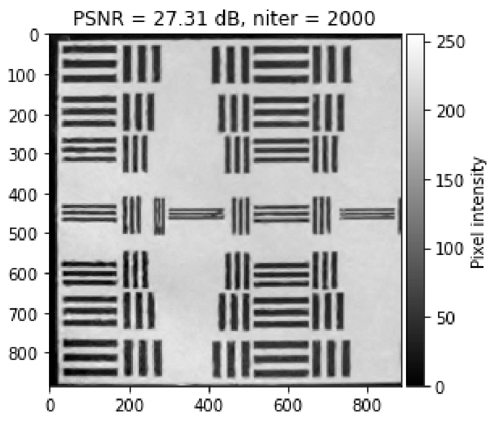

PSNR = 18.04 dB, niter = 20
Pixel intensity


PSNR = 25.41 dB, niter = 50
Pixel intensity


PSNR = 26.19 dB, niter = 100
Pixel intensity

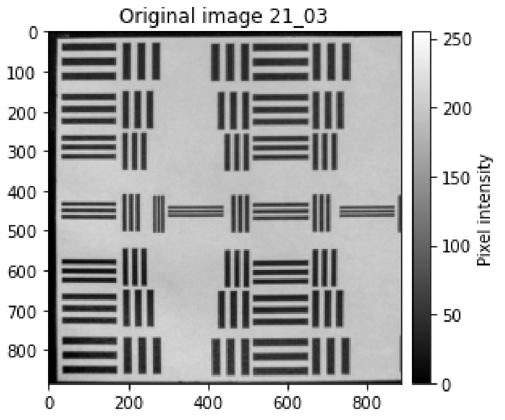

PSNR = 27.10 dB, niter = 500
Pixel intensity

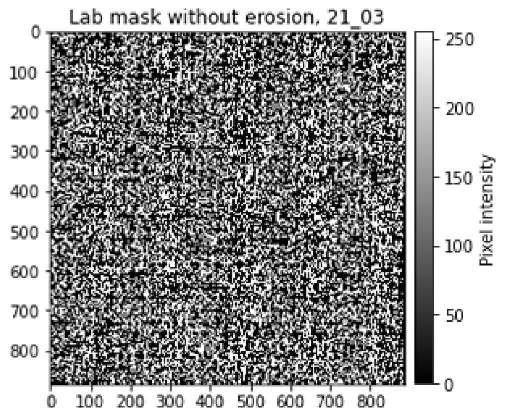

PSNR = 27.26 dB, niter = 1000
Pixel intensity

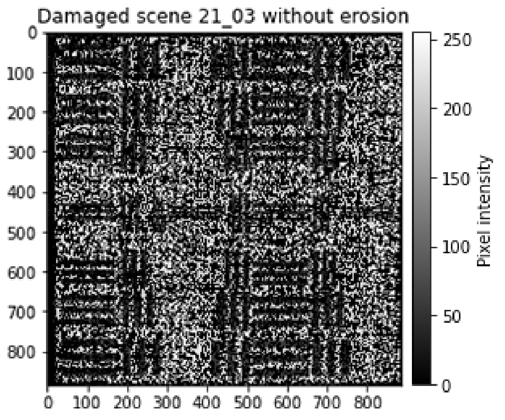

PSNR = 27.31 dB, niter = 2000
Pixel intensity

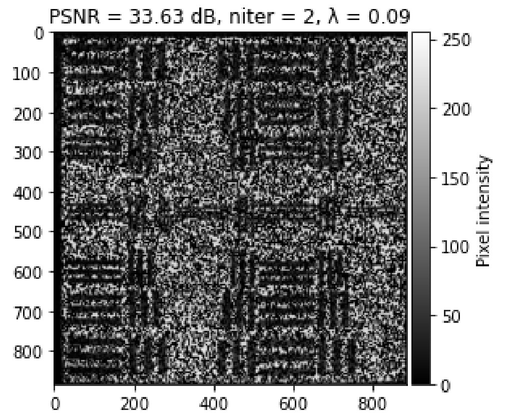
A.3 Comparison between non-eroded and eroded masks

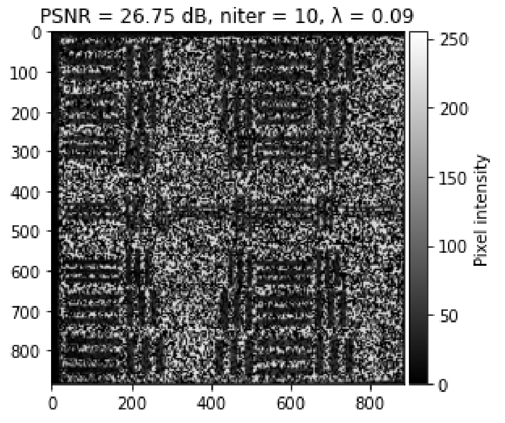
A.3.1 Mask without erosion

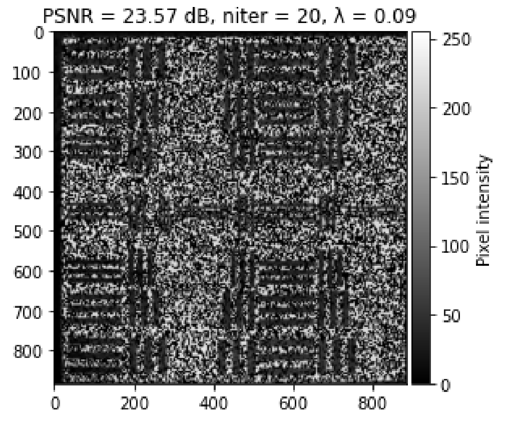


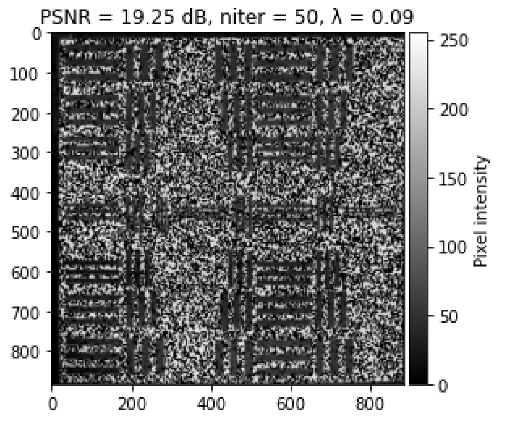


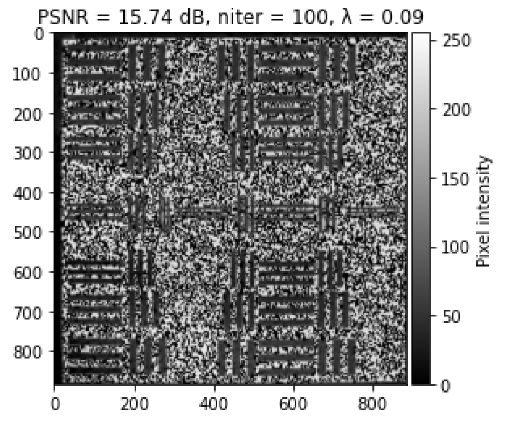


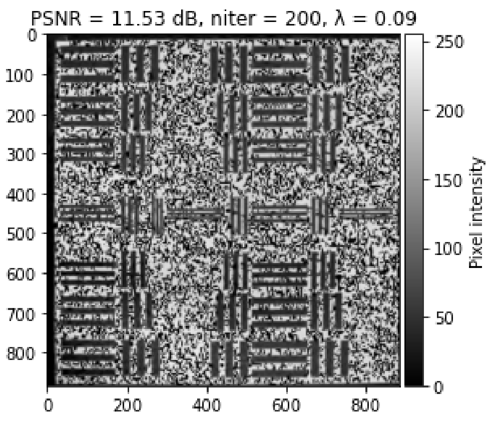


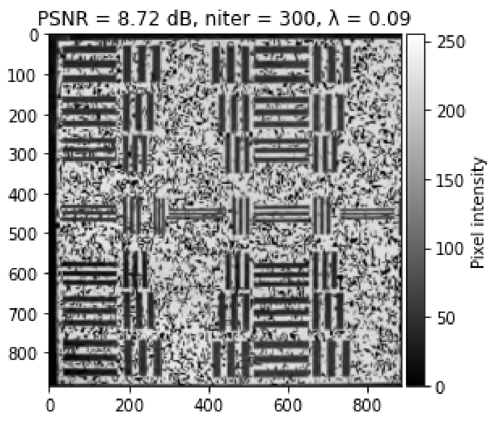


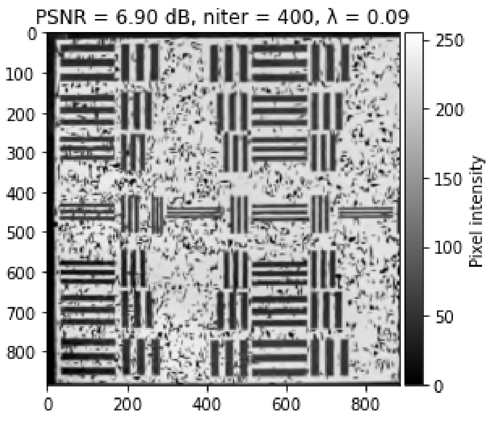


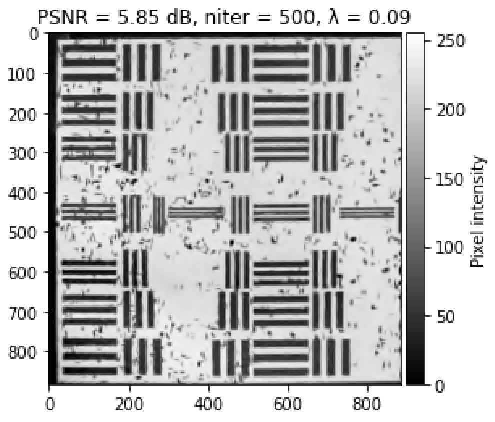


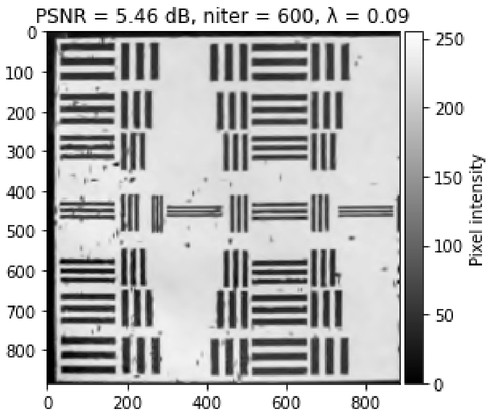


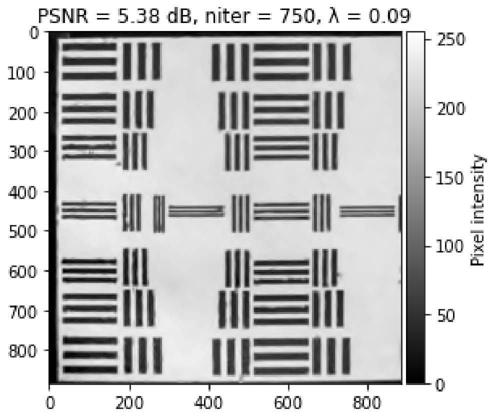

PSNR = 11.53 dB, niter = 200, λ = 0.09
Pixel intensity

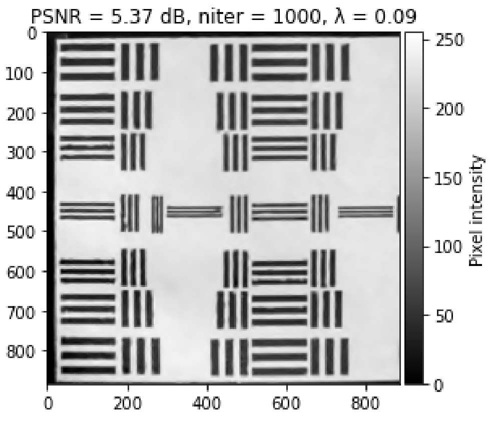

PSNR = 8.72 dB, niter = 300, λ = 0.09
Pixel intensity

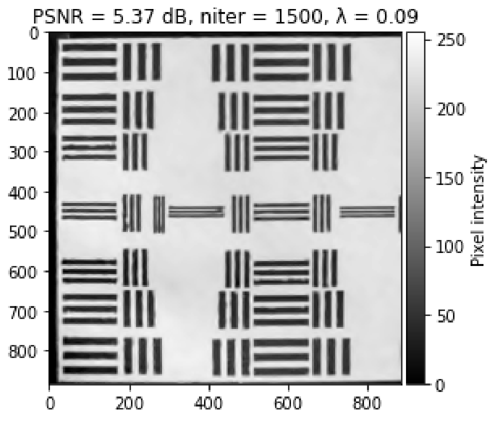

PSNR = 6.90 dB, niter = 400, λ = 0.09
Pixel intensity

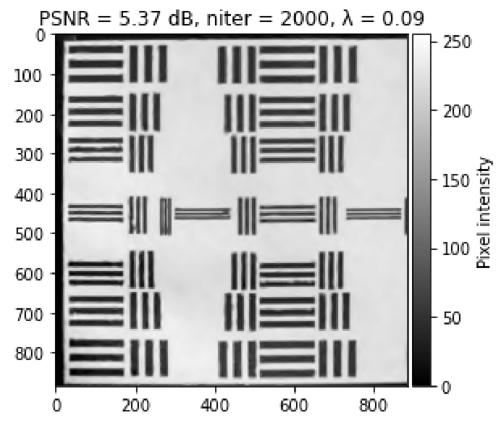

PSNR = 5.85 dB, niter = 500, λ = 0.09
Pixel intensity


PSNR = 5.46 dB, niter = 600, λ = 0.09
Pixel intensity

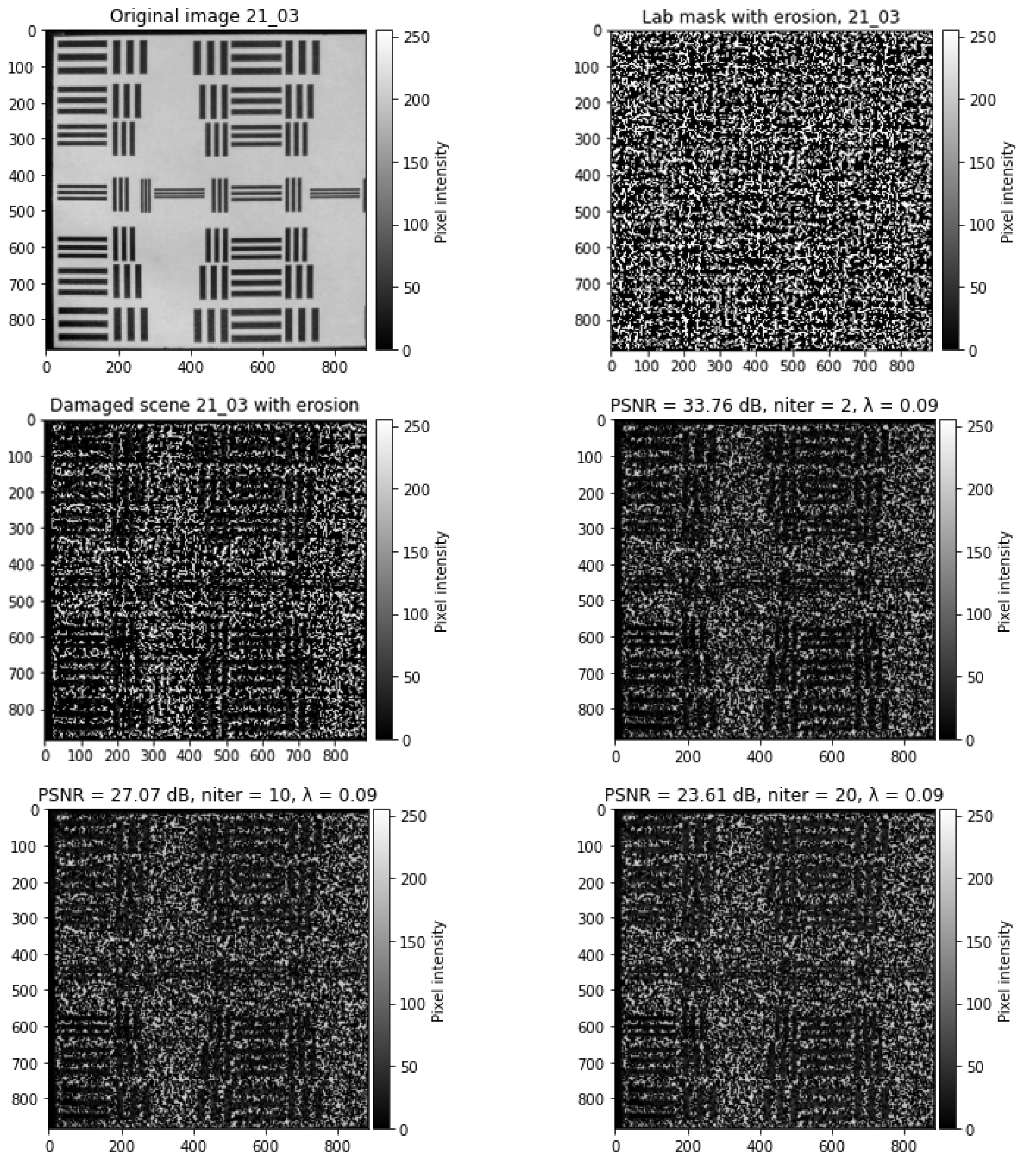

PSNR = 5.38 dB, niter = 750, λ = 0.09
Pixel intensity

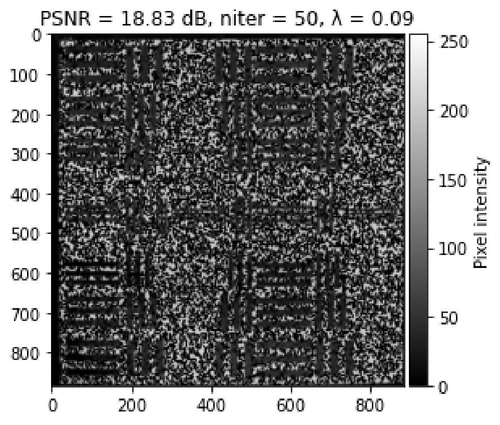

PSNR = 5.37 dB, niter = 1000, λ = 0.09
Pixel intensity

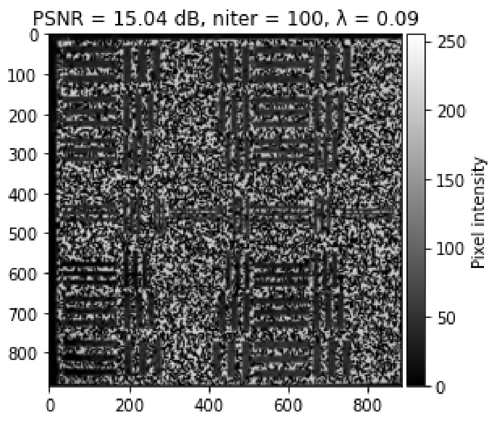

PSNR = 5.37 dB, niter = 1500, λ = 0.09
Pixel intensity

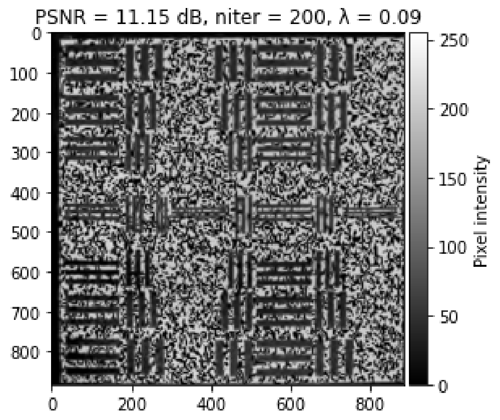


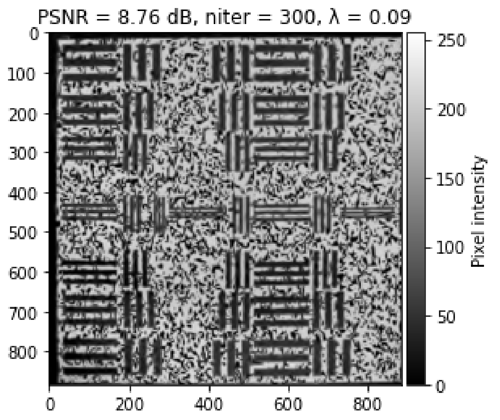
A.3.2 Mask with erosion

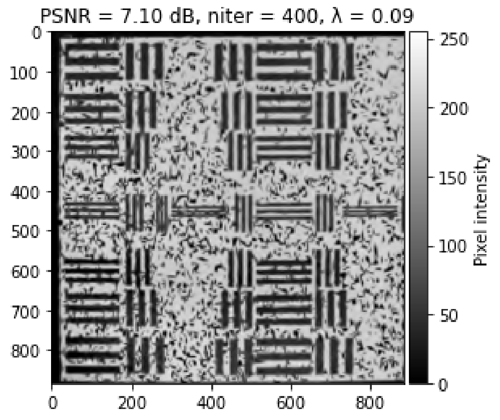


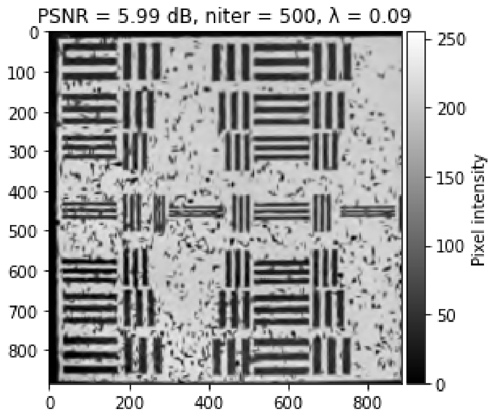

PSNR = 18.83 dB, niter = 50, λ = 0.09
Pixel intensity

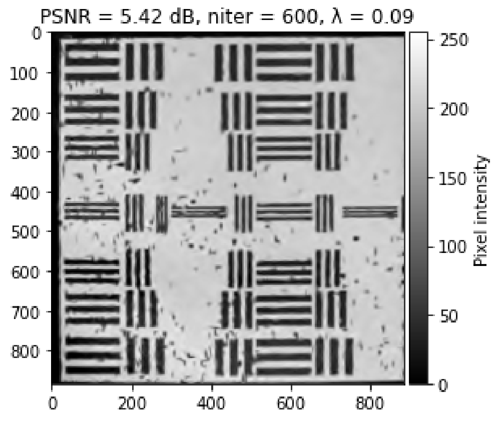

PSNR = 15.04 dB, niter = 100, λ = 0.09
Pixel intensity

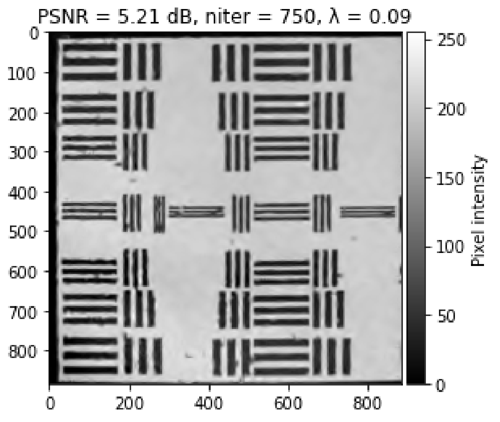

PSNR = 11.15 dB, niter = 200, λ = 0.09
Pixel intensity

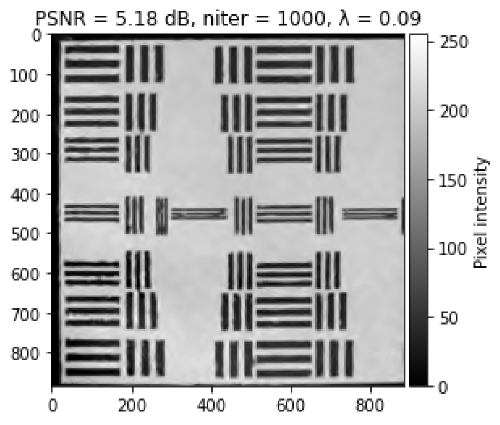

PSNR = 8.76 dB, niter = 300, λ = 0.09
Pixel intensity

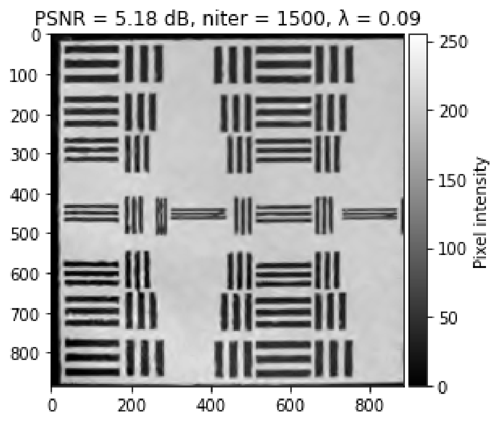

PSNR = 7.10 dB, niter = 400, λ = 0.09
Pixel intensity

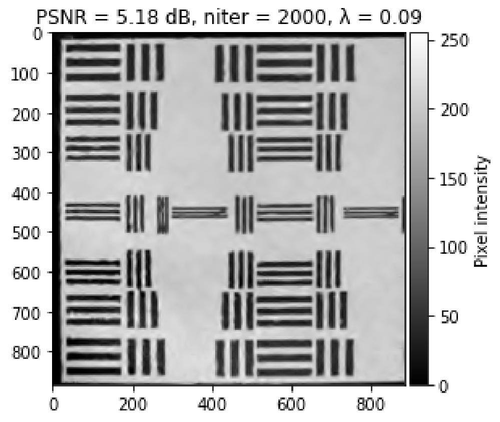

PSNR = 5.99 dB, niter = 500, λ = 0.09
Pixel intensity


PSNR = 5.42 dB, niter = 600, λ = 0.09
Pixel intensity


PSNR = 5.21 dB, niter = 750, λ = 0.09
Pixel intensity

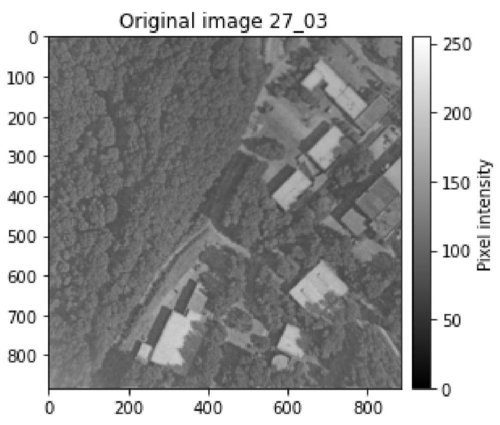

PSNR = 5.18 dB, niter = 1000, λ = 0.09
Pixel intensity

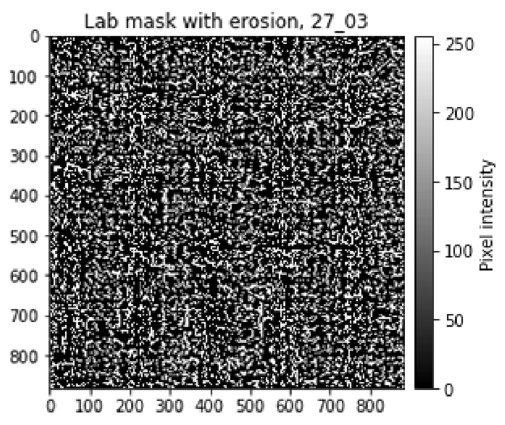

PSNR = 5.18 dB, niter = 1500, λ = 0.09
Pixel intensity

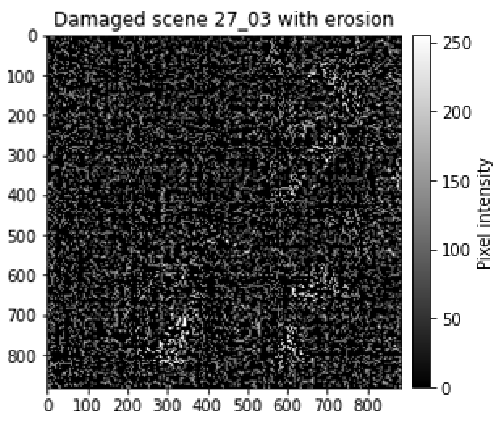

PSNR = 5.18 dB, niter = 2000, λ = 0.09
Pixel intensity

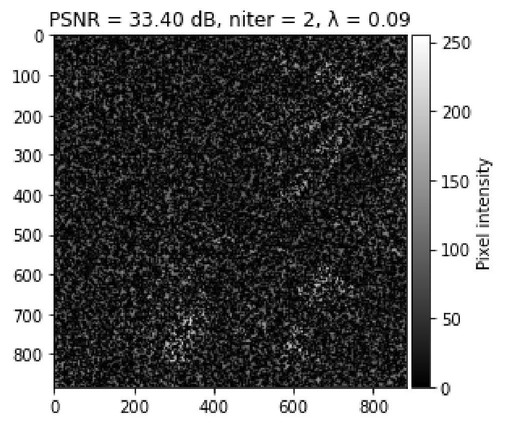
A.4 Extension of the mask filling ratio

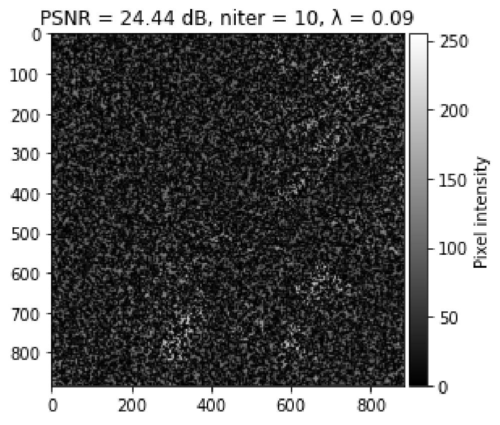
A.4.1 Mask with erosion 70% 27/03

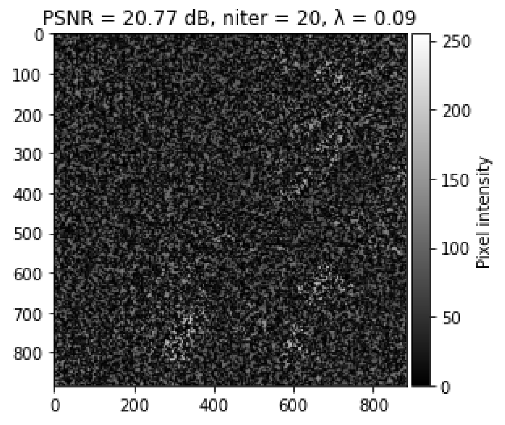


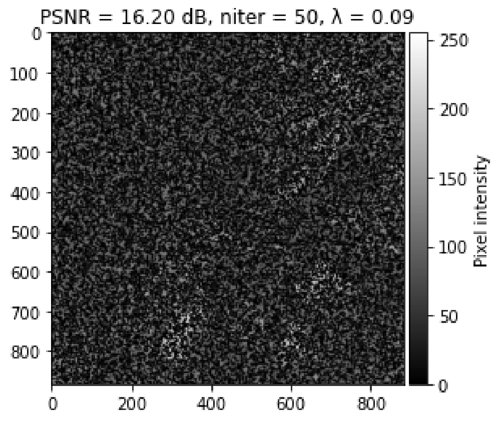


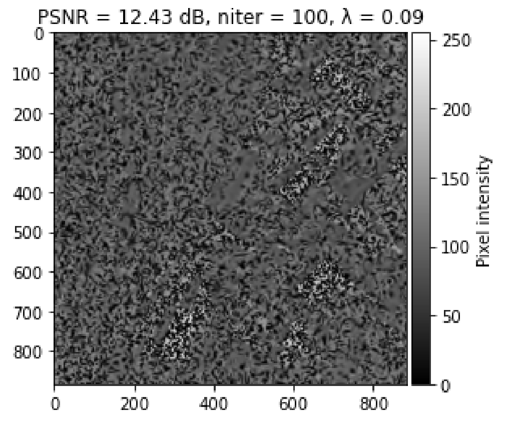


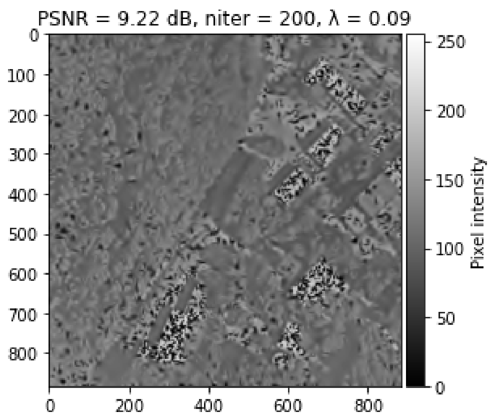


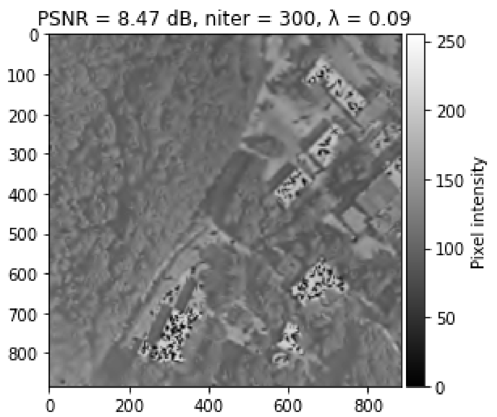


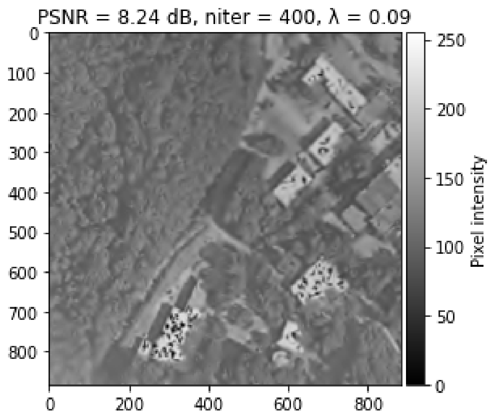


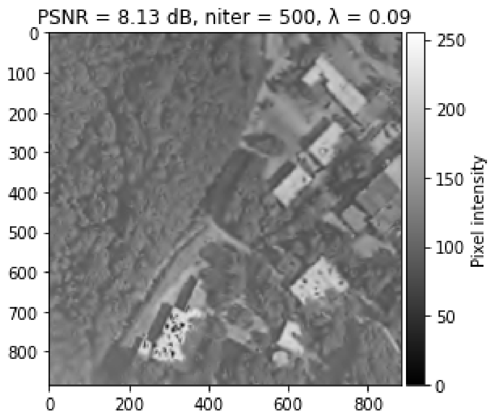


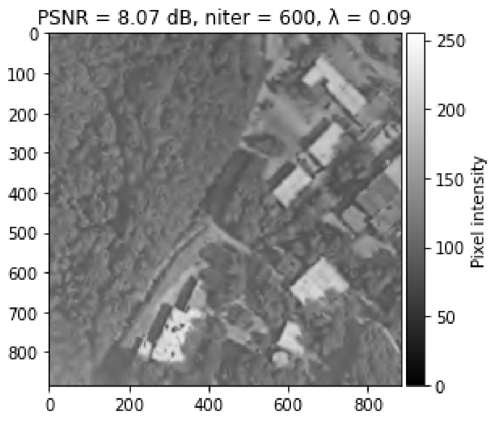


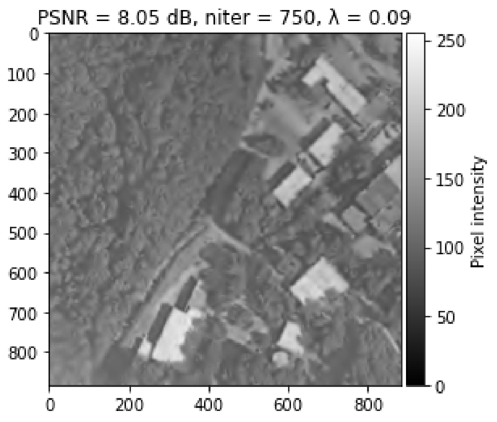

PSNR = 9.22 dB, niter = 200, λ = 0.09
Pixel intensity

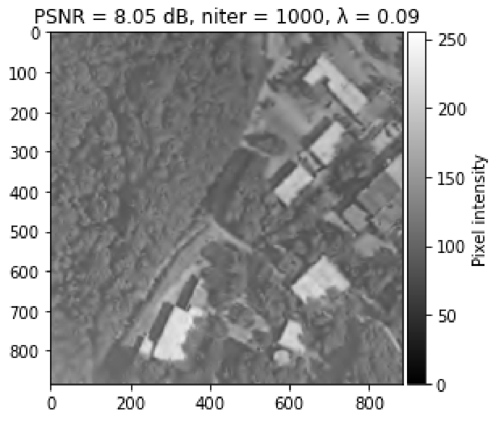

PSNR = 8.47 dB, niter = 300, λ = 0.09
Pixel intensity


PSNR = 8.24 dB, niter = 400, λ = 0.09
Pixel intensity

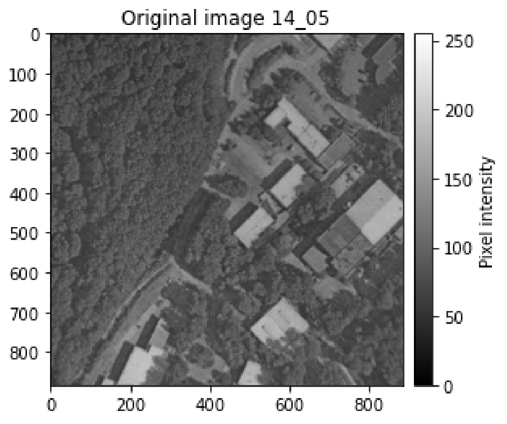

PSNR = 8.13 dB, niter = 500, λ = 0.09
Pixel intensity

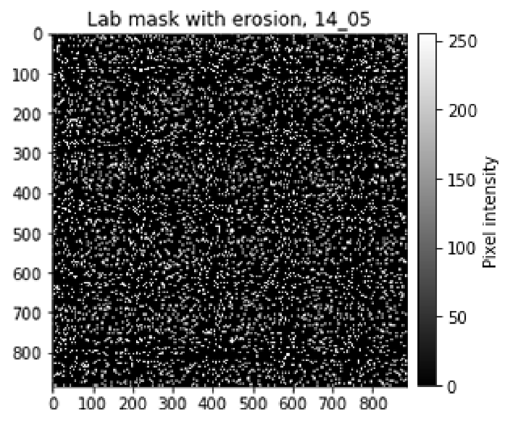

PSNR = 8.07 dB, niter = 600, λ = 0.09
Pixel intensity

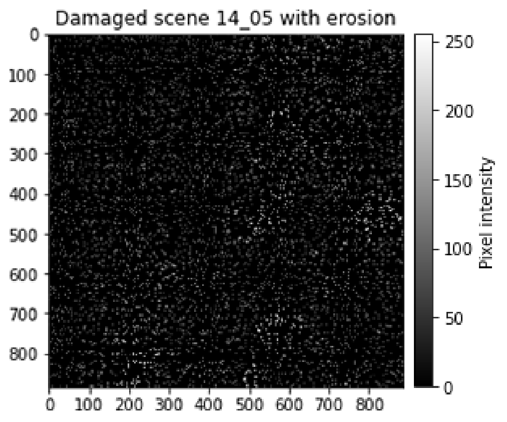

PSNR = 8.05 dB, niter = 750, λ = 0.09
Pixel intensity

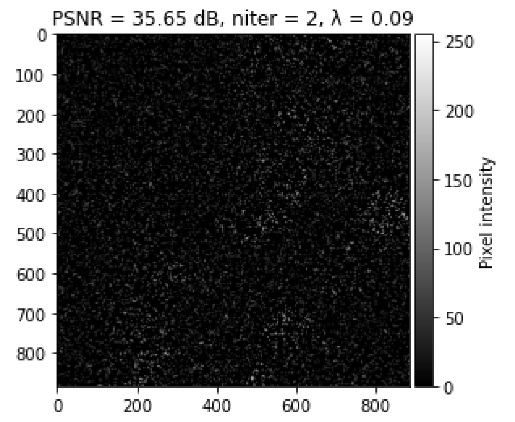

PSNR = 8.05 dB, niter = 1000, λ = 0.09
Pixel intensity

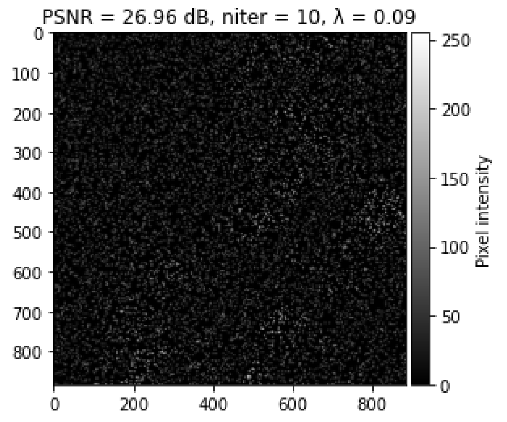
A.4.2 Mask with erosion 90% 14/05

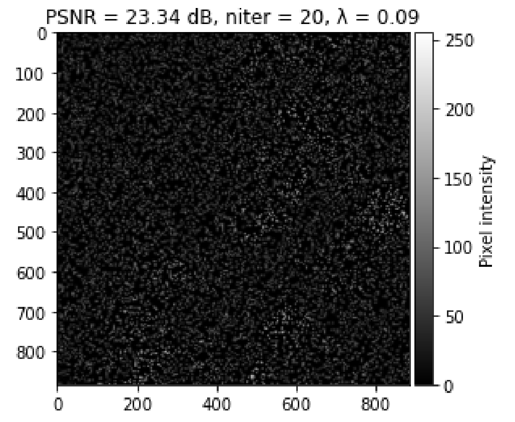


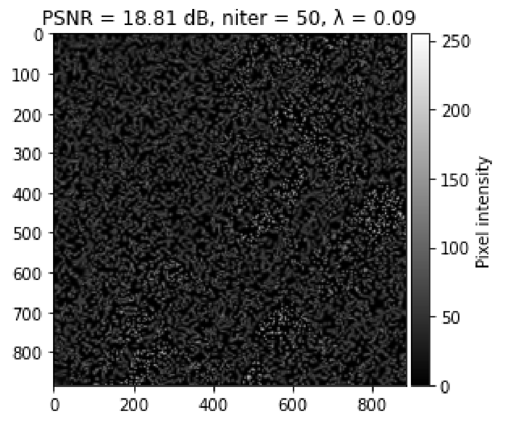


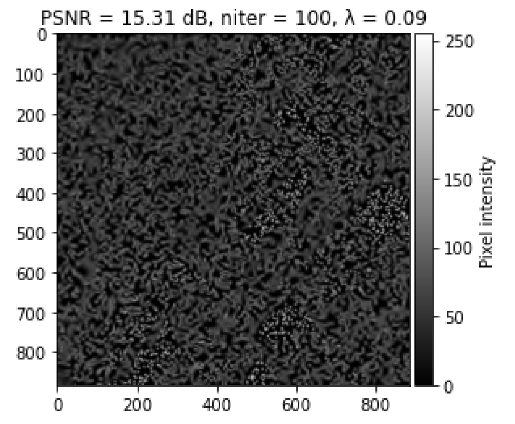


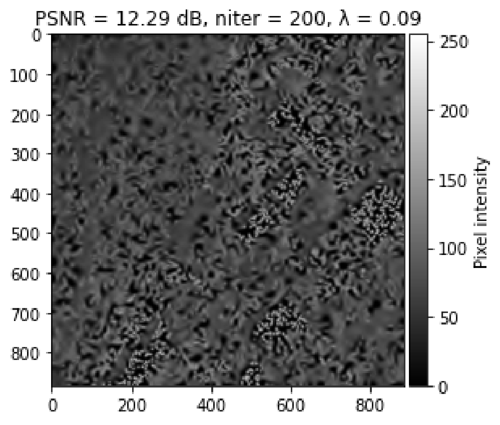


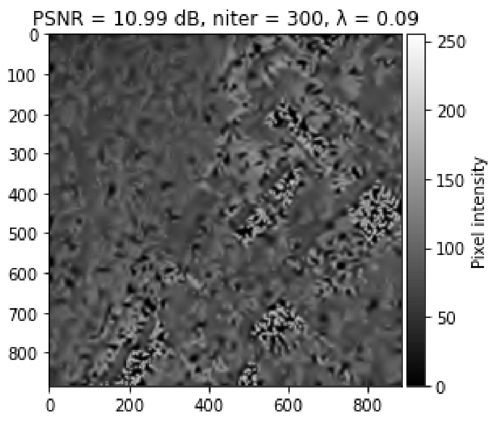


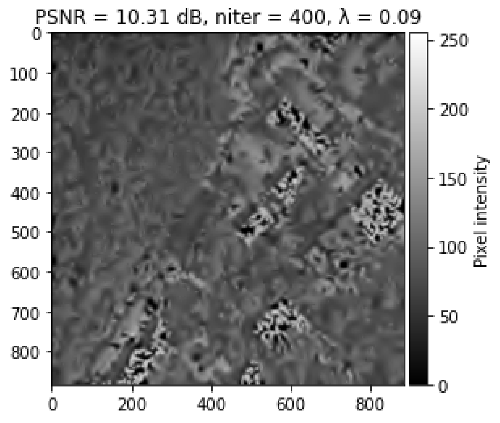


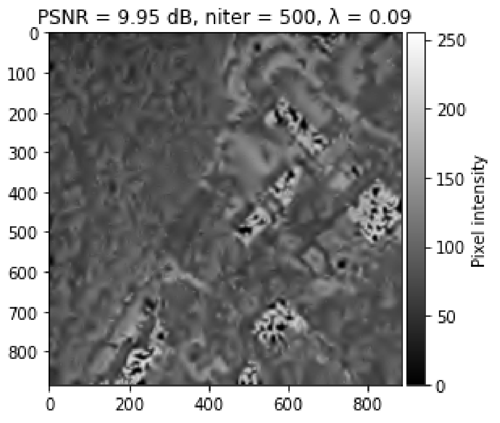


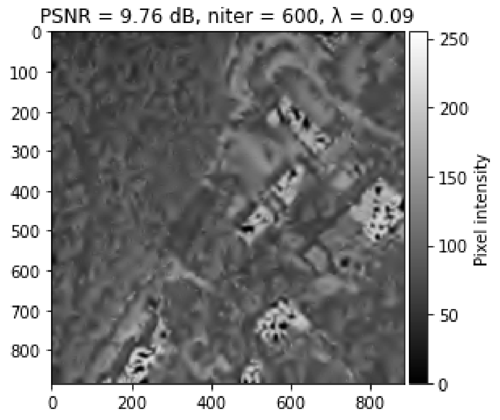


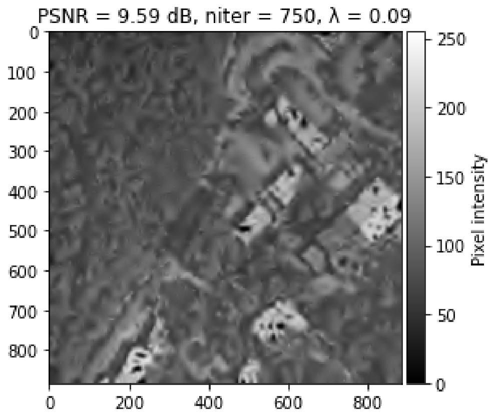

PSNR = 12.29 dB, niter = 200, λ = 0.09
Pixel intensity

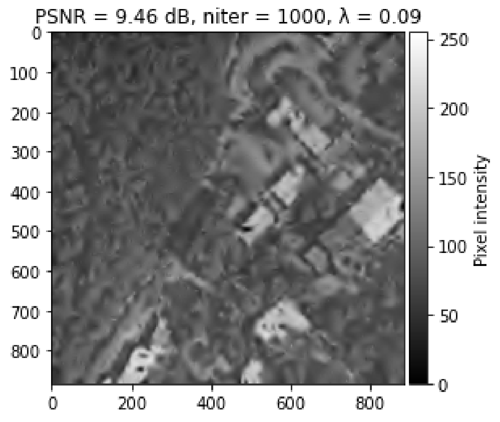

PSNR = 10.99 dB, niter = 300, λ = 0.09
Pixel intensity

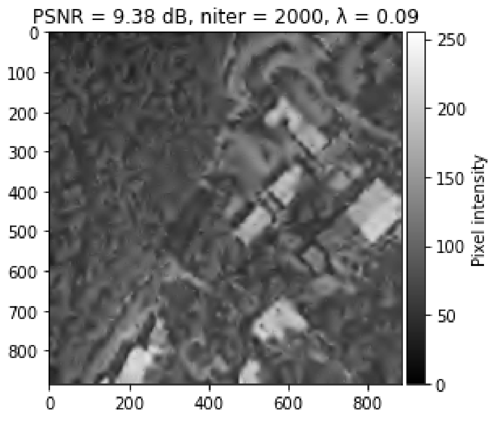

PSNR = 10.31 dB, niter = 400, λ = 0.09
Pixel intensity


PSNR = 9.95 dB, niter = 500, λ = 0.09
Pixel intensity


PSNR = 9.76 dB, niter = 600, λ = 0.09
Pixel intensity

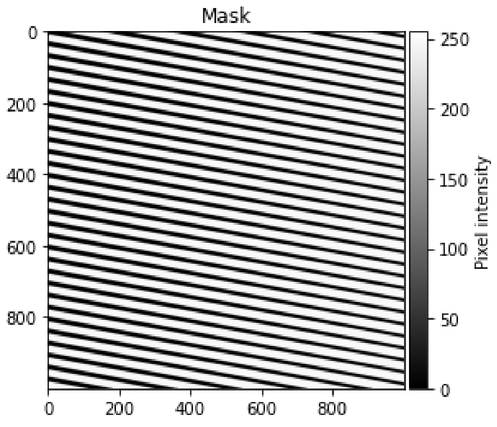

PSNR = 9.59 dB, niter = 750, λ = 0.09
Pixel intensity

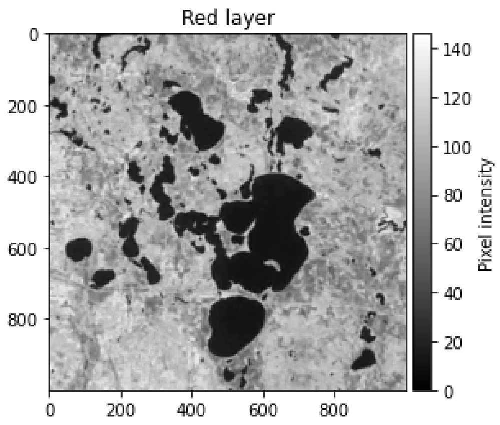

PSNR = 9.46 dB, niter = 1000, λ = 0.09
Pixel intensity

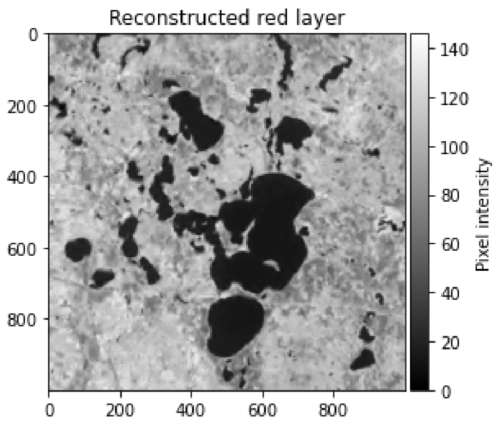

PSNR = 9.38 dB, niter = 2000, λ = 0.09
Pixel intensity

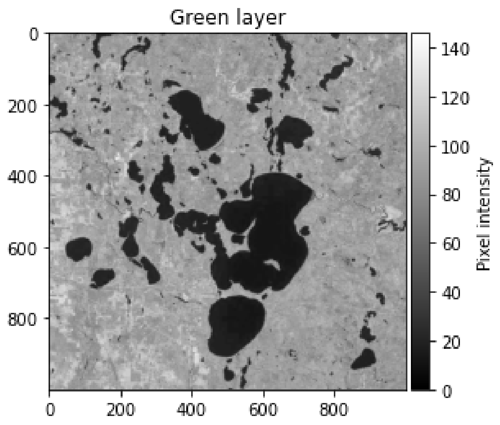
A.5 Landsat 7 images

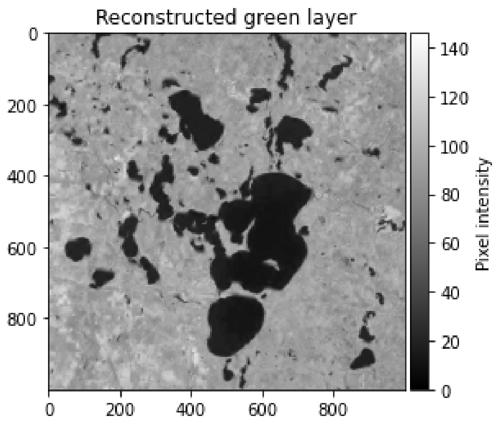
A.5.1 Undamaged scene

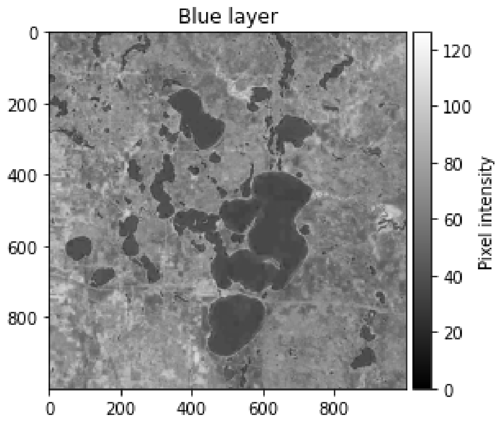


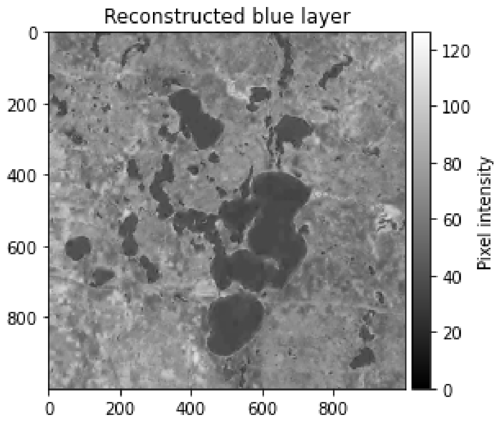


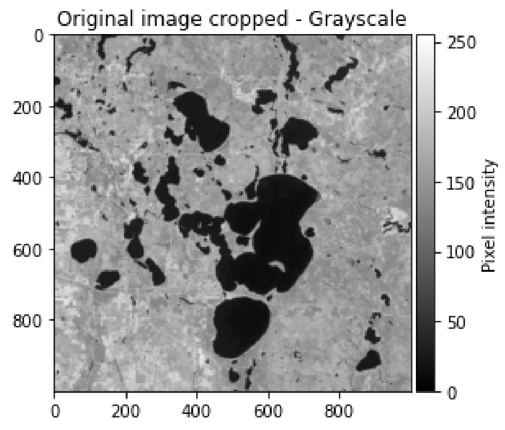


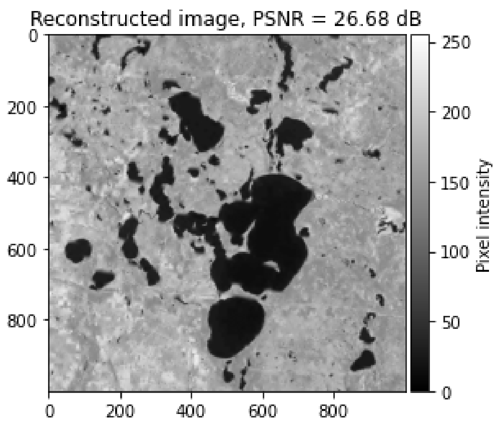


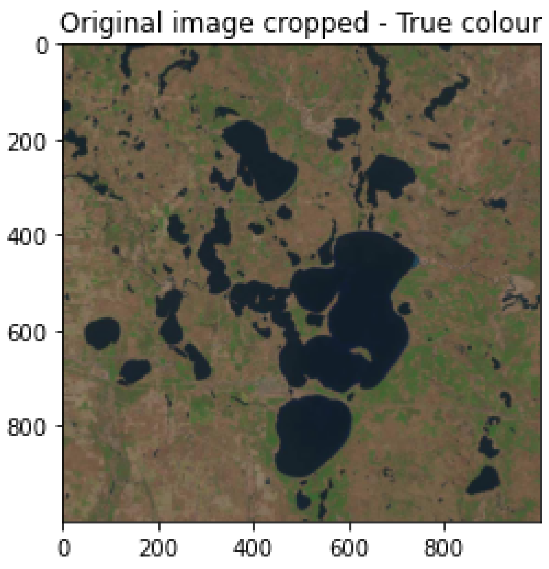


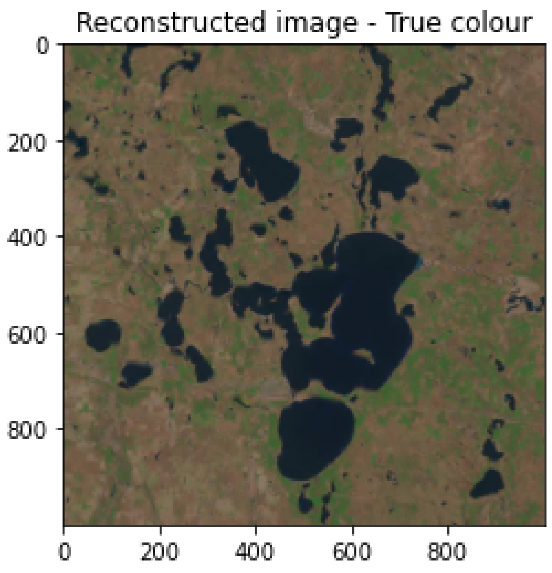




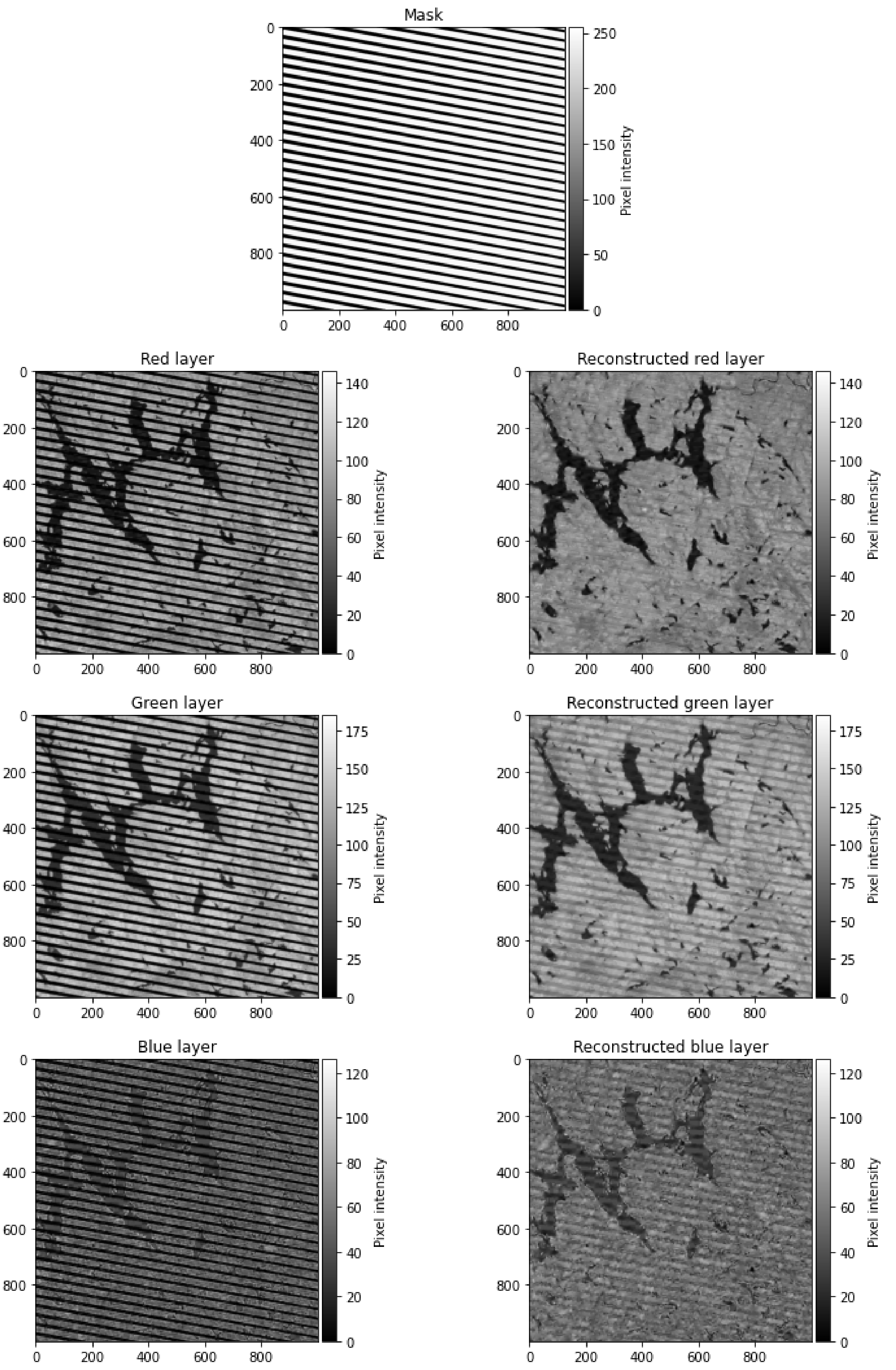

Original image cropped - Grayscale
Pixel intensity

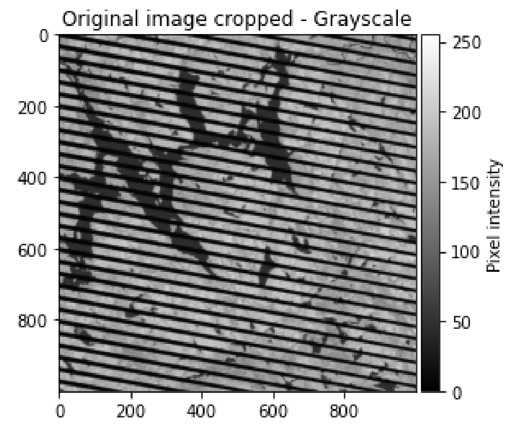

Reconstructed image, PSNR = 26.68 dB
Pixel intensity

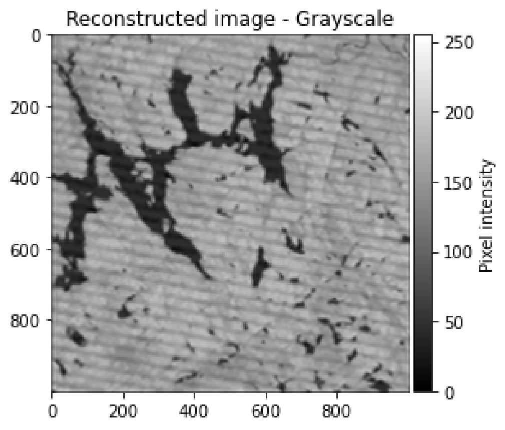

Original image cropped - True colour

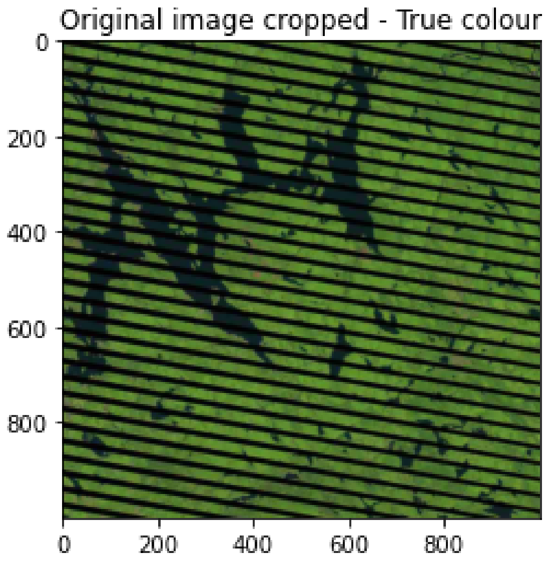

Reconstructed image - True colour

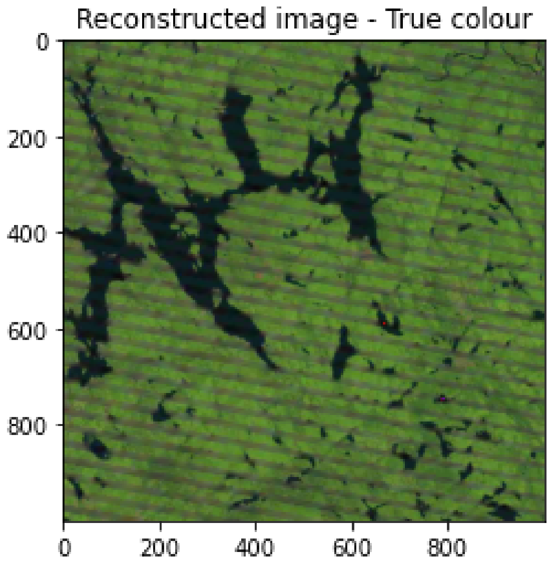
A.5.2 Damaged scene




Original image cropped - Grayscale
0
200
400
600
800
0
200
400
600
800
250
200
150
100
50
0
Pixel intensity


Reconstructed image - Grayscale
0
200
400
600
800
0
200
400
600
800
250
200
150
100
50
0
Pixel intensity


Original image cropped - True colour
0
200
400
600
800
0
200
400
600
800


Reconstructed image - True colour
0
200
400
600
800
0
200
400
600
800

Appendix B

Python codes

B.1 Iterative Hard Thresholding

Adapted from [4], this first numerical code, centred on the IHT image reconstruction algorithm, is complemented using the methods of ZP and grayscale image adjustment presented in chapter 3.

```
#################
# IMPORTING TOOLS
#################

from __future__ import division

# Scientific computing libraries :
import numpy as np
import scipy as scp

# Graphic visualisation :
import pylab as pyl
import matplotlib.pyplot as plt

from nt_toolbox.general import *
from nt_toolbox.signal import *

import warnings
warnings.filterwarnings('ignore')

from skimage.transform import rescale, resize, downscale_local_mean

%matplotlib inline
%load_ext autoreload
%autoreload 2

#######################
# IMPORT ORIGINAL IMAGE
#######################

f0 = load_image("nt_toolbox/data/original_scene_with_erosion_pattern_90%_14_05_2024.bmp")

# Image display
plt.figure(figsize = (5,5))
imageplot(f0, 'Original image 14_05')
```

```
#########################################################
# IMPORT DAMAGED IMAGE + determination of dimensions:
#########################################################

damaged_scene = load_image("nt_toolbox/data/damaged_scene_with_erosion_pattern_90%_14_05_2024.bmp")

# Image display
plt.figure(figsize = (5,5))
imageplot(damaged_scene, 'Observed damaged scene 14_05 with erosion')

# Image dimensions
dim = np.array([np.shape(damaged_scene)])
dim1 = dim [0,0]
dim2 = dim [0,1]

###################################################################
# CORRECTED DISPLAY FOR ERODED PATTERN (MASK OR DAMAGED SCENE)
###################################################################

from PIL import Image

damaged_scene_new = Image.open("nt_toolbox/data/pattern_mask_with_erosion_90%_14_05_2024.bmp")
damaged_scene_data = np.array(damaged_scene_new)
plt.matshow(damaged_scene_data)

height = damaged_scene.shape[0]
width = damaged_scene.shape[1]
imageArray = [0] * height * width

for x in range(width):
    for y in range(height):
        imageArray[x + width*y] = damaged_scene[y, x]

#######################
# IMPROVED ZERO PADDING
#######################
# Initial image dimensions (number of columns and lines)

print ("Image dimensions (number of pixels): ",)
col = dim2
lign = dim1

print("Length: ",col)
print("Width: ",lign)

print()

# Defining new variables

col_new = 0

lign_new = 0

# Conditions on the number of lines or columns added by the zero padding method

if col < lign :
    col_new = lign
    nbr_col_ajoutees = col_new - col

elif col > lign :
    col_new = col
    nbr_col_ajoutees = col - col_new

if lign < col :
    lign_new = col
    nbr_lign_ajoutees = lign_new - lign

elif lign > col :
    lign_new = lign
    nbr_lign_ajoutees = lign - lign_new

# If image already square at the start
```

```
if col == lign :
    nbr_col_ajoutees = 0
    nbr_lign_ajoutees = 0
    print("The original image is already square: no zero padding")
    print()

# Verification step: display of the number of added lines and columns

print ("Number of columns added (zero padding): ", nbr_col_ajoutees)

print ("Number of lines added (zero padding): ", nbr_lign_ajoutees)

# Definition of the image with zero padding

image_zero_padding = np.pad(damaged_scene, [(0, nbr_lign_ajoutees), (0, nbr_col_ajoutees)], mode='constant', constant_values=0)

# Verification step: display of the images (original and with zero padding)

fig, axes = plt.subplots(nrows=1, ncols=2)

ax = axes.ravel()

ax[0].imshow(damaged_scene, cmap='gray')
ax[0].set_title("Observed damaged scene 14_05")


ax[1].imshow(image_zero_padding, cmap='gray')
ax[1].set_title("Image with zero padding improved")

ax[0].set_xlim(0, col)
ax[0].set_ylim(lign, 0)

ax[1].set_xlim(0, col + nbr_col_ajoutees)
ax[1].set_ylim(lign + nbr_lign_ajoutees, 0)

plt.tight_layout()

###############################
# MASK Omega AND OBSERVATION y
###############################

# Determination of the square image dimension:

dim_new = 0

if col < lign :
    dim_new = lign

elif lign < col :
    dim_new = col

elif col == lign :
    dim_new = lign

### 1st possibility: generation of a random mask with same dimension (n) than the one of the initial square image

n = dim_new

# rho = 1 - mask filling ratio

rho = .3

from numpy import random

Omega_new = np.zeros([n, n])
sel = random.permutation(n**2)
np.ravel(Omega_new)[sel[np.arange(int(rho*n**2))]] = 1

### 2nd possibility:

Omega_new = load_image("nt_toolbox/data/pattern_mask_with_erosion_90%_14_05_2024.bmp")
```

```
Omega = 1 - Omega_new

# The damaging operator (mask) put to zeros the pixel locations x for which Omega(x) = 1 :

Phi = lambda f, Omega: f*(1-Omega)

### Observation y gathering the initial image with zero padding and the mask Omega

y = Phi(image_zero_padding, Omega)

########################################
# IHT: INITIALISATION OF THE PARAMETERS
########################################

SoftThresh = lambda x, T: x*np.maximum(1-T/np.maximum(abs(x), 1e-10*np.ones(np.shape(x))), np.zeros(np.shape(x)))

# Parameters of the wavelet transform

Jmax = np.log2(n)-1
Jmin = (Jmax-3)

# Shortcut for Ψ and Ψ* in the orthogonal case

from nt_toolbox.perform_wavelet_transf import *
Psi = lambda a: perform_wavelet_transf(a, Jmin, -1, ti=0)
PsiS = lambda f: perform_wavelet_transf(f, Jmin, +1, ti=0)

# Denoising operator

SoftThreshPsi = lambda f, T: Psi(SoftThresh(PsiS(f), T))

# ORTHOGONAL WAVELET SPARSITY

lambd = .03

ProjC = lambda f, Omega: Omega*f + (1-Omega)*y
fSpars = y

fSpars = ProjC(fSpars, Omega)

fSpars = SoftThreshPsi(fSpars, lambd)

#########################################
# TRANSLATION INVARIANT WAVELET SPARSITY
#########################################

J = Jmax-Jmin + 1
u = np.hstack(([4**(-J)], 4**(-np.floor(np.arange(J + 2./3,1,-1./3)))))
U = np.transpose(np.tile(u, (n,n,1)),(2,0,1))

lambd = .01

Xi = lambda a: perform_wavelet_transf(a, Jmin, -1, ti=1)
PsiS = lambda f: perform_wavelet_transf(f, Jmin, + 1, ti=1)
Psi = lambda a: Xi(a/U)

tau = 1.9*np.min(u)

a = U*PsiS(fSpars)

fTI = Psi(a)
a = a + tau*PsiS(Phi(y-Phi(fTI, Omega), Omega))

a = SoftThresh(a, lambd*tau)

fTI = Psi(a)

#######################################
#######################################

# ITERATIVE HARD THRESHOLDING
```

```
#######################################
#######################################

# Definition of the vector including the different niter values

nbr_iter = np.array([2,10,20,50,100,200,300,400,500,600,750,1000,2000])

# Vector length

nbr_iter_length = len(nbr_iter)

# Definition of the vector including the different PSNR values (filled with zeros which will be replaced in the loop),
# for image deterioration (PSNR) and image restoration (PSNR2)

PSNR = np.zeros(nbr_iter_length)
PSNR2 = np.zeros(nbr_iter_length)

# Array that will contain the images generated in the loop

image_array = []

###################################################################################################################
############################################### Loop to cover the different number of iterations values
###################################################################################################################

k = 0

for k in range (0,nbr_iter_length):

# Definition of a shortcut for this vectorialized hard thresholding

    HardThresh = lambda x, t: x*(abs(x) > t)

# Number of iterations

    niter = nbr_iter[k]

# List of thresholds
# One must start by a large enough initial threshold

    lambda_list = np.linspace(1, 0, niter)

# Initialization

    fHard = y

# Gradient descent

    fHard = ProjC(fHard, Omega)

# Hard threshold (here λ=λ0 is used)

    fHard = Xi(HardThresh(PsiS(fHard), tau*lambda_list[1]))

# Perform the iteration with a decaying value of λ
# lambda_list goes from 1 to 0 with niter-2 values in between (there is niter value taking into account 1 and 0)

    lambda_list = np.linspace(1, 0, niter)
    fHard = y

#################################### i goes from 0 to niter - 1 (with an increment of 1)

    for i in range(niter):
        fHard = Xi(HardThresh(PsiS(ProjC(fHard, Omega)), lambda_list[i]))

# lambda_list[i] in the line above must be replaced by a fixed number for a determined value of λ

# Display of the images with the corresponding PSNR and niter values

    plt.figure(figsize=(6,6))
    imageplot(fHard, "IHT, PSNR = %.1f dB" %psnr(image_zero_padding, fHard) + ", niter = " + str(nbr_iter[k]))

# Storage of the results in the corresponding vectors
```

```
    PSNR[k] = psnr(image_zero_padding, fHard)
    PSNR2[k] = psnr(f0, fHard)
    image_array.append (fHard)


##################################################################################################################
############################################### End of the loop
##################################################################################################################

####################################################################
# GRAPH OF PSNR AS A FUNCTION OF THE NUMBER OF ITERATIONS
# (comparison of reconstructions with damaged and undamaged scenes)
####################################################################

plt.scatter(nbr_iter, PSNR)
plt.scatter(nbr_iter, PSNR2)
plt.xlabel('Number of iterations (niter)')
plt.ylabel('PSNR [dB]')
plt.title('PSNR as a function of niter')
plt.show()

print("Nbr_iter: ", nbr_iter)
print("PSNR: ", PSNR)
print("PSNR2: ", PSNR2)

#################################
# CORRECTION OF IMAGES INTENSITY
#################################

# Using the np.clip function to restrict the value included in pixels between 0 and 1: allows to
# have an intensity similar to the one of the initial image

f_clipped = np.clip(image_array[0],0,1)

# np.around function to round all values from matrix to closer integer

f_clipped_corrected = np.around(f_clipped * 255)

##################################
# DISPLAY OF IMAGES WITH COLORBAR
##################################

import matplotlib.pyplot as plt
import numpy as np

from mpl_toolkits.axes_grid1 import make_axes_locatable

ax = plt.subplot()
ax.title.set_text('')
im = ax.imshow(f_clipped_corrected)

# Creation of an axis on the right side of ax
# The width of cax will be 5% of ax and the padding between cax and ax will be fixed at 0.05 inch

divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)

plt.gray()
plt.colorbar(im, cax=cax, label = 'Pixel intensity')

plt.show()
```

B.2 Image deconvolution

This second code performs image deconvolution using a GK. Deconvolution of images involving a measured PSF is achieved by replacing the initial GK with the PSF image having the same dimensions. Adapted from [5] and entirely rewritten in Python.

```
#################
# IMPORTING TOOLS
#################

from __future__ import division

# Bibliothèques de calcul scientifique :
import numpy as np
import scipy as scp

# Graphic visualisation :
import pylab as pyl
import matplotlib.pyplot as plt

from nt_toolbox.general import *
from nt_toolbox.signal import *

import warnings
warnings.filterwarnings('ignore')

from skimage.transform import rescale, resize, downscale_local_mean


%matplotlib inline
%load_ext autoreload
%autoreload 2

#####################################################
# IMPORT ORIGINAL IMAGE / GROUND TRUTH  (1024 x 1280)
#####################################################

f0 = load_image("nt_toolbox/data/original_scene_image_complete_14_05_2024.bmp")
plt.figure(figsize = (5,5))
imageplot(f0, 'Original image')

#######################################################
#######################################################
#######################################################

# PROCESSING WITH GAUSSIAN KERNEL

#######################################################
#######################################################
#######################################################

###################################
# GENERATION OF THE GAUSSIAN KERNEL
###################################

import numpy as np

n = 884
x = np.concatenate((np.arange(0, n/2), np.arange(-n/2, 0)), axis=0)
X, Y = np.meshgrid(x, x)
s = 1.1
h = np.exp(-(X**2 + Y**2) / (2 * s**2))
h = h / np.sum(h)

# Kernel 20 x 20 (cropped)

gk_cropped = np.fft.fftshift(h)[884//2 - 10:884//2 + 10 , 884//2 - 10:884//2 + 10]

# Zero padding : square image 1280 x 1280

def pad_with(vector, pad_width, iaxis, kwargs):
    pad_value = kwargs.get('padder', 0)
    vector[:pad_width[0]] = pad_value
    vector[-pad_width[1]:] = pad_value
np.pad(h, 396 , pad_with)

gk_zero_padding = np.pad(gk_cropped, 630 , pad_with)
```

```
# Cropped image kernel : 1024 x 1280 (to match the original image)

gk_zero_padding_cropped = gk_zero_padding[128:1152, 0:1280]

##################################################################
# FOURIER TRANSFORM OF THE GAUSSIAN KERNEL AND DENOISING OPERATOR
##################################################################

# FT

FT_gk = np.fft.fft2(np.fft.fftshift(gk_zero_padding_cropped)).real

# Denoising operator

Phi = lambda f: np.fft.fftshift(np.fft.ifft2(np.fft.fft2(f)*FT_gk)).real

# Observation

y0 = Phi(f0)

# Display of the observation

plt.figure(figsize = (5,5))
imageplot(np.fft.fftshift(y0), 'Observation without noise gaussian kernel')

##############################################################################
# DISPLAY OF THE IMAGES: COMPARISON BETWEEN ORIGINAL IMAGE AND DENOISED SCENE
##############################################################################

fig, axes = plt.subplots(nrows=1, ncols=2)

ax = axes.ravel()


ax[0].imshow(f0, cmap='gray')
ax[0].set_title("Original image")
ax[1].imshow(np.fft.fftshift(y0), cmap='gray')
ax[1].set_title("Observation without noise")

############################################################
# CROPPING IMAGES TO DESIRED DIMENSIONS FOR RECONSTRUCTIONS
############################################################

### Reconstruction for square image 884 x 884

# Cropped image kernel : 884 x 884

gk_zero_padding_new = np.pad(gk_cropped, 432 , pad_with)

# Cropped f0 and y0 884 x 884

f0_cropped = f0 [28:912 , 157:1041]

y0_cropped = np.fft.fftshift(y0) [28:912 , 157:1041]

plt.figure(figsize=(6,6))
imageplot(y0_cropped, "Observation without noise gaussian kernel, PSNR = %.1f dB" %psnr(f0, np.fft.fftshift(y0)))

# FT 884 x 884

FT_gk_new = np.fft.fft2(np.fft.fftshift(gk_zero_padding_new)).real

# Denoising operator

Phi = lambda f: np.fft.fftshift(np.fft.ifft2(np.fft.fft2(f)*FT_gk_new)).real

# Observation

y1 = Phi(f0_cropped)

# Display of the observation
```

```python
plt.figure(figsize=(6,6))
imageplot(np.fft.fftshift(y1), 'Observation without noise gaussian kernel cropped')

####################################################
# INITIALISATION OF PARAMETERS FOR RECONSTRUCTION
####################################################

SoftThresh = lambda x, T: x*np.maximum(1-T/np.maximum(abs(x), 1e-10*np.ones(np.shape(x))), np.zeros(np.shape(x)))

# Parameters of the wavelet transform

Jmax = np.log2(n)-1
Jmin = (Jmax-3)

# Shortcut for Ψ and Ψ* in the orthogonal case

from nt_toolbox.perform_wavelet_transf import *
Psi = lambda a: perform_wavelet_transf(a, Jmin, -1, ti=0)
PsiS = lambda f: perform_wavelet_transf(f, Jmin, +1, ti=0)

# Denoising operator

SoftThreshPsi = lambda f, T: Psi(SoftThresh(PsiS(f), T))

# ORTHOGONAL WAVELET SPARSITY

lambd = .02

tau = 1.5

niter = 100

fSpars = y1

fSpars = fSpars + tau * Phi( y1-Phi(fSpars) )

fSpars = SoftThreshPsi( fSpars, lambd*tau )

#########################################
# TRANSLATION INVARIANT WAVELET SPARSITY
#########################################

J = Jmax-Jmin + 1
u = np.hstack(([4**(-J)], 4**(-np.floor(np.arange(J + 2./3,1,-1./3)))))
U = np.transpose(np.tile(u, (n,n,1)),(2,0,1))

lambd = .01

Xi = lambda a: perform_wavelet_transf(a, Jmin, -1, ti=1)
PsiS = lambda f: perform_wavelet_transf(f, Jmin, + 1, ti=1)
Psi = lambda a: Xi(a/U)

tau = 1.9*np.min(u)

a = U*PsiS(fSpars)

fTI = Psi(a)
a = a + tau*PsiS(Phi(y1-Phi(fTI)))

a = SoftThresh(a, lambd*tau)

fTI = Psi(a)

############################
# SPARSITY DECONVOLUTION TIW
############################

lambda_max = .005
lambda_max = .01
lambda_list_TI = np.linspace(lambda_max/4,lambda_max,niter)
a = PsiS(y1)*0
errTI = []
```

```
for i in range(niter):
    fTI = Psi(a);
    d = y1-Phi(fTI);
    a = SoftThresh( a + tau*PsiS(Phi(d)), lambda_list_TI[1]*tau )

# Descent

for i in range(niter):
    fTI = Psi(a)
    d = y1-Phi(fTI)
    # step
    a = SoftThresh( a + tau*PsiS(Phi(d)), lambda_list_TI[i]*tau )
    errTI.append(psnr(f0_cropped, fTI))
    if i > 1 and errTI[-1] > max(errTI[:-1]):
        fBestTI = fTI


plt.figure(figsize=(6,6))
imageplot(clamp(fBestTI), "Sparsity deconvolution, PSNR = %.1f dB" %psnr(f0_cropped, fBestTI))

print(psnr(f0_cropped, fBestTI))
```

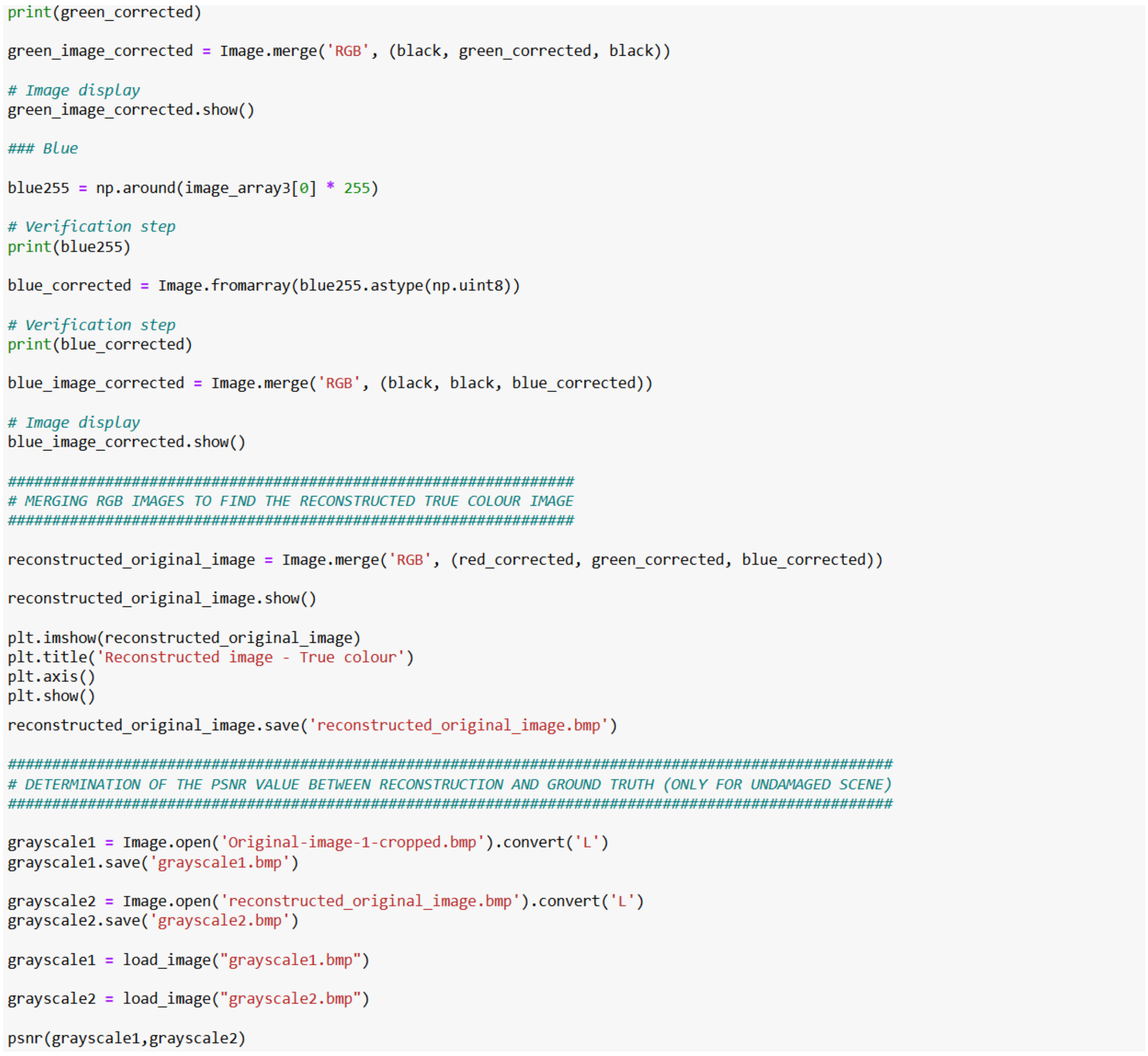
B.3 Landsat 7 images reconstructions

Dedicated to the reconstruction of reflective colour bands of USGS data, this third and last code presents the way RGB images can be numerically processed. Here, the main RGB images manipulations are inspired by information found in [25].

```
#################
# IMPORTING TOOLS
#################

from __future__ import division

# Scientific computing libraries :
import numpy as np
import scipy as scp

# Graphic visualisation :
import pylab as pyl
import matplotlib.pyplot as plt

from nt_toolbox.general import *
from nt_toolbox.signal import *

import warnings
warnings.filterwarnings('ignore')

from skimage.transform import rescale, resize, downscale_local_mean

%matplotlib inline
%load_ext autoreload
%autoreload 2

###############
# IMPORT IMAGES
###############

### Mask

Omega_new = load_image("nt_toolbox/data/Mask - 1.bmp")

# Image display
plt.figure(figsize = (10,10))
imageplot(Omega_new, 'Omega_new')

### Original image

original1 = load_image("nt_toolbox/data/Original-image-1.bmp")

# Image display
plt.figure(figsize = (10,10))
imageplot(original1, 'Original image')

### Original image cropped

original1_cropped = load_image("nt_toolbox/data/Original-image-1-cropped.bmp")

# Image display
plt.figure(figsize = (10,10))
imageplot(original1_cropped, 'Original image cropped')

#############################
# IMAGES DISPLAY: TRUE COLOUR
#############################

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

from mpl_toolkits.axes_grid1 import make_axes_locatable


image = mpimg.imread("nt_toolbox/data/Original-image-1.bmp")
image2 = mpimg.imread("nt_toolbox/data/Original-image-1-cropped.bmp")
image3 = mpimg.imread("nt_toolbox/data/Mask - 1.bmp")


plt.imshow(image)
plt.title('Original image - True colour')
plt.axis()
plt.show()

plt.imshow(image2)
plt.title('Original image cropped - True colour')
plt.axis()
plt.show()


plt.imshow(image3)
plt.title('Mask')
plt.axis()
plt.show()

####################################
# IMAGE SPLITTING: RGB DECOMPOSITION
####################################

from PIL import Image

img = Image.open('nt_toolbox/data/Original-image-1-cropped.bmp')

### Decomposition in red, green and blue layers + visualisation

red, green, blue = img.split()
red.show()
green.show()
blue.show()

### Formation of red, green and blue images + visualisation

black = Image.new('L', img.size)

red_image = Image.merge('RGB', (red, black, black))
green_image = Image.merge('RGB', (black, green, black))
blue_image = Image.merge('RGB', (black, black, blue))

red_image = Image.merge('RGB', (red, black, black))
green_image = Image.merge('RGB', (black, green, black))
blue_image = Image.merge('RGB', (black, black, blue))

red_image.show()
green_image.show()
blue_image.show()

############################################
# CROPPING THE MASK TO FIT THE CROPPED IMAGE
############################################

Omega_new_cropped = Omega_new [618:1618 , 1484:2484]

plt.figure(figsize = (20,20))
imageplot(Omega_new_cropped, 'Omega_new_cropped')

# Visualisation of the observation (verification)

plt.figure(figsize = (20,20))
imageplot(red*Omega_new_cropped, 'red*Omega_new_cropped')
```

Once an appropriate portion of the mask is selected, the *red*, *green* and *blue* variables are introduced separately in the IHT code presented in section B.1. Three distinct reconstructions are therefore carried out, and each of the colour layers is thus reconstructed individually. The IHT code is consequently replicated three times but it is not displayed here to avoid redundancy. Some operations are however needed to change the nature of the objects (RGB layers), so that they can be introduced into the reconstruction process. These changes take place after the ZP step and before the IHT reconstructions and are exposed hereafter.

```
#################################################################
# AFTER ZP AND BEFORE IHT RECONSTRUCTIONS: DATA TYPE CONVERSION
#################################################################

### Red

image_zero_padding1 = np.array(red).astype(np.float64)

image_zero_padding_red = image_zero_padding1/255

print(image_zero_padding_red)

### Green

image_zero_padding2 = np.array(green).astype(np.float64)

image_zero_padding_green = image_zero_padding2/255

print(image_zero_padding_green)

### Blue

image_zero_padding3 = np.array(blue).astype(np.float64)

image_zero_padding_blue = image_zero_padding3/255

print(image_zero_padding_blue)

#########################################
# DISPLAY OF THE RECONSTRUCTIONS RESULTS
#########################################

plt.figure(figsize = (20,20))
imageplot(image_array[0], 'image_array[0]')

plt.figure(figsize = (20,20))
imageplot(image_array2[0], 'image_array2[0]')

plt.figure(figsize = (20,20))
imageplot(image_array3[0], 'image_array3[0]')

###############################################################################
# DATA TYPE CONVERSION BEFORE IMAGES MERGING (VERIFICATION OF IMAGES CONTENT)
###############################################################################

### Red

red255 = np.around(image_array[0] * 255)

# Verification step
print(red255)

red_corrected = Image.fromarray(red255.astype(np.uint8))

# Verification step
print(red_corrected)

red_image_corrected = Image.merge('RGB', (red_corrected, black, black))

# Image display
red_image_corrected.show()

### Green

green255 = np.around(image_array2[0] * 255)

# Verification step
print(green255)

green_corrected = Image.fromarray(green255.astype(np.uint8))

# Verification step
```



```python
print(green_corrected)

green_image_corrected = Image.merge('RGB', (black, green_corrected, black))

# Image display
green_image_corrected.show()

### Blue

blue255 = np.around(image_array3[0] * 255)

# Verification step
print(blue255)

blue_corrected = Image.fromarray(blue255.astype(np.uint8))

# Verification step
print(blue_corrected)

blue_image_corrected = Image.merge('RGB', (black, black, blue_corrected))

# Image display
blue_image_corrected.show()

################################################################
# MERGING RGB IMAGES TO FIND THE RECONSTRUCTED TRUE COLOUR IMAGE
################################################################

reconstructed_original_image = Image.merge('RGB', (red_corrected, green_corrected, blue_corrected))

reconstructed_original_image.show()

plt.imshow(reconstructed_original_image)
plt.title('Reconstructed image - True colour')
plt.axis()
plt.show()

reconstructed_original_image.save('reconstructed_original_image.bmp')

####################################################################################################
# DETERMINATION OF THE PSNR VALUE BETWEEN RECONSTRUCTION AND GROUND TRUTH (ONLY FOR UNDAMAGED SCENE)
####################################################################################################

grayscale1 = Image.open('Original-image-1-cropped.bmp').convert('L')
grayscale1.save('grayscale1.bmp')

grayscale2 = Image.open('reconstructed_original_image.bmp').convert('L')
grayscale2.save('grayscale2.bmp')

grayscale1 = load_image("grayscale1.bmp")

grayscale2 = load_image("grayscale2.bmp")

psnr(grayscale1,grayscale2)
```