Processing Image Sequences
Without Motion Estimation

Matan Protter
Processing Image Sequences
Without Motion Estimation

Research thesis

In Partial Fulfillment of the
Requirements for the
Degree of Doctor of Philosophy

Matan Protter

Submitted to the Senate of
the Technion - Israel Institute of Technology

Shebat, 5771 Haifa January 2011
The Research Thesis Was Done Under The Supervision of Prof. Michael Elad in the Department of Computer Science

First and foremost I would like to express my sincerest gratitude to my advisor, Prof. Miki Elad. He knew when to push me hard and when to push me harder. He was all that I could ask for in an advisor.

I would like to thank all faculty members I have had the privilege of working with or taking courses from. I have learned a lot from each of them.

I would also like to thank my parents, Hana and Moshe, for teaching me to work hard, encouraging me to be the best I can be, and always being willing to help. Finally, I would like to thank my life-partner Ortal, for always being so incredibly supportive and understanding.

The Generous Financial Help Of the Jacobs-Qualcomm Foundation Is Gratefully Acknowledged
# Contents

Abstract 1

Abbreviations and Notations 3

1 Introduction 5

1.1 Acquisition of Degraded Signals .............................. 5
1.2 Images as Signals ........................................... 6
1.3 Models for Images ............................................. 7
1.4 From Images to Image Sequences .............................. 9
1.5 Overview of the Thesis ...................................... 10

2 Image Sequence Denoising via Sparse and Redundant Representations 13

2.1 Introduction .................................................. 14
2.2 Image Denoising Using Sparsity and Redundancy .............. 16
2.3 Extension to Video Denoising ................................ 17
  2.3.1 Constructing The Algorithm .............................. 18
  2.3.2 Overall Algorithm and Parameter Selection ............ 23
  2.3.3 Complexity of the Algorithm ............................ 26
2.4 Comparison to State-of-the-Art ................................ 28
  2.4.1 Overview of other methods ............................... 28
  2.4.2 Comparison Results ...................................... 29
2.5 Conclusions .................................................. 33

3 Generalizing the Non-Local-Means to Super-Resolution Reconstruction 35

3.1 Introduction .................................................. 36
3.2 The Bilateral and the NLM Denoising Filters .................. 39
3.3 NLM Via Energy Minimization ................................ 40
List of Tables

2.1 Denoising Comparison With [140] .................................. 30
2.2 Denoising Comparison With [13] .................................. 31
2.3 PSNR of Noist Sequences ........................................ 31
2.4 Comparison with NLM, VBM3D and Single Image Denoising ....... 32

3.1 Core SR Algorithm .................................................. 55
3.2 Entire SR Algorithm ............................................... 56
3.3 Mean-PSNR Results for the Test Sequences ...................... 65

5.1 The general signal generation model, and the implied MAP and MMSE
estimators .......................................................... 102
## List of Figures

2.1 PSNR gain by using 3D atoms ............................................. 20  
2.2 Examples of 3D Atoms ...................................................... 20  
2.3 PSNR gain by dictionary propagation ................................. 21  
2.4 Visual dictionary comparison of dictionary propagation .......... 22  
2.5 PSNR gain by temporally extended training set ..................... 24  
2.6 Example denoising result on Football sequence .................... 24  
2.7 Example denoising result on Tennis sequence ....................... 25  
2.8 Example denoising result on Garden sequence ....................... 25  
2.9 Visual comparison of denoising with NLM and BM3D ............... 33  
3.1 The imaging process to be reversed by super resolution ............ 36  
3.2 Simple super-resolution with known global translations ........... 37  
3.3 NLM as fuzzy motion estimation ........................................ 48  
3.4 Relationship between patch sizes .................................... 50  
3.5 Description of patch decimation operator ............................ 50  
3.6 SR results for synthetic text sequence ............................... 59  
3.7 SR results for synthetic text sequence with one image missing .... 60  
3.8 SR results for ”Miss America” sequence ............................... 62  
3.9 SR results for ”Foreman” sequence .................................... 63  
3.10 SR results for ”Suzie” sequence ...................................... 64  
4.1 SR results for ”Suzie” sequence ....................................... 80  
4.2 SR results compared to NLM followed by deblurring ............... 81  
4.3 De-Interlacing results with proposed framework .................... 82  
5.1 Denoising results on synthetic test: simple signal generation model .... 104
5.2 Denoising results on synthetic test: constant small cardinality, heteroscedastic coefficient set and different atom probabilities .......... 105
5.3 Denoising results on synthetic test: constant large cardinality, heteroscedastic coefficient set and different atom probabilities .......... 105
5.4 Denoising results on synthetic test: general signal generation model ........ 106
5.5 Effective representation obtained for one signal .................. 106
5.6 Denoising results on real-world images .......................... 109
5.7 PSNR gap between MMSE and MAP denoising .................... 109
5.8 Visual comparison of MAP and MMSE denoising .................. 110
5.9 Estimated parameters of the model for ”Boat” image ............... 111
Abstract

Digital image restoration is a prominent field in signal processing, focusing on improving the quality of images suffering from various degradation effects such as noise and blur. Performing the restoration usually requires modelling the image content in order to separate the true image content from the degradation effects and restoring the degradation-free content.

Restoration of image sequences can obtain better results compared to restoring each image individually, provided the temporal redundancy is adequately used. Most image sequence processing algorithms rely on estimating the motion between the frames in order to be able to merge the data from various frames.

However, in most sequences, the motion patterns are very complex, and as a result motion estimation, a severely under-determined problem, tends to be error-prone and inaccurate. Thus, algorithms relying on motion estimation tend to reduce to single image processing in areas of the image containing these complex motion patterns. Unfortunately, as most sequences indeed exhibit mostly complex motion patterns, relying on motion estimation is not able to fully exploit the benefits of having multiple frames of the same scene.

In the last several years there has been a trend of circumventing motion estimation in the denoising of image sequences. Such a feat has been made possible by the emergence of powerful image models, extended to model image sequences as well. We propose a contribution along these lines, extending the model of sparse and redundant representations to the denoising of image sequences. We show that state-of-the-art results are obtained with this method, indeed proving that motion estimation can be avoided for denoising of image sequences.

Another restoration field relying on motion estimation is super-resolution, in which several images of the same scene are merged into a high-quality image (or sequence) of the scene. Each image offers a different sampling of the scene (assuming the sequence indeed contains motion), and motion estimation is used to merge the images. Even more than in denoising, this procedure requires very high accuracy, which is not possible in the majority of sequences.
Relying on the intuition gained from denoising sequences while avoiding explicit motion estimation, we offer two different approaches to achieve the same feat in super-resolution. This is done by relying on crude, probabilistic motion estimation to replace the explicit one. We show that this alternative path is indeed able to successfully handle sequences previously considered outside the realm of super-resolution due to their complicated motion patterns.

We conclude by revisiting the denoising problem, focusing on signals obeying the sparse and redundant modelling of signals. It has been shown that averaging several sparse representations achieves better denoising than the sparsest representation alone. This has been explained by relating these two solutions to approximations of the MMSE and MAP estimators, respectively. In general, both MAP and MMSE cannot be computed directly. We show that in the special case where the dictionary is unitary, both estimators enjoy a closed-form formula, with the MMSE out-performing the MAP in this case as well.
## Abbreviations and Notations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LHS, RHS</td>
<td>Left or Right Hand Side (of an equation or formula)</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum-A-Posteriori</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum-Likelihood</td>
</tr>
<tr>
<td>MMSE</td>
<td>Minimum-Mean-Squared-Error</td>
</tr>
<tr>
<td>NLM</td>
<td>Non Local Means; algorithm</td>
</tr>
<tr>
<td>OMP</td>
<td>Orthogonal Matching Pursuit; algorithm</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>$\alpha, \beta$</td>
<td>A vector of coefficients (usually assumed sparse)</td>
</tr>
<tr>
<td>$\epsilon, \mu$</td>
<td>A cost function on an image</td>
</tr>
<tr>
<td>d</td>
<td>An atom in a dictionary</td>
</tr>
<tr>
<td>$i, j, k, l$</td>
<td>Indices of pixels</td>
</tr>
<tr>
<td>s</td>
<td>Scale factor; assumed to be integer, unless specified otherwise</td>
</tr>
<tr>
<td>x</td>
<td>High-quality (assumed unknown) signal</td>
</tr>
<tr>
<td>$\hat{x}$</td>
<td>Recovered signal (algorithm’s result)</td>
</tr>
<tr>
<td>y</td>
<td>Degraded, usually input, signal</td>
</tr>
<tr>
<td>$w[i, j, k, l]$</td>
<td>A weight (non-negative) computed between pixels $(i, j)$ and $(k, l)$</td>
</tr>
<tr>
<td>D</td>
<td>Dictionary; depending on context</td>
</tr>
<tr>
<td>$D_p$</td>
<td>Decimation operator operating on a patch</td>
</tr>
<tr>
<td>$D_{ij}$</td>
<td>Decimation operator operating on a patch</td>
</tr>
<tr>
<td>F</td>
<td>A warping operator (applied to an image)</td>
</tr>
<tr>
<td>$N(k, l)$</td>
<td>Neighbourhood of the pixel $(k,l)$; actual location depends on context</td>
</tr>
<tr>
<td>$R_{ij}$</td>
<td>Patch extraction operator (around the pixel $(i, j)$)</td>
</tr>
<tr>
<td>$R_H^{ij}, R_L^{ij}$</td>
<td>Patch extraction operators from high and low resolution images respectively</td>
</tr>
<tr>
<td>S</td>
<td>Support; defines the locations of non-zeros in a sparse coefficient vector</td>
</tr>
<tr>
<td>TV</td>
<td>Total Variation; a function applied to a signal, measuring it’s smoothness</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Acquisition of Degraded Signals

The way we interact with the world around us is by acquisition (and transmission) of signals. Speech (voice signal), sight (image and video signals), weather information (temperature as a function of time) and financial data (stock prices) are just a few examples of signals we deal with in everyday life. A system dealing with these signals must first acquire them, converting them into a series of samples. The acquired signals can then be used for a variety of purposes - either for entertainment (by playing back the signal), analysis (figuring out what happened), forecast and more.

Unfortunately, rarely is the acquisition process faultless and completely accurate. In most cases, the acquired signal is a degraded version of the original perfect one. Noise is possibly the most common degradation effect, causing the measurements to deviate from the true values. Missing samples are also very often encountered, and may be the result of the inability to sample with sufficient frequency or due to obstructions (e.g., a tree hiding part of the view in a photograph). One of the most important fields of signal processing, which this thesis belongs to, is therefore devoted to the theory and applications of countering the degradation effects.

In order to be able to restore as much as possible from the original high-quality signal, restoration methods rely on understanding and then exploiting the inherent structure of the signal, called model. Of course, different sources of signals require different models. A model that is appropriate to stock prices is not likely to competently describe images. Consequently, different types of signals are being handled with different models and tools. This thesis focuses on one prominent family of signals - images and image sequences, and
1.2 Images as Signals

One of the most important points in photography is undoubtedly image quality. Unfortunately, there are many factors jeopardizing image quality, resulting in a degraded image. The field this thesis belongs to is that of ”Inverse Problems”, where the goal is to invert the degradation encountered in the acquisition process, thus returning to an image with the highest quality possible.

The emergence of digital photography, coupled with the decreasing prices of photographic equipment, has led to a wide-spread use of digital cameras. Photography is no longer a field attempted only by experts, using high-end equipment and in near sterile conditions to create perfect photographs. Instead, images are taken in various disadvantaged scenarios using mediocre equipment, while the photographer still desires high quality results. This is where the field of image processing can help.

Let us denote the high-quality image by $x$. Noise is almost always an inherent part of the acquisition process, and is therefore assumed to be added (with some magnitude) to the image taken, so $y = x + n$, with $n$ being the noise image. The task of removing noise is termed denoising, with numerous papers attempting this feat. A representative review of this field can be found in [15, 98, 108]. Deblurring [11, 55] is also a very common task, attempting to sharpen the acquired image. This sharpening faces blurriness in the image, which is usually the result of natural effects of the atmosphere or of the use of lenses in the camera. The blur is commonly denoted by a linear, space-invariant (LSI) operator $H$ (although in some cases it is assumed to be location dependent), so the connection between the desired image and acquired image is $y = Hx + n$ (with noise also being added).

The in-painting task deals with filling in missing parts of the image [8, 9, 28, 40, 48]. Such missing parts can be caused by a variety of reasons, such as faulty pixels, dirt on the lens or simply an obstructing element in the image we wish to remove. Such missing areas are normally termed masked, and are denoted by a masking operator $M$ applied to the image, so $y = Mx + n$. Scaling-up a single image is also a popular task [31, 45, 110, 61]. Such degradation effects - noise, blur, and more - are only some of the problems usually encountered. For handling these (and similar) tasks, a solid model describing image content is needed [63, 72].
1.3 Models for Images

Let us start with the simplest task, the removal of noise from an image. Its (relative) simplicity allows easier deployment of concepts to the task at hand. The given image \( y \) was created by the addition of noise \( n \) (usually assumed to be white, Gaussian and independent between pixels) to the original image \( x \). The task of denoising requires finding the denoised image \( \hat{x} \) which is as close to the original (unknown) image as possible, minimizing the distance \( \|x - \hat{x}\|_2 \) as possible.

Suppose we know what the image is supposed to behave like. This knowledge is in the form of a model for image content. Using the model, the denoising task becomes simple in principle. Given the image \( y \), we seek the image \( \hat{x} \) that on one hand behaves according to the model, and on the other hand is close enough to \( y \) for the difference to be explained by the noise.

Put formally, a model of an image is usually written as a penalty function \( Pr(x) \), usually termed prior. The prior is a non-negative function that assigns a cost to each and every image. The lower the cost, the more the image behaves according to the model. The ideal image - in the eyes of the prior - is an image that is assigned a cost of 0. Therefore, the denoising task is carried out by solving the minimization problem

\[
\hat{x} = \arg\min_x \|x - y\|_2 + \lambda \cdot Pr(x). \tag{1.1}
\]

The first term requires proximity between the recovered image and the input image, the second term is the prior, and the constant \( \lambda \) balances the contributions of both terms. Similar formulations can be used for other inverse problems, such as deblurring.

A simple example of such a model assumes spatial smoothness, by requiring that the sum of the gradient’s norm over the entire image is small:

\[
Pr(x) = \int_{(v,h) \in \Omega} \left[ (\frac{\partial x}{\partial h})^2 + (\frac{\partial x}{\partial v})^2 \right] dh \, dv, \tag{1.2}
\]

with \( h \) and \( v \) representing the horizontal and vertical direction respectively. A better model would be one that replaces the \( \ell_2 \) norm with an \( \ell_1 \) norm, thus allowing some sharp edges in the image. This option is known in general as Bounded Variation (BV) norm [108]. When applied to image processing tasks, it is commonly referred to as the Total Variation (TV) technique [108, 115, 116].

The quality of models for images has greatly improved with the years by improving descriptiveness and thereby improving the quality of denoising and other restoration prob-
lems. Unfortunately, the improved descriptiveness is usually accompanied by the models becoming more complicated and thus harder to deploy.

Principal-Component-Analysis (PCA, also known as the Karhunen-Loeve Transform - KLT) [128, 65, 66, 67] is a popular model, with the key assumption being that there exists a transform (such as DCT) that condenses the image energy to few leading coefficients, basically assuming that image content is low-dimensional in nature. Sparsity over a specific domain of representation (e.g. Wavelets [118] and its extensions Contourlets [35] and Curvelets [20]) is very popular, extending the PCA from linear to non-linear approximation [14]. Alternative models include Markov Random Field (MRF) [58, 59, 107, 136] and other stochastic description of image content, entropy and cross-entropy behaviours, local geometric behaviour, and more. These models all share the benefit of proposing a simple mathematical formulation for the model. However, it is hard to expect images to actually follow simplistic mathematical formulas.

A promising alternative type of models is the example-based approach. The basic idea is that only images can be used to represent an actual image, thus accounting for the great wealth and variation in image content. Such models either learn the parameters of a model from a given set of images, or use images directly within the model.

There are three main types of models, differing by the source of the images used in the model and thus what is attempted to be modelled. One option is building a general model of images, and thus the model is trained on a set of high-quality (degradation-free) images [2, 3, 50, 18, 89, 107, 136]. The benefit of such a model is that it is general, and can be applied to any image and any restoration task. However, here also lies its weakness, as it is not able to exploit specifics related to the task at hand. Furthermore, it may be preferable to train specific models to specific types of images (e.g., nature scenes, facial images) rather than attempt representing all types of images using the same model.

A second option is training the model in such a way that it takes into account the expected degradation process [31, 45, 56, 57, 68, 77, 130]. This can be done by training the model on high-quality images and their degraded counterparts (using the known degradation process). While being appropriate only to the specific degradation, this approach allows tuning the model specifically to the quality lost in this degradation process.

The third option is training the model on the actual degraded data available (instead of a pre-determined set of images) [28, 40, 13, 15, 16, 43, 44, 100, 29, 137]. While not benefiting from having high-quality examples, it is hoped that the training procedure is
able to infer the low-dimensional model of the image while avoiding the degradation effects. The expected benefit of this approach is that it yields a model tailored specifically to the image content to be processed, thus avoiding problems related to model mismatches.

When turning to handle image sequences, it is possible to treat its restoration as a set of independent restorations tasks, essentially restoring each image in the sequence separately. However, such an approach is not able to benefit from the fact that consecutive images are usually very similar to each other. Ignoring this similarity does not take all available information into account, and thus not achieving as good results as possible. A better approach must take into account (in some way or the other) the connection between the images.

1.4 From Images to Image Sequences

The area of image sequence processing poses much the same challenges as single image processing. Amongst these tasks are denoising [13, 17, 39, 73, 97, 100, 106, 29, 112, 137, 139, 138], inpainting [79, 94], super-resolution [4, 7, 23, 46, 47, 52, 53, 54, 64, 69, 70, 71, 76, 80, 88, 93, 91, 93, 95, 96, 111, 113, 114, 121, 125, 127, 141], and many more.

Having several images of the same scene, if used correctly, can be greatly beneficial to the restoration results. For denoising, it allows having several noisy samples of the same data. For inpainting, hidden details may become revealed in other images. Different samplings of the scene can also be used to recover sub-pixel details, thus creating an image with better resolution, a task known as super-resolution. In fact, most restoration tasks are likely to greatly benefit, in one way or another, from the availability of many frames, offering temporal redundancy which can - and should - be exploited.

The first step in exploiting temporal redundancy is inferring the connection between the images. This connection is what sets apart treating an image sequence from treating a random set of images. The connection is usually inferred by estimating the motion between the frames in the sequence, i.e., detecting for each pixel in each image its locations in every other image [39, 73, 97, 106, 112, 139, 138, 131]. In some cases, when the motion is of global nature (e.g., an affine transform), it is relatively simple to accurately estimate the motion trajectories.

However, most sequences contain very complex motion patterns, of non-rigid shapes and with many occlusions. In such cases, motion estimation is a severely under-determined
problem, and is very prone to errors and inaccuracies. Thus, many algorithms relying on motion estimation attempt to detect areas where the motion estimation is unreliable, and turn to single image processing mechanisms for those areas.

Several recent algorithms, developed for the denoising of image sequences, show that example-based methods are able to bypass the classic explicit motion estimation need and yet lead to state-of-the-art results [13, 17, 100, 29, 137]. This approach offers great possibilities in image sequence restoration, and this is the path we follow in this work.

While this entire thesis focuses on motion-estimation-free algorithms, it is important to note that the alternative approach of using sophisticated algorithms to obtain very accurate optical flow can also be pursued. Recent results indicate that high accuracy is indeed possible [87], which can possibly be harnessed to applications such as denoising and super-resolution.

1.5 Overview of the Thesis

The first task we tackle is image sequence video denoising. Our path starts with the very successful sparse and redundant representations model for images [43, 44]. We then test several ways to extend it to perform denoising on video-sequences, selecting those extensions that lead to the best results. The extensions result in a motion-estimation-free algorithm for image sequence denoising, which also displays state-of-the-art results. This research has been summarized in [100] and is detailed in Chapter 2.

With the intuition gained from this work, we proceed to handle a much more complicated problem, super-resolution, which deals with merging several low-quality images into one (or several) images with better optical resolution. Since the introduction of this field, it has relied on estimating the motion between the images for the merging procedure [4, 7, 23, 46, 47, 52, 53, 54, 64, 69, 70, 71, 76, 80, 88, 93, 91, 93, 95, 96, 111, 113, 114, 121, 125, 127, 141]. However, the accuracy demanded from the motion estimation limited its applicability to only that small portion of sequences displaying simple motion. Circumventing the need for motion estimation allows super-resolution processing of sequences containing even complex motion patterns.

Our approach to bypassing motion estimation in super-resolution starts with the Non-Local-Means (NLM) video denoising algorithm [17]. By carefully analysing it, we develop a mathematical framework for this intuitively suggested algorithm. This framework is then
adapted to handle the different scales involved in super-resolution, finally arriving at an algorithm for super-resolution, which is reminiscent of the NLM. The resulting algorithm does not rely on explicit motion estimation, and we show that it is indeed able to process sequences containing complex, local motion patterns. This work has been reported in [102], and detailed in Chapter 3.

We then revisit the super-resolution problem, again with the goal of avoiding motion estimation, while trying to offer a simpler and more intuitive approach. This attempt starts with the classic super-resolution framework, and replaces the reliance on estimated motion with probabilistic motion estimation. This approach has been reported in [101], and detailed in Chapter 4. Interestingly, under some simplifying assumptions, these two approaches lead to the same simple algorithm. However, they are not identical, as the generalizations offered by each is different.

In Chapter 5 we return to denoising of signals, regardless of their source. We want to build on the intuition gained in our previous work. In image sequences, it turned out that while an explicit model exists (i.e., the motion field between the images), using an implicit model (such as probabilistic motion estimation) can be a useful way to treat it.

It turns out that for signals created using the sparse and redundant representation model, using an implicit model to denoise the signal can improve the denoising results, even though an explicit model (the sparse representation used to generate the signal) indeed exists. This is achieved by averaging several semi-randomly generated sparse representations for the noisy signal, instead of seeking the sparsest one alone [49]. Mathematically, this averaging approximates the Minimum-Mean-Squared-Error (MMSE) estimator, while the sparsest representation approximated the Maximum A-Posteriori (MAP) estimator. Approximations are used since both the MAP and MMSE estimators require combinatorial work to compute, and is usually impractical. In our work, reported in [104] and detailed in Chapter 5, we show that when the dictionary is unitary, an approximation is not required, as closed-form solutions for both the MMSE and the MAP are achievable, as well as the benefit for denoising of using the MMSE.

Lastly, in Chapter 6, we summarize this thesis, outlining its main intuitions and contributions. We also list several future directions we believe are promising.
Chapter 2

Image Sequence Denoising via Sparse and Redundant Representations


In this chapter we consider denoising of image sequences that are corrupted by zero-mean additive white Gaussian noise. Relative to single image denoising techniques, denoising of sequences aims to also utilize the temporal dimension. This assists in getting both faster algorithms and better output quality. This chapter focuses on utilizing sparse and redundant representations for image sequence denoising, extending the work reported in [43, 44]. In the single image setting, the K-SVD algorithm is used to train a sparsifying dictionary for the corrupted image. This chapter generalizes the above algorithm by offering several extensions: (i) the atoms used are three-dimensional; (ii) the dictionary is propagated from one frame to the next, reducing the number of required iterations; and (iii) averaging is done on patches in both spatial and temporal neighboring locations. These modifications lead to substantial benefits in complexity and denoising performance, compared to simply running the single image algorithm sequentially. The algorithm’s per-
formance is experimentally compared to several state-of-the-art algorithms, demonstrating comparable or favorable results.

2.1 Introduction

Denoising of images is one of the most basic tasks of image processing, and as such, it has been extensively studied in the past several decades. This problem is the simplest among the family of problems known as Inverse Problems, aiming to recover a high quality signal from a degraded version of it. There is a wealth of single image denoising algorithms; a comprehensive review of these techniques can be found in [15] and in [98].

Denoising image sequences extends the above task to handle the temporal dimension as well. Such sequences can be, TV broadcast, camcorder files, and more. In many cases, one can assume the noise to be an additive zero-mean white Gaussian noise, as common also in the still image denoising literature. Algorithms for the denoising of image sequences aim to remove the additive noise while utilizing both the spatial and the temporal domains. Such an approach is expected to lead to a gain both in the denoising performance and the computational load, when compared to applying a single image denoising algorithm to each image separately. These desired and expected gains emerge from the high temporal redundancy in image sequences. Indeed, in many cases, image sequences are noisier than single images due to the high capture rate, making the use of the temporal dimension that much more important.

Denoising of video sequences attracted some attention in the past decade, with various suggested algorithms and principles. One suggested approach that utilizes the temporal redundancy is motion estimation [139, 73, 39, 140]. The estimated trajectories are used to filter along the temporal dimension, either in the wavelet domain [139, 73, 140] or the signal domain [39]. Spatial filtering may also be used, with stronger emphasis in areas in which the motion estimation is not as reliable. A similar approach described in [97] detects for each pixel whether it has undergone motion or not. Spatial filtering is applied to each image, and for each pixel with no motion detected, the results of the spatial filtering are recursively averaged with results from previous frames. The method described in [137] employs a similar principle with a fuzzy logic used to replace the binary motion detection.

A different approach to video denoising is treating the image sequence as a three-dimensional volume, and applying various transforms to this volume in order to attenuate
the noise. Such transforms can be a Fourier-wavelet transform [132], an adaptive wavelet transform [106], or a combination of 2D and 3D dual-tree complex wavelet transform [112].

A third approach towards video denoising employs spatio-temporal adaptive average filtering for each pixel. The method described in [138] uses averaging of pixels in the neighborhood of the processed pixel, both in the current frame and the previous one. The weights in this averaging are determined according to the similarity of \(3 \times 3\) patches around the two pixels matched. The method in [17] extends this approach by considering a full 3D neighborhood around the processed pixel (which could be as large as the entire sequence), with the weights being computed using larger patches and thus producing more accurate weights. An adaptive selection of the neighborhood size used for averaging is employed in [13] for obtaining improved results. The current state-of-the-art reported in [29] also finds similar patches in the neighborhood of each pixel; however, instead of a weighted averaging of the centers of these patches, noise attenuation is performed in the transform domain. More on some of these algorithms and their comparison to our proposed scheme is brought in Section 2.4.

In this chapter we explore a method that utilizes sparse and redundant representations for image sequence denoising, extending the work reported in [43, 44]. In the single image setting, the K-SVD algorithm, as presented in [2, 3], is used to train a sparsifying dictionary for the corrupted image, forcing each patch in the image to have a sparse representation describing its content. Put in a maximum a-posteriori probability (MAP) framework, the developed algorithm in [43, 44] leads to a simple algorithm with state-of-the-art performance for the single image denoising application.

This chapter extends the above algorithm by considering 3D (spatio-temporal) patches, a propagation of the dictionary over time, and averaging that is done on neighboring patches both in space and time. As the dictionary of adjacent frames (belonging to the same scene) is expected to be nearly identical, the number of required iterations per frame can be significantly reduced. Utilizing patches in nearby frames for the denoising process is also examined. All these modifications lead to substantial benefits both in complexity and denoising performance, outperforming all the recently published video denoising methods.

The structure of the chapter is as follows: In section 2.2 we describe the principles of sparse and redundant representations and their deployment to single image denoising. Section 2.3 discusses the generalization to video, discussing various options of using the temporal dimension with their expected benefits and drawbacks. Each proposed extension
is experimentally validated. Section 2.4 surveys the literature, describing several leading
and competitive video denoising algorithms. A performance comparison of these methods
and the one introduced in this chapter is given, demonstrating the superiority of the
proposed approach. Section 2.5 summarizes and concludes the chapter.

2.2 Image Denoising Using Sparsity and Redundancy

A method of denoising images based on sparse and redundant representations is developed
and reported in [43, 44]. In this section we provide a brief description of this algorithm,
as it serves as the foundation for the video denoising we develop in Section 2.3.

A noisy image $Y$ results from noise $V$ superimposed on an original image $X$. We assume
the noise $V$ to be white, zero-mean Gaussian noise, with a known standard deviation
$\sigma$,

$$Y = X + V,$$  \hspace{1cm} (2.1)

The basic assumption of the denoising method developed in [43, 44] is that each image
patch (of a fixed size) can be represented as a linear combination of a small subset of
patches (atoms), taken from a fixed dictionary. Using this assumption, the denoising task
can be described as an energy minimization procedure. The following functional describes
a combination of three penalties to be minimized:

$$f_{\text{Stat}} \left( \{\alpha_{ij}\}_{i,j}, X \right) = \lambda \|X - Y\|_2^2 + \sum_{ij \in \Omega} \|D\alpha_{ij} - R_{ij}X\|_2^2 + \sum_{ij \in \Omega} \mu_{ij} \|\alpha_{ij}\|_0.$$  \hspace{1cm} (2.2)

The first term demands a proximity between the measured image, $Y$, and its denoised (and
unknown) version $X$. The second term demands that each patch from the reconstructed
image (denoted by $^1 R_{ij}X$) can be represented up to a bounded error by a dictionary $D$,
with coefficients $\alpha_{ij}$. The third part demands that the number of coefficients required to
represent any patch is small. The values $\mu_{ij}$ are patch-specific weights. Minimizing this
functional with respect to its unknown yields the denoising algorithm.

The choice of $D$ is of high importance to the performance of the algorithm. In [43, 44]

\[\text{The matrix } R_{ij} \text{ stands for an operator that extracts a patch of fixed size from the image in location } [i,j].\]
relaxation method, that fixes all the unknowns apart from the one to be updated, and alternates between the following update stages:

1. **Update of the sparse representations** \( \{\alpha_{ij}\} \): Assuming that \( D \) and \( X \) are fixed, we solve a set of problems of the form

\[
\hat{\alpha}_{ij} = \arg\min_\alpha \|D\alpha - R_{ij}X\|_2^2 + \mu\|\alpha\|_0
\]  

per each location \([i, j]\). This means that we seek for each patch in the image the sparsest vector to describe it using atoms from \( D \). In [43, 44], the orthogonal matching pursuit (OMP) algorithm is used for this task [84, 123, 36].

2. **Update the dictionary \( D \)**: In this stage we assume that \( X \) is fixed, and we update one atom at a time in \( D \), while also updating the coefficients in \( \{\alpha_{ij}\}_{ij} \) that use it. This is done via a rank-one approximation of a residual matrix, as described in [62, 2, 3].

3. **Update the estimated image \( X \)**: After several rounds of updates of \( \{\alpha_{ij}\}_{ij} \) and \( D \), the final output image is computed by fixing these unknowns and minimizing (2.2) with respect to \( X \). This leads to the quadratic problem

\[
\hat{X} = \arg\min_X \lambda\|X - Y\|_2^2 + \sum_{ij} \|D\alpha_{ij} - R_{ij}X\|_2^2,
\]  

which is solved by a simple weighting of the represented patches with overlaps, and the original image \( Y \).

The improved results obtained by training a dictionary based on the noisy image itself stem from the dictionary adapting to the content of the actual image to be denoised. An added benefit is that the K-SVD algorithm has noise averaging built into it, by taking a large set of noisy patches and creating a small, relatively clean representative set. More recently, the above described algorithm was generalized to handle color image denoising, demosaicing, and inpainting, leading in all these applications to state-of-the-art results [83].

### 2.3 Extension to Video Denoising

In this section we describe, step by step, the proposed extensions to handling image sequences. We also provide experimental results for each proposed extension, validating
its efficiency. A description of the test set and a comparison of the overall algorithm to other state-of-the-art methods appear in the next section. To validate the efficiency of the proposed extension steps, we super-impose white Gaussian noise on several sequences, which are then denoised and quantitatively compared. The measure of quality of the denoising result $\hat{X}$ versus the original signal $X$ is the Peak-Signal-to-Noise-Ratio (PSNR), given by

$$ PSNR = 10 \log_{10} \left( \frac{255^2 \cdot p}{\| X - \hat{X} \|_2^2} \right) \ [\text{dB}], $$

where both signals use the scale $0 - 255$. The PSNR is computed for each image in the sequence, and then averaged over the entire sequence, for assessment of the overall denoising quality.

2.3.1 Constructing The Algorithm

Considering the objective function in Equation (2.2), extending it to handle image sequences might seem to be a simple task. By letting $Y$ and $X$ represent the noisy and clean videos respectively (instead of the noisy and clean images), and adding an index $t$ in the range $[1, T]$ to account for the time dimension, we arrive at a desired penalty term that contains all the forces described in the single image denoising setting. This formulation forms one MAP energy function for the entire sequence:

$$ f_{\text{All Video}} (\{ \alpha_{ijt} \}_{ijt}, X, D) = \lambda \| X - Y \|_2^2 + \sum_{ij \in \Omega} \sum_{t=1}^{T} \mu_{ijt} \| \alpha_{ijt} \|_0 + \sum_{ij \in \Omega} \sum_{t=1}^{T} \| D \alpha_{ijt} - R_{ijt} X \|_2^2. $$

The term $R_{ijt} X$ extracts a patch of a fixed size from the volume $X$ in time $t$ and spatial location $[i, j]$. This patch may be 3D in general, this way exploiting the temporal axis to our benefit.

Minimizing this functional with respect to its unknows generates a single dictionary for the entire sequence, and cleans all the images at once with such a dictionary. The transition to three dimensions appears in the finer details of the algorithm. The patches are transformed into 3D ones, in the sense that they can contain pixels from more than one image. A 3D patch is created by taking a block around the pixel $(i, j, t)$ that extends by $(\Delta i, \Delta j, \Delta t)$ in each axis respectively. This makes the patch symmetrical around the center image, and is therefore not causal\(^2\). This structure also requires paying special

\(^2\)Causality can be enforced, but was found to lead to a slightly inferior performance.
attention to the dictionary initialization. In our tests, the basis for the initial dictionary is the same 2D overcomplete DCT dictionary. Each atom is then replicated \((2\Delta t + 1)\) times to create a three dimensional atom of the wanted temporal width.

As already mentioned, in the penalty term in Equation (2.5) all the patches in the sequence are used for training a single dictionary, that is then applied to the entire sequence. However, training a single dictionary for the entire sequence is problematic; The scene is expected to change rapidly, and objects that appear in one frame might not be there five or ten frames later. This either means that the dictionary will suit some images more than others, or that it would suit all of the images but only moderately so. Obtaining state-of-the-art denoising results requires better adaptation.

An alternative approach could be proposed by defining a locally temporal penalty term, that on one hand allows the dictionary to adapt to the scene, and on the other hand, exploits the benefits of the temporal redundancy. A natural such attempt is rewriting the penalty in Equation (2.5) for each image separately,

\[
\begin{align*}
f^{(t)}_{\text{Video}} (\{\alpha_{ij}\}_{ij}, X_t, D_t) &= \lambda \|X_t - Y_t\|_2^2 + \\
&\quad + \sum_{ij \in \Omega} \mu_{ij} \|\alpha_{ij}\|_0 + \sum_{ij \in \Omega} \|D_t \alpha_{ij} - R_{ijt} X\|_2^2, \end{align*}
\]
defined for \(t = 1, 2, \ldots, T\).

The temporal repetitiveness of the video sequence can be further used to improve the algorithm. As consecutive images \(X_t\) and \(X_{t-1}\) are similar, their corresponding dictionaries are also expected to be similar. This temporal coherence can help speed-up the algorithm. Fewer training iterations are necessary if the initialization for the dictionary \(D_t\) is the one trained for the previous image.

Returning to the first step of migrating from 2D to 3D patches, how important is this change? To answer this question we test the effects of two and three dimensional atoms on the performance of the overall video denoising algorithm (with a temporally adaptive and propagated dictionaries). The results of such a comparison appears in Figure 2.1. For the 3D case, one set of parameters (noise-level dependent) is used for all movies. The 3D atoms used in these tests are 5 images wide (two images away in each direction). For the 2D case, it is difficult to find one set of parameters that does justice to all movies. We therefore use the optimal set found for each movie, as this does not change the conclusions drawn from this comparison.

The performance gain achieved by using 3D atoms is very noticeable. A better under-
Figure 2.1: PSNR gain (difference in dB relative to the 2D atoms’ method) achieved by using 3D atoms versus 2D atoms.

Understand the reasons the 3D atoms outperform the 2D ones can be gained by looking at the content of the dictionary. Figure 2.2 shows several atoms from the trained 3D dictionary, for image #10 of the “garden” sequence (several frames from this sequence appear at the end of this section). This sequence has a camera motion to the right, so the entire scene moves to the left. The described motion can be seen clearly in the atoms. The central part of the atom (coming from the current image) has moved to the left compared to the top part (coming from the previous image). In the same manner, the bottom part (coming from the next image) has moved to the left relative to the center part. The question of what happens to the dictionary when the motion is not global naturally arises. In such cases, the dictionary has several atoms reflecting each of the patterns and motions.

Figure 2.2: Several (8) 3D atoms (each of size $8 \times 8 \times 5$, but only the temporal center is shown) trained for image #10 of the garden sequence. Each vertical column is one atom.

To gauge the possible speed-up and improvement achieved by temporally adaptive and
propagated dictionary, we test the required number of iterations to obtain similar results to the non-propagation alternative. Figure 2.3 presents the results of such an experiment. Several options for the number of training iterations that follow the dictionary propagation are compared to the non-propagation (using 15 training iterations per image) option.

![Figure 2.3: PSNR gain (average of all sequences) by propagating the dictionary and using the specified number of training iterations for each image. The reference option is with no propagation, and 15 K-SVD iterations.](image)

The clear conclusion from these results is that propagation is crucial and leads to improved denoising performance. More insight to the reasons for such an improvement can be seen in Figure 2.4, that shows the dictionaries trained for frame #30 of the “garden” sequence. The left dictionary is the one trained after propagating the dictionary (from image #10), using 4 training iterations for each image. The right part of the figure shows the trained dictionary from the DCT using 15 training iterations. This comparison shows that propagation of the dictionary leads to a cleaner version with clearer and sharper texture atoms. These benefits are attributed to the memory induced by the propagation. Indeed, when handling longer sequences, we expect this memory feature of our algorithm to further benefit in denoising performance. This was verified in tests on longer sequences.

A second conclusion is that the number of training iterations should not be constant, but rather depend on the noise level. It appears that the less noisy the sequence, the

---

3In a real system, employing dictionary propagation requires some scene-cut detection algorithm to reset the dictionary when the scene changes at once.
Figure 2.4: Dictionaries trained by propagating (left) and not propagating (right) the dictionary between images. Top: Central part (time-wise) of each dictionary. Bottom: Several enlarged atoms, showing the three center temporal layers.
more training iterations are needed for each image. In higher noise levels, adding more iterations hardly results in any denoising improvement.

So far we discussed the use of 3D atoms and a temporally adaptive dictionaries. However, in the formulation written in Equation (2.6), only patches centered in the current image are used for training the dictionary and cleaning the image. In the global temporal term as in Equation (2.5), all the patches in the sequence were used for these tasks. A compromise between temporal locality and exploiting the temporal redundancy is again called for. This compromise is achieved by using patches centered in a limited number of neighboring images of the image currently denoised, both in training and cleaning. Introducing this into the penalty term in Equation (2.6) leads to the modified version,

\[
J_{\text{Video}}^{(t \pm \Delta t)} \left( \{ \alpha_{ijk} \}_{ijk}^t, X_t, D_t \right) = \lambda \| X_t - Y_t \|_2^2 + \\
+ \sum_{ij \in \Omega} \sum_{k=t-\Delta t}^{t+\Delta t} \mu_{ijk} \| \alpha_{ijk} \|_0 + \sum_{ij \in \Omega} \sum_{k=t-\Delta t}^{t+\Delta t} \| D_t \alpha_{ijk} - R_{ijk} X \|_2^2,
\]

defined for \( t = 1, 2, \ldots, T \).

The effectiveness of the extended training set can be seen by observing Figure 2.5, in which we show the gain (or loss) in PSNR achieved by also using patches centered one image away (i.e., the patches are taken from 3 frames). All the tests in this experiment are done on one quarter of the patches in the spatio-temporal region (chosen randomly) so as to lead to a reduced complexity algorithm.

It is visible from the graph that using an extensive set indeed results in an improved performance. Tests taking patches also from two images away were also run, however there was no significant advantage in performance to justify this additional computational burden.

### 2.3.2 Overall Algorithm and Parameter Selection

The penalty term in Equation (2.7) is the penalty term we target in the algorithm that follows. The algorithm for minimizing this functional is founded on the same principles as the ones described in Section 2.2, with the obvious modifications due to the 3D nature of the treatment done here. Three visual examples of the denoising results can be seen in Figures 2.6, 2.7, and 2.8.

We have run many tests to tune the various parameters of the proposed algorithm, which have a crucial effect on the overall denoising performance. These tests have resulted in a selection of a single set of parameters (as a function of noise level). In conjunction with
Figure 2.5: PSNR gain achieved by also using patches that are one image away for training and cleaning.

Figure 2.6: Football Sequence (frames 15 and 70) with $\sigma = 30$. Left: Original frame; Middle: Noisy frame; Right: Cleaned frame.
Figure 2.7: Tennis Sequence (frames 15 and 60) with $\sigma = 20$. Left: Original frame; Middle: Noisy frame; Right: Cleaned frame.

Figure 2.8: Garden Sequence (frames 10 and 20) with $\sigma = 50$. Left: Original frame; Middle: Noisy frame; Right: Cleaned frame.
the previously described experiments, this set of parameters includes 3D atoms (which are 5 images wide), propagation of the dictionary with the number of training iterations being noise-level dependent, and a training set which extends one image in each temporal direction. During the experimentation with other parameters, we have found, for example, that higher noise levels require that the spatial size of the blocks is slightly smaller than at low noise levels.

We have also found that at high noise levels, the redundancy factor (the ratio between the number of atoms in the dictionary to the size of an atom) should be smaller. At high noise levels, obtaining a clean dictionary requires averaging of a large number of patches for each atom. This is why only a relatively small number of atoms is used. At low noise levels, many details in the image need to be represented by the dictionary. Noise averaging takes a more minor role in this case. This calls for a large number of atoms, so they can represent the wealth of details in the image.

A comparison between the final 3D algorithm and the original single image algorithm can be seen in table 2.4, which appears in the next section.

2.3.3 Complexity of the Algorithm

The algorithm is divided into two parts: (i) dictionary training and (ii) image cleaning. The dictionary training is an iterative process, of repeatedly running sparse coding (OMP) followed by a dictionary update (SVD step). The image cleaning is composed of a simple per-pixel averaging. In order to analyze the complexity of the algorithm, we present the following notations: $n$ – the number of pixels in an atom; $d$ – the number of atoms in the dictionary; $\ell$ – the average number of atoms used in the representation of a patch; $p$ – the number of pixels in one frame; $2\Delta t + 1$ – the number of frames taken into account in the training and denoising; and $J$ – the number of training iterations.

The sparse-coding stage – solving the problem posed in Equation (2.3) using the OMP algorithm – requires $n \cdot d \cdot \ell$ operations for one patch [84]. Applying this to each of the $(2\Delta t + 1)p$ patches in the spatio-temporal window requires $n \cdot d \cdot \ell \cdot (2\Delta t + 1) \cdot p$ operations. The update of the dictionary requires $n \cdot \ell \cdot (2\Delta t + 1) \cdot p$ operations [2, 3]. The image cleaning is done by a simple averaging of patches, requiring $n \cdot (2\Delta t + 1) \cdot p$ operations. Since there are $J$ iterations of sparse coding and dictionary update, and one final estimate of the output image, the overall algorithm complexity is given by

$$\text{Complexity} = [J(d + 1)\ell + 1] \cdot n \cdot (2\Delta t + 1) \cdot p \text{ Operations/Frame.}$$
Let us illustrate this complexity for a nominal case: The value of $\ell$ is noise-level dependant, being $\approx 1$ for $\sigma = 10 - 25$. The number of training iterations $J$ is 2 when propagating the dictionary. Assuming that we process patches of size $n = 125$ pixels each ($5 \times 5 \times 5$ patches) using a dictionary with $d = 300$ atoms, over a window of one frame ($\Delta t = 0$), there are $\approx 75,000$ operations per pixel. This very demanding algorithm is essentially such because of the need to multiply the patches by the dictionary – a matrix of size $d \times n$.

Reducing the complexity of the algorithm is a necessary step in turning the algorithm into a reasonable one. There are several methods of reducing its complexity:

1. The image can be divided into several parts (with small overlaps), and a smaller dictionary (i.e. with a smaller number of atoms) can be trained for each part. This saves computations, as each patch considers less atoms, thus reducing the value of the effective $d$ in the above formula. Further, one part’s dictionary can be used for initialization for all other parts, saving more computations. Several tests using this approach indicate that not only does this lead to a speedup factor of $4 - 10$, it actually leads to an improvement (approx. $0.1 - 0.2$dB in our tests) in the denoising performance. We believe that delicate parameter tuning for this approach will result in a more substantial improvement, especially in weak noise scenarios.

2. Choosing non-optimal values for the parameters of the algorithm is another way to control complexity. One of the parameters is the temporal extent ($\Delta t$), i.e. from which images are patches drawn for the training process (it is important not to confuse with the temporal extent of the atoms, which should remain 3D). Using only patches from the current image (and not from neighboring images) cuts complexity by a factor of 3 and costs approx. $0.2 - 0.4$dB. Reducing the overlap between patches, by using only every other patch in each axis, reduces complexity by a factor of 4, and costs only around $0.1$dB. Combining them causes a reduction of more than $0.5$dB, probably because too few patches are left for use.

3. Running all iterations, but maybe the last one (which is also the cleaning iteration) on a subset of the patches, doesn’t cause noticeable degradation in performance. When using $J = 2$, this can gain almost a factor of 2 in the overall complexity.

4. The core operation of matrix multiplication can be replaced by an approximation of it (e.g., using singular value decomposition (SVD) on the dictionary [62]). We have not explored this option in detail, and thus cannot report of its effectiveness.
5. The facts that OMP is done independently on each patch, and the K-SVD independently on each atom, lend this algorithm easily to a parallel implementation on any number of processors, again leading to a substantial speedup. Since 8-CPU structures or equivalent FPGA are currently available at affordable prices, and due to the parallel nature of almost the entire algorithm, this can be viewed as a viable option for reducing the overall run-time by about one order of magnitude.

To summarize, by making slight adaptations to the algorithm (e.g. dividing the image to parts, controlling the overlap between patches, and more), 1 – 2 orders of magnitude in the number of computations can be gained without a noticeable drop in performance. This leads to a rough estimate of 2,000 operations per pixel, which is definitely more reasonable. Note that another one order of magnitude can be gained by a parallel implementation.

The entire simulations described in this and the next section were run with a non-optimized Matlab implementation of the proposed algorithm on a 2.4GHz Pentium with 2Gb RAM, with the required variations for each test. Using our implementation, processing a CIF (approx. 280 × 360) frame with this Matlab implementation requires 5 – 120 seconds, depending on the noise level (higher noise level requires less time) and the content of the scene (more textured scenes requiring more time).

2.4 Comparison to State-of-the-Art

2.4.1 Overview of other methods

We compare the proposed algorithm to four methods that have been show to display state-of-the-art results – those are reported in [140, 17, 13, 29]. We do not provide a comprehensive comparison to the algorithm reported in [138] since it is a simplified version of the one described in [17].

The method described in [140] operates fully in the wavelet domain. Motion estimation and adaptive temporal filtering (along the estimated trajectories) are applied recursively, followed by an intra-frame spatially adaptive filter. Two types of motion reliability measures are estimated. One is a reliability measure per orientation, applied in the motion estimation stage. The other is a reliability measure per wavelet band, effecting the parameters of the temporal filter. The subsequent spatial filtering is designed to have an increased effect where the temporal filtering had been less effective due to low reliability.
The Non-Local Means (NL-Means) algorithm reported in [15, 17] takes an alternative approach to the problem. Instead of an explicit motion estimation for each patch, all the patches in its three dimensional neighborhood are considered and their center pixels are averaged. Each patch is weighted according to its similarity to the center patch, making the motion-estimation a fuzzy and implicit one. Instead of computing a single motion vector for each patch, several possible vectors are allowed to co-exist, each with a different probability. As contributing patches may also appear within the same image, the interpretation of this approach as fuzzy motion estimation is inaccurate; still, it provides some intuition into the success of this approach. This approach focuses on the fusion of noisy estimates rather than obtaining accurate motion estimation.

The method described in [13] extends the NL-Means approach. Statistical measures are used for optimal adaptive selection of the neighborhood size for each pixel. Furthermore, the parameters for computing the weights assigned to each pixel in the neighborhood are also adaptively selected. These powerful tools result in improved denoising performance compared to the original NL-Means approach.

The VBM3D, reported in [29], also uses a multitude of patches in the three-dimensional neighborhood of each pixel for attenuating the noise. However, the patches are used in a different manner. The most similar patches in the neighborhood are collected and stacked into a 3D array. A 3D wavelet transform is then applied, with hard-thresholding used for noise suppression. After the inverse transform is applied, the patches are returned to their original locations, and averaged. A second iteration follows, with Wiener filtering used to improve denoising results.

We now turn to present a comprehensive performance comparison between these four methods and the proposed algorithm. Note that since the above papers chose different video sequences to test on, we provide several groups of tests, to address each.

2.4.2 Comparison Results

The comparison to the method reported in [140] was done on the sequences appearing in that paper (and also found in the first author’s website). The mean PSNR results of our proposed method, the results of [140], and the differences between them (for frames 5−35 of each sequence) all appear in Table 2.1. Averaging over all these tests, the proposed method outperforms the one reported in [140] by 1.63dB, and specifically on each and every test.
We now turn to compare the proposed method to the work reported in [13], which displayed superior denoising results relative to other methods it was compared to. We synthesized the same experiments as those in [13], and report the results in Table 2.2. Again, it is clear that the proposed method performs better in these tests with an average gain of 1.79dB.

We also compare the proposed method to the results of the classic NL-Means ([15, 17]) and to the current benchmark in video denoising, the VBM3D ([29]). The tests were run on a set of four different image sequences - “Football”, “Tennis”, “Flower Garden”, and “Mobile”. Each of the four test sets is superimposed with synthetic white Gaussian noise, using noise levels $\sigma = 5, 10, 15, 20, 25, 30, 35, 40,$ and $50$. The translation between noise level and mean PSNR of the noisy sequences appears in Table 2.3, as the clipping of out-of-range gray-values causes some variation, especially noticed in the strong noise cases.

For the NL-Means algorithm, we used our implementation to obtain denoising results for the test sequences. For fairness, we have varied the parameters, searching for the optimal configuration. It is worth to note that there was no one set of parameters fitting all sequences at a defined noise level. Instead, we report the results when all the parameters have been optimized for each specific test, keeping in mind that this gives the NL-Means algorithm an advantage in these comparisons.

<table>
<thead>
<tr>
<th>Test Sequence</th>
<th>Input Noise Level</th>
<th>Results from [140] [dB]</th>
<th>Proposed Method Results [dB]</th>
<th>Difference [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garden</td>
<td>10</td>
<td>30.86</td>
<td>32.22</td>
<td>+1.36</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>28.24</td>
<td>29.78</td>
<td>+1.54</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>26.46</td>
<td>28.08</td>
<td>+1.62</td>
</tr>
<tr>
<td>Tennis</td>
<td>10</td>
<td>32.08</td>
<td>33.78</td>
<td>+1.70</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>29.46</td>
<td>31.65</td>
<td>+2.19</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>28.29</td>
<td>30.18</td>
<td>+1.89</td>
</tr>
<tr>
<td>Salesman</td>
<td>10</td>
<td>35.84</td>
<td>37.95</td>
<td>+2.11</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>33.91</td>
<td>35.17</td>
<td>+1.26</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>32.37</td>
<td>33.33</td>
<td>+0.96</td>
</tr>
</tbody>
</table>

Table 2.1: Results of the proposed algorithm compared to those reported in [140]. The chosen sequences and noise powers are those reported in [140].
<table>
<thead>
<tr>
<th>Test Sequence</th>
<th>Input PSNR [dB]</th>
<th>[13] Results PSNR [dB]</th>
<th>Our Results PSNR [dB]</th>
<th>Difference [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesman</td>
<td>28</td>
<td>35.13</td>
<td>37.91</td>
<td>+2.78</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>32.60</td>
<td>35.59</td>
<td>+2.99</td>
</tr>
<tr>
<td>Garden</td>
<td>28</td>
<td>31.33</td>
<td>32.13</td>
<td>+0.80</td>
</tr>
<tr>
<td>Miss America 1</td>
<td>28</td>
<td>39.39</td>
<td>40.49</td>
<td>+1.10</td>
</tr>
<tr>
<td>Suzie</td>
<td>28</td>
<td>37.07</td>
<td>37.96</td>
<td>+0.89</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>35.11</td>
<td>35.95</td>
<td>+0.84</td>
</tr>
<tr>
<td>Trevor</td>
<td>28</td>
<td>36.68</td>
<td>38.10</td>
<td>+1.42</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>34.79</td>
<td>35.97</td>
<td>+1.18</td>
</tr>
<tr>
<td>Foreman</td>
<td>28</td>
<td>34.94</td>
<td>37.86</td>
<td>+2.92</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>32.90</td>
<td>35.86</td>
<td>+2.96</td>
</tr>
</tbody>
</table>

Table 2.2: Results of the proposed algorithm compared to those reported in [13]. The chosen sequences and noise powers are those reported in [13].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>34.1446</td>
<td>34.1596</td>
<td>34.1623</td>
<td>34.1785</td>
</tr>
<tr>
<td>15</td>
<td>24.6191</td>
<td>24.6201</td>
<td>24.7063</td>
<td>24.6875</td>
</tr>
<tr>
<td>20</td>
<td>22.1287</td>
<td>22.1398</td>
<td>22.2673</td>
<td>22.2590</td>
</tr>
<tr>
<td>25</td>
<td>20.2348</td>
<td>20.2096</td>
<td>20.3964</td>
<td>20.4008</td>
</tr>
<tr>
<td>30</td>
<td>18.6824</td>
<td>18.6517</td>
<td>18.8939</td>
<td>18.9238</td>
</tr>
<tr>
<td>35</td>
<td>17.3685</td>
<td>17.3242</td>
<td>17.6543</td>
<td>17.6862</td>
</tr>
</tbody>
</table>

Table 2.3: The PSNR of noisy sequences for each sequence and noise level combination. The difference is due to the out-of-range values.
For the VBM3D, the authors of [29] were very kind to provide us with an implementation of their algorithm (which later was made available in their website). We compare in Table 2.4 the results obtained by four algorithms: VBM3D [29], NL-Means [17], the original K-SVD denoising applied on single images [43], and the proposed algorithm that is proposed in this chapter.

<table>
<thead>
<tr>
<th>σ</th>
<th>Football</th>
<th>Tennis</th>
<th>Garden</th>
<th>Mobile</th>
<th>MeanPSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>36.96</td>
<td>35.95</td>
<td><strong>38.32</strong></td>
<td>37.12</td>
<td>36.84</td>
</tr>
<tr>
<td></td>
<td>36.03</td>
<td><strong>37.42</strong></td>
<td>35.45</td>
<td>38.16</td>
<td>35.74</td>
</tr>
<tr>
<td>10</td>
<td>32.87</td>
<td>31.82</td>
<td><strong>34.60</strong></td>
<td>32.05</td>
<td><strong>32.38</strong></td>
</tr>
<tr>
<td></td>
<td>31.77</td>
<td><strong>33.29</strong></td>
<td>30.89</td>
<td>34.33</td>
<td>30.72</td>
</tr>
<tr>
<td>15</td>
<td>30.74</td>
<td>29.61</td>
<td><strong>32.46</strong></td>
<td>29.56</td>
<td><strong>30.16</strong></td>
</tr>
<tr>
<td></td>
<td>29.66</td>
<td><strong>30.96</strong></td>
<td>28.81</td>
<td>32.10</td>
<td>28.03</td>
</tr>
<tr>
<td>20</td>
<td>29.22</td>
<td>28.14</td>
<td><strong>30.76</strong></td>
<td>28.14</td>
<td><strong>28.58</strong></td>
</tr>
<tr>
<td></td>
<td>28.25</td>
<td><strong>29.39</strong></td>
<td>27.64</td>
<td>30.51</td>
<td>26.25</td>
</tr>
<tr>
<td>25</td>
<td>28.07</td>
<td>27.12</td>
<td>29.10</td>
<td>27.35</td>
<td><strong>27.33</strong></td>
</tr>
<tr>
<td></td>
<td>27.24</td>
<td><strong>28.24</strong></td>
<td>26.83</td>
<td>29.32</td>
<td>24.89</td>
</tr>
<tr>
<td>30</td>
<td>27.19</td>
<td>26.52</td>
<td>27.86</td>
<td>26.68</td>
<td><strong>26.26</strong></td>
</tr>
<tr>
<td></td>
<td>26.47</td>
<td><strong>27.23</strong></td>
<td>26.31</td>
<td><strong>28.43</strong></td>
<td>23.84</td>
</tr>
<tr>
<td></td>
<td>25.81</td>
<td><strong>26.50</strong></td>
<td>25.83</td>
<td><strong>27.75</strong></td>
<td>22.96</td>
</tr>
<tr>
<td>40</td>
<td>25.84</td>
<td>24.97</td>
<td>26.45</td>
<td>25.93</td>
<td><strong>24.49</strong></td>
</tr>
<tr>
<td></td>
<td>25.22</td>
<td><strong>25.86</strong></td>
<td>25.52</td>
<td><strong>27.22</strong></td>
<td>22.20</td>
</tr>
<tr>
<td>50</td>
<td><strong>24.81</strong></td>
<td>24.11</td>
<td>25.79</td>
<td>25.34</td>
<td>22.48</td>
</tr>
<tr>
<td></td>
<td>24.12</td>
<td>24.73</td>
<td>24.92</td>
<td><strong>26.34</strong></td>
<td>21.03</td>
</tr>
</tbody>
</table>

Table 2.4: Comparison of the denoising results of several methods on a number of test sequences and noise levels. Top Left: VBM3D; Top Right: NL-Means; Bottom Left: K-SVD single image; and Bottom Right: The proposed algorithm. The best result for each set is written in bold. Results are for images 10-20 of each set, using other images (for temporal filtering) as necessary.

Averaging the above results, we get an average performance (from the best downwards) of 29.23dB for our method, 28.9dB for the VBM3D, 27.92dB for the NL-Means, and finally, 27.08dB for the single-frame K-SVD algorithm. The proposed method is slightly favorable compared to the VBM3D, obtaining better mean PSNR at every noise level. We also note that these results mean that the proposed extension of [43, 44] to handle video yields about 2dB better results on average than the single image method. This comes only to prove the necessity and potential of using the temporal axis in video denoising.
As a final experiment, we compare the visual quality of the results produced by the three algorithms - the VBM3D, the NL-Means, and the proposed method. This comparison appears in Figure 2.9, along with the original high-quality image. We deliberately show results for very strong noise ($\sigma = 40$), since all these methods are very effective and low-noise cases appear to be near-perfect and with only delicate differences.

Figure 2.9: Visual comparison of denoising results for one image of the Mobile sequence with noise level 40. Top Left: Original. Top Right: NL-Means. Bottom Left: VBM3D. Bottom Right: Proposed Method.

2.5 Conclusions

In this chapter we propose an image sequence denoising algorithm based on sparse and redundant representation. This algorithm is based on the single image denoising algorithm introduced in [43, 44]. The extension of this basic algorithm to handle image sequences is discussed both on the mathematical level and in practical terms. Three extensions are proposed: the use of spatio-temporal (3D) atoms, dictionary propagation coupled with fewer training iterations, and an extended patch-set for dictionary training and image cleaning. All these extensions are thoroughly tested on an extensive set of image sequences and noise levels, and are found to dramatically improve denoising performance. The
proposed algorithm is also compared to other state-of-the-art methods, and shown to produce comparable or favorable results.

Acknowledgements

We thank the authors of [29], D. Rusanovskyy, K. Dabov, and Prof. K. Egiazarian, for their willingness to provide us with a working implementation of the BM3D algorithm. A special thanks goes to K. Dabov, who provided us with the source code of the improved BM3D (coined VBM3D) and the test sequences used to test it on. We would also like to thank the authors of [13], J. Boulanger, C. Kervrann, and P. Bouthemy, for their willingness to help in comparing the denoising methods.
Chapter 3

Generalizing the Non-Local-Means to Super-Resolution Reconstruction


Super-resolution reconstruction proposes a fusion of several low quality images into one higher quality result with better optical resolution. Classic super resolution techniques strongly rely on the availability of accurate motion estimation for this fusion task. When the motion is estimated inaccurately, as often happens for non-global motion fields, annoying artifacts appear in the super-resolved outcome. Encouraged by recent developments on the video denoising problem, where state-of-the-art algorithms are formed with no explicit motion estimation, we seek a super-resolution algorithm of similar nature that will allow processing sequences with general motion patterns. In this chapter we base our solution on the Non-Local-Means (NLM) algorithm. We show how this denoising method is generalized to become a relatively simple super-resolution algorithm with no explicit motion estimation. Results on several test movies show that the proposed method is very successful in providing super-resolution on general sequences.
3.1 Introduction

Super-resolution reconstruction proposes a fusion of several low quality images into one higher quality result with better optical resolution. This is an Inverse Problem that combines denoising, deblurring, and scaling-up tasks, aiming to recover a high quality signal from degraded versions of it. Figure 3.1 presents the process that explains how a low-resolution image sequence \( y_t \) is related to an original higher resolution movie \( x_t \). During the imaging, the scene may become blurred due to atmospheric, lens, or sensors’ effects. The blur is denoted by \( H \), assumed for simplicity to be linear space and time invariant. Similarly, the loss of spatial resolution due to the sensor array sampling is modeled by the fixed decimation operator \( D \), representing the resolution factor \( s \) between the original sequence, and the measured one. White Gaussian iid noise is assumed to be added to the measurements, both in order to refer to actual noise in imaging systems, as well as for accommodating model mismatches.

![Figure 3.1: The imaging process to be reversed by super resolution.](image)

The super resolution goal is the recovery of \( x_t \) from the input set of images \( y_t \), reversing the above process. Such reconstruction relies on motion in the scene to recover details that are finer than the sampling grid. Figure 3.2 demonstrates how small details can be recovered when the motion between the images in the sequence is known with a high degree of accuracy. The top row in the figure is the input sequence. The middle row is the up-scaled version of each image (unknown values are set to a background color), shifted by the known translation between the current image and the first (reference) image. The bottom row shows the construction of the super resolution image, from left to right. Initially, the first image is placed on the grid. Then, every new image in the sequence is placed on the same grid, with a displacement reflecting the motion it underwent. The merger of all images represents the outcome of the super resolution algorithm. We note
that this description of the mechanics of super-resolution is somewhat simplistic; In most
cases, one cannot assume the translations to be exact multiples of the high-resolution
pixel sizes. This makes the estimation of accurate motion parameters and the merger of
all images much more complex than described here.

![Figure 3.2: Super resolving an image using low resolution inputs with known translations. Reconstruction proceeds from left to right. Top: input images; Middle: Corresponding up-scaled images, shifted using known translations; and Bottom: Accumulated reconstruction by adding the current low resolution image to the output canvas.]

While the above-described method is somewhat simplistic, it is a faithful description of
the foundations for all classic super resolution algorithms. The first step of such algorithms
is an estimation of the motion in the sequence, followed by a fusion of the inputs according
to these motion vectors. A wide variety of super resolution algorithms have been developed
in the past 2 decades; we refer to [70, 76, 71, 95, 125, 111, 93, 69, 46, 113, 141, 88, 47, 121,
4, 7, 91, 52, 80, 53, 54, 64, 24, 127, 23, 114] as representatives of this vast literature.

In the currently available super-resolution algorithms, only global motion estimation
(e.g. translation or affine global warp) is accurate enough to lead to a successful recon-
struction of a super-resolved image. This is very limiting, as most actual scenes contain
motion that is local in its nature (e.g. a person talking). Obtaining highly accurate local
motion estimation, known as optical flow, is a very difficult task, and particularly so in the
presence of aliasing and noise. When inaccurately estimated motion is used within one of
the existing reconstruction algorithms, it often leads to disturbing artifacts that cause the
output to be inferior, even when compared to the given measurements. This discussion
leads to the commonly agreed unavoidable conclusion that general content movies are not
likely to be handled well by classical super-resolution techniques.

This severe restriction leads us to seek a different approach to super-resolution. Can
such an algorithm be proposed with no explicit motion estimation? Our starting point
for this quest (after a super-resolution algorithm that is able to process sequences with a
general motion pattern) is the video denoising application, where several recent contribu-
tions demonstrate state-of-the-art results with algorithms that avoid motion estimation
Among these, we choose to take a closer look at the Non-Local Means (NLM) algorithm, with the aim to generalize it to perform super-resolution reconstruction.

The NLM is the weakest among the recent motion-estimation-free video denoising algorithms, and yet, it is also the simplest. As such, it stands as a good candidate for generalization. The NLM is posed originally in [15] as a single image denoising method, generalizing the well-known bilateral filter [122, 41]. Denoising is obtained by replacing every pixel with a weighted average of its neighborhood. The weights for this computation are evaluated by using block-matching fit between image patches centered around the center pixel to be filtered, and the neighbor pixels to be averaged. Recent work has shown how this method can be used for video denoising by extending the very same technique to 3D neighborhoods [17]. An improvement of this technique, considering varying size neighborhoods is suggested in [13], so as to trade bias versus variance in an attempt to get the best mean-squared-error (MSE).

The NLM was proposed intuitively in [15, 17] and thus it is natural to try to extend it to perform super-resolution using a similar intuition. This intuition leads to independent up-scaling of each image in the sequence using a smart interpolation method, followed by NLM processing. However, extensive experiments indicate that this intuitive method does not provide super-resolution results. For this reason, a more profound understanding of the NLM filter is required for its successful generalization to super-resolution.

In order to gain a better understanding of the NLM, we propose redefining it as an energy minimization task. We show that the novel penalty term we propose indeed leads when minimized to the NLM. We then carefully extend the penalty function to the super-resolution problem. We show how a tractable algorithm emerges from the minimization of this penalty, leading to a local, patch-based, super-resolution process with no explicit motion estimation. Empirical tests of the derived algorithm on actual sequences with general motion patterns are then presented, thus demonstrating the capabilities of the derived algorithm.

The structure of the chapter is as follows. Section 3.2 describes the NLM denoising filter, as posed in [15]. This section can be skipped by readers who are familiar with the NLM. Section 3.3 introduces an energy function to be minimized for getting a denoising effect for a single image; We show that this minimization leads to a family of image denoising algorithms, NLM included as a special case. We also provide a simpler penalty function addressing the same goal, which will be effectively used in the later part of
the chapter. Section 3.4 proposes a generalization of the introduced energy function to cope with resolution changes, thereby enabling super-resolution reconstruction. In this section we also derive the eventual super-resolution algorithm we propose, and discuss its numerical structure. Section 3.5 shows results on sequences with general motion, demonstrating the successful recovery of high-frequencies. We conclude in Section 3.6, outlining the key contribution of this work, and describing several directions for further research.

3.2 The Bilateral and the NLM Denoising Filters

We begin our journey with a description of the bilateral and the NLM filters, as the development that follows relies on their structure. The description given in this section is faithful to the one found in [15, 122]. The bilateral and the NLM filters are two very successful image denoising filters. While not the very best in denoising performance, these methods are very simple to understand and implement, and this makes them a good starting point for our needs.

Both the bilateral and the NLM filters are based on the assumption that image content is likely to repeat itself within some neighborhood. Therefore, denoising each pixel is done by averaging all pixels in its neighborhood. This averaging is not done in a blind and uniform way, however. Instead, each of the pixels in the relevant neighborhood is assigned a weight, that reflects the probability that this pixel and the pixel to be denoised had the same value, prior to the additive noise degradation. A formula describing these filters looks like

$$\hat{x}[k,l] = \frac{\sum_{(i,j) \in N(k,l)} w[k,l,i,j] y[i,j]}{\sum_{(i,j) \in N(k,l)} w[k,l,i,j]}$$

(3.1)

where $N(k,l)$ stands for the neighborhood of the pixel $(k,l)$, and the term $w[k,l,i,j]$ is the weight for the $(i,j)$-th neighbor pixel. The input pixels are $y(k,l)$, and the output result in that location is $x(k,l)$.

The two filters differ in the method by which the weights are computed. The weights for the bilateral filter are computed based both on radiometric (gray-level) proximity and

---

1As we shall see next, in this framework the coefficients $w[k,l,i,j]$ are all restricted to be positive. This is a shortcoming, which can be overcome by extending the framework to higher order – see [119].
geometric proximity between the pixels, namely

\[ w_{BL}[k, l, i, j] = \exp \left\{ \frac{-\left( y[k, l] - y[i, j] \right)^2}{2\sigma_r^2} \right\} \cdot f \left( \sqrt{(k - i)^2 + (l - j)^2} \right). \tag{3.2} \]

The function \( f \) takes the geometric distance into account, and as such, it is monotonically non-increasing. It may take many forms, such as a Gaussian, a box function, a constant, and more. The parameter \( \sigma_r \) controls the effect of the grey-level difference between the two pixels. This way, when the two pixels that are markedly different, the weight is very small, implying that this neighbor is not to be trusted in the averaging.

The radiometric part in the weights of the NLM is computed slightly differently, by computing the Euclidean distance between two image patches centered around these two involved pixels. Defining \( \hat{R}_{k,l} \) as an operator that extracts a patch of a fixed and predetermined size (say \( q \times q \) pixels) from an image, the expression \( \hat{R}_{k,l}y \) (\( y \) is represented as a vector by lexicographic ordering) results with a vector of length \( q^2 \) being the extracted patch. Thus, the NLM weights are given by

\[ w_{NLM}[k, l, i, j] = \exp \left\{ \frac{-\left\| \hat{R}_{k,l}y - \hat{R}_{i,j}y \right\|^2}{2\sigma_r^2} \right\} \cdot f \left( \sqrt{(k - i)^2 + (l - j)^2} \right). \tag{3.3} \]

Obviously, setting \( \hat{R}_{k,l} \) to extract only a single pixel, the bilateral filter emerges as a special case of the NLM algorithm.

We note that there are various other ways to choose the weights in Equation (3.1), and the above separable choice of the weights (product of radiometric and Euclidean distance terms) is only one choice. For example, the steering kernel may provide an interesting alternative, taking into account the correlation between the pixel positions and their value [119]. Nevertheless, in this chapter we shall restrict our choice of weights to those used by the NLM.

### 3.3 NLM Via Energy Minimization

Both the bilateral and the NLM filters described above were presented intuitively as algorithmic formulas, as in Equation (3.1). We claim that both these filters can be derived by minimizing a properly defined penalty function. Following the rationale and steps taken in [41, 60], we present such a penalty function, and show how these algorithms emerge from it. This will prove valuable when taking the next step of generalizing these methods to a super-resolution reconstruction algorithm, as will be shown in Section 3.4. Section
3.3.1 and 3.3.3 present two possible and novel penalty functions for denoising, and derive from both the NLM algorithm and some variations of it. The readers interested in the super-resolution portion of this work can start their reading in Section 3.3.3.

### 3.3.1 The Penalty Function

The penalty function we start with reflects two forces: (i) We desire a proximity between the reconstructed and the input images - this is the classic likelihood term; and (ii) We would like each patch in the resulting image to resemble other patches in its vicinity. However, we do not expect such a fit for every pair, and thus we introduce weights to designate which of these pairs are to behave alike. Putting these two forces together with proper weighting\(^2\), we propose a Maximum A-posteriori Probability (MAP) penalty of the form

\[
\epsilon^2 (x) = \lambda \cdot \|x - y\|^2_2 + \frac{1}{4} \cdot \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k,l,i,j] \cdot \|R_{k,l}x - R_{i,j}x\|^2_2.
\]  

(3.4)

The first term is the log-likelihood function for a white and Gaussian noise. The second term stands for a prior, representing the (minus) log of the probability of an image \(x\) to exist. The weights in the above expression, \(w[k,l,i,j]\), are assigning a confidence that the patches around \(x[k,l]\) and \(x[i,j]\) are to be close to each other. Computing these weights can be done in a number of ways, one of which is using Equation (3.3) and using \(y\) instead of the unknown \(x\). In order to keep the discussion simple, from this point on we shall assume that the weights \(w[k,l,i,j]\) are the NLM ones. It is important to note that the patch extraction operator \(\hat{R}_{kl}\) used for computing the weights (as in Equation (3.3)) and the operator \(R_{kl}\) used in the penalty term are generally of different sizes.

The notation \(\Omega\) stands for the support of the entire image. Thus, the second term sweeps through each and every pixel \((k,l)\) in the image, and for each we require a proximity to surrounding patches in its neighborhood.

### 3.3.2 Derivation of the NLM Filter

Assuming the weights are pre-determined and considered as constants, we can minimize this penalty term with respect to \(x\) by zeroing its derivative,

\(^2\)The reason for the factor \(\frac{1}{4}\) in the second term will be made clear shortly.
Using these two assumptions, we get the following two equalities:

\[
0 = \frac{d^2(x)}{dx} = \lambda (x - y) + \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] (R_{k,l} - R_{i,j})^T (R_{k,l} - R_{i,j}) x. \tag{3.5}
\]

In order to simplify this equation, we open the brackets,

\[
\frac{d^2(x)}{dx} = 0 = \lambda (x - y) + \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{k,l} x - \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{i,j}^T R_{i,j} x
- \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{i,j} x + \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{i,j}^T R_{i,j} x.
\]

We proceed by invoking two assumptions: (i) The neighborhood is symmetric, i.e. if \((i, j) \in N(k,l)\), then \((k,l) \in N(i,j)\) is also true; and (ii) The weights are symmetric, i.e., \(w[k, l, i, j] = w[i, j, k, l]\). Both these assumptions are natural – typical neighborhood definitions satisfy the first condition, and the weights of the NLM satisfy the second one.

Using these two assumptions, we get the following two equalities:

\[
\sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{i,j}^T R_{k,l} x = \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{i,j} x
\]

\[
\sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{k,l} x = \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{i,j}^T R_{i,j} x.
\]

A formal proof (Theorem 1) for these equalities is given in Appendix 3.A. Using these, Equation (3.6) simplifies and becomes

\[
0 = \lambda (x - y) + \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{k,l} x - \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{i,j} x - \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{i,j}^T R_{i,j} x.
\]

As can be seen, the choice 1/4 in the original definition led to a simpler final outcome.

While solving the above equation directly is possible in principle, it requires an inversion of a very large matrix. Instead, we adopt an iterative approach based on the fixed-point strategy [81, 10]. Denoting \(x^{n-1}\) the outcome of the previous iteration, and \(x^n\) the desired outcome of the current iteration, we rewrite Equation (3.7) with assignments of iteration stage per each instance of the unknown \(x\). The equation we propose is

\[
0 = \lambda (x^n - y) + \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{k,l} x^n - \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{i,j} x^{n-1}, \tag{3.8}
\]
which leads to the relation

\[
\left( \lambda I + \sum_{(k,l) \in \Omega} \left[ \sum_{(i,j) \in N(k,l)} w[k,l,i,j] R_{k,l}^T R_{k,l} \right] \right) x^n = \lambda y + \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k,l,i,j] R_{k,l}^T R_{i,j} x^{n-1}.
\] (3.9)

Notice that the term \( \sum_{(i,j) \in N(k,l)} w[k,l,i,j] \) generates a scalar, being a function of \((k,l)\) – we shall denote this as \( \bar{w}[k,l] \).

In the obtained equation, the right-hand-side (RHS) creates an image by manipulating image patches: for each location \((k,l)\) in the image, we copy surrounding neighboring patches in locations \((i,j)\) to the center position \((k,l)\), multiplied by the weights \(w[k,l,i,j]\). Once built, this image is added to \(y\) with a proper weight \(\lambda\).

The matrix multiplying \(x^n\) on the left-hand-side is a diagonal positive definite matrix (see Appendix 3.A). This matrix’s only task is normalization of the weighted average that took place on the RHS. As this matrix is invertible, the new solution is obtained by

\[
x^n = \left( \lambda I + \sum_{(k,l) \in \Omega} \bar{w}[k,l] R_{k,l}^T R_{k,l} \right)^{-1} \left( \lambda y + \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k,l,i,j] R_{k,l}^T R_{i,j} x^{n-1} \right).
\] (3.10)

When using the fixed-point method, as we did above, every appearance of the unknown in the equation is assigned with an iteration number. Among the many possible assignments, one should seek one that satisfies two important conditions: (i) The computation of \(x^n\) from \(x^{n-1}\) should be easy; and (ii) The obtained iterative formula should lead to convergence. As for the first requirement, we indeed have an assignment that leads to a simple iterative step. Convergence of the above algorithm is guaranteed if the overall operator multiplying \(x^{n-1}\) is convergent, i.e.,

\[
\rho \left\{ \left( \lambda I + \sum_{(k,l) \in \Omega} \bar{w}[k,l] R_{k,l}^T R_{k,l} \right)^{-1} \left( \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k,l,i,j] R_{k,l}^T R_{i,j} \right) \right\} < 1,
\]

where \(\rho(A)\) is the spectral radius of \(A\). It is easily seen that for sufficiently large \(\lambda\), this condition is met. Nevertheless, we do not worry about convergence, as we will be using the above for one iteration only, with the initialization of \(x^0 = y\). This means that the output for the denoising process is \(\hat{x} = x^1\), obtained by
\[
\hat{x} = \left( \lambda I + \sum_{(k,l) \in \Omega} \bar{w}[k,l]R_{k,l}^T R_{k,l} \right)^{-1} \cdot \left( \lambda I + \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k,l,i,j]R_{k,l}^T R_{i,j} \right) y.
\]

(3.11)

The above computation is done just as described above, with the obvious substitution of \( x^0 = y \). This process is quite reminiscent of the way NLM and the bilateral filters operate, and yet it is different. The obtained algorithm is a more general and more powerful denoising algorithm than NLM.

In order to see how the NLM emerges from this formulation as a special case, we shall assume further that the patch extraction operation we use, \( R_{i,j} \), extracts a single pixel in location \((i,j)\). This change means that \( R_{i,j} y \) is in fact the single pixel \( y[i,j] \). Thus, in this case we have

\[
\hat{x} = \left( \lambda I + \sum_{(k,l) \in \Omega} \bar{w}[k,l]R_{k,l}^T R_{k,l} \right)^{-1} \cdot \left( \lambda y + \sum_{(k,l) \in \Omega} R_{k,l}^T \sum_{(i,j) \in N(k,l)} w[k,l,i,j]y[i,j] \right).
\]

(3.12)

The term \( R_{k,l}^T x \) implies that a zero image is generated, and the value \( x \) is inserted to location \((k,l)\). Using this property in Equation (3.11), we obtain a pixel-wise denoising formula

\[
\hat{x}[k,l] = \frac{\lambda y[k,l] + \sum_{(i,j) \in N(k,l)} w[k,l,i,j]y[i,j]}{\lambda + \sum_{(i,j) \in N(k,l)} w[k,l,i,j]}.
\]

(3.13)

This formula seems familiar, as it is effectively the same as the one in Equation (3.4), with a slight modification due to the additional \( \lambda \). This means that under the simplifying assumption we made about \( R_{i,j} \), the first iteration of the developed algorithm reduces to NLM. A natural question to ask is: Does the above formula still stands for a patch-based algorithm? The answer is positive, as the weights may be computed using patches, as the NLM does. Thus, the next natural question to ask is: Why have we used a patch extraction operation in the definition of the penalty in Equation (3.4)? The answer to this is the generalization that follows next for the super-resolution case.

### 3.3.3 Bayesian Versus Proximity – An Alternative Path

The penalty term in Equation (3.4) was formed like a MAP estimator, having a likelihood term that ties the measurements to the unknown, and a regularization term, posing a
prior on the desired outcome,
\[ \epsilon^2(x) = \frac{\lambda}{2} \cdot ||x - y||_2^2 + \frac{1}{4} \cdot \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] \cdot \|R_{k,l}x - R_{i,j}y\|_2^2. \]

In the previous section we developed a general denoising scheme based on the above penalty, showing that the NLM and the bilateral filters arise from it as special cases. Recall that one of the last steps in the derivation was the use of a single iteration and initialization with \( x^0 = y \). This basically means that the prior term requires pixels in the output image \( x_1 \) to have similar grey values to pixels in the corresponding neighborhood in \( y \), provided they have similar surrounding. This idea can be put directly into the penalty term, giving
\[ \eta^2(x) = \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] \cdot \|R_{k,l}x - R_{i,j}y\|_2^2. \] (3.14)

Targeting now the minimization of this penalty term, we null the derivative of this function with respect to \( x \), getting the equation
\[ \frac{d\eta^2(x)}{dx} = 0 = \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T (R_{k,l}x - R_{i,j}y). \] (3.15)

This time we do not need the fixed-point strategy, as a simple solution is easily derived, leading to
\[ \hat{x} = \left( \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{k,l} \right)^{-1} \cdot \left( \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^T R_{i,j}y \right). \] \[ \] (3.16)

Notice the resemblance between this formula and the one obtained by a different route in Equation (3.11). In fact, for \( \lambda = 0 \), the two formulas are the same. Furthermore, following the same assumptions and steps that led us from Equation (3.11) to Equation (3.13), it is clear that pixel-wise, the above aligns perfectly with the NLM filter, as written in Equation (3.1), namely,
\[ \hat{x}[k, l] = \frac{\sum_{(i,j) \in N(k,l)} w[k, l, i, j] y[i, j]}{\sum_{(i,j) \in N(k,l)} w[k, l, i, j]}. \]
3.3.4 Introducing the Temporal Domain

We shall use hereafter the penalty function in Equation (3.14) to derive further algorithms, due to its simplicity, and the fact that it leads directly to NLM. Before we turn to super-resolution, we must introduce the temporal axis into the penalty function, so as to process a sequence of images and not just a single one, as done so far. The following small changes in (3.14) lead to such a treatment:

\[ \eta^2_T(x) = \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{t \in [1..T]} \sum_{(i,j) \in N(k,l)} w[k,l,i,j,t] \cdot \| R_{k,l} x - R_{i,j} y_t \|^2_2. \]  

(3.17)

The above expression makes use of the input sequence \( \{ y_t \}_{t=1}^T \), summing over all these images. The term \( R_{i,j} y_t \) remains a 2D patch, but one that is extracted at location \((i, j)\) from the image at time \(t\). The image \( x \) remains a single image, representing the desired output image to be created. Let us assume that this image aims to become a denoised version of \( y_{t_0} \). This fact is used only within the computation of the weights \( w[k,l,i,j,t] \), which matches a reference patch \( R_{k,l} y_{t_0} \) with the candidate ones \( R_{i,j} y_t \), both 2D. The temporal distance, \( t - t_0 \), can also influence the weight \( w \), just as spatial distances \( k - i \) and \( l - j \) do in Equation (3.3).

Assuming that the patches extracted by the operator \( R \) are of size of one pixel\(^3\), using the same algebraic steps as shown in the previous sub-section we get a closed-form formula for the computation of \( x \),

\[ \hat{x}[k,l] = \frac{\sum_{t \in [1..T]} \sum_{(i,j) \in N(k,l)} w[k,l,i,j,t] y_t[i,j]}{\sum_{t \in [1..T]} \sum_{(i,j) \in N(k,l)} w[k,l,i,j,t]}. \]  

(3.18)

This is a generalization of the NLM to handle video sequences, and was shown to be an effective algorithm in [17].

Let us explain this formula in words, as a similar structure will serve us in the next Section quite similarly. Each output pixel is computed as a weighted average of pixels in its 3D neighborhood in the input sequence \( y \). We would like pixels that originally had the same grey level to have high weights, and the weights are computed in a way that reflects this.

By taking a slightly different perspective, we can regard the weight \( w[k,l,i,j,t] \) as reflecting the (relative) probability that the pixel \((k,l)\) in the image to be denoised has gone to (or come from) the pixel \((i,j)\) in the image \( y_t \). This is a very basic motion

---

\(^3\)We remind the reader that the patch size related to the weights-computation is different than the one in the penalty function itself in general.
estimation approach for each pixel. However, since for each \((k,l)\), there may be several non-zero values, this implies that each pixel may be assigned with \textit{several} motion vectors, and not just one, as in classical motion estimation methods. Indeed, the above expression suggests that all motion vectors are allowed, and considered according to the probability that they actually took place based on the patch matching they exhibit. This way, one could consider the above as a fuzzy motion estimation process, rather than an explicit one.

Figure 3.3 presents the concept of fuzzy motion estimation. A specific pixel on the scene at time \(t\) (the left image), has several equally probable destinations in the surrounding images (including the same image at time \(t\)). This can be done for any pixel, some having many probable destinations, and some only a few.

An important benefit to the above fuzziness, is the ability to get a stronger denoising effect. Whereas exact motion estimation leads to a correspondence between the reference patch and a single match in every other image, the NLM approach may find several good matches due to spatial (and not just temporal) redundancy. As seen in Figure 3.3, several very good matches can be found, and thus, when averaged, lead to a more effective noise suppression due to the independence between the noise in each of those patches.

Generalization of the NLM approach to video denoising was shown to be very effective \([17, 13, 29]\), leading to state-of-the-art results, while leaning strongly on the fuzziness of the estimated motion. These methods’ successes encourage us to seek ways to exercise the same fuzzy motion estimation concept in other video processing tasks, where it is commonly assumed that explicit motion estimation is a mandatory step. One such task is super-resolution. In the rest of this chapter we focus on a super-resolution scheme that relies on fuzzy, rather than explicit, motion estimation, generalizing the NLM as presented above.

### 3.4 Super Resolution With No Explicit Motion Estimation

#### 3.4.1 The Super-Resolution Penalty Function

The main advantage of the fuzzy motion approach is its flexibility, allowing it to handle complex scenes. Whereas classic super resolution methods make many limiting assumptions regarding the motion in the scene, this is not the case with fuzzy motion estimation, which can handle local motions, occlusions, and so on. Our goal in this section is to exploit this flexibility to perform super-resolution on sequences with arbitrary motion patterns,
thus avoiding the global motion limitation.

Developing a super resolution algorithm will again begin with writing a proper penalty term and minimizing it. The input to this algorithm is the set of low resolution images $y_t$. For clarity of description, we target one high resolution image $X$ at a time (instead of working on the entire sequence at once). The usage of a capital letter to denote the output image, and lower-case letters to denote the input sequence, serves to remind us that they are indeed of different scales.

Since we want to exploit the insight gained in previous sections, we aim at defining a penalty function that is similar to that written for the video denoising problem in Equation (3.17). However, the target image $X$ and the input images $y_t$ are not of the same scale, forcing a change to the penalty term. This change should reflect the fact that $X$ undergoes blurring and decimation steps, in order to account for the different scales. This leads us to the following preliminary (and not final!) proposal:

$$
\eta_{SR}(X) = \frac{1}{2} \sum_{(k,l) \in \Omega} \sum_{t \in [1..T]} \sum_{(i,j) \in \mathcal{N}(k,l)} w[k, l, i, j, t] \|R_{k,l}DX - R_{i,j}y_t\|_2^2 + \lambda \cdot TV(X).
$$

Introduce into the penalty term are two operators we have met in the introduction: $H$ – the blurring operator, and $D$ – the decimation operator. Applying these operators on $X$ simulates the imaging process, bringing $X$ to the same resolution as the input sequence $y_t$. The additional TV (Total Variation) expression comes to regularize the deblurring that should take place in this expression [108], forcing piece-wise smoothness of the desired image, by accumulating the norms of the gradients with $L_1$ norm.
Taking a close look at the penalty term we have just written reveals that it cannot provide super resolution reconstruction. By denoting \( x = DHX \), it is clearly seen the above is identical to the video denoising penalty term in equation (3.17) for \( x \). This means that minimizing this penalty term effectively divides the problem into two parts – the first performs denoising of the input sequence, and the second that interpolates and de-blurs the resulting image.

The reason this penalty term is not able to perform super resolution is the fact that the operator \( R_{k,l} \) “sees” only \( \frac{1}{s^2} \) of the pixels in the image (\( s \) is the decimation factor) because of the decimation that precedes it. In order to solve this problem, we reverse the order of the patch extraction and decimation, so that a patch is first extracted from \( HX \), and then decimated to the same resolution level as \( y_t \) to be compared to a patch from it. This change requires a change in the operator \( D \) as well, as we no longer use the constant decimation grid of the image. Instead, we introduce the notation \( D_p \) – a patch decimation operation that ensures that the center pixel of the patch is on the decimation grid.

Since there are two different scales (and sizes) of patches used, we also must differentiate between the patch extraction operators. To this end, we denote by \( R_{k,l}^H \) and \( R_{i,j}^L \) the high- and low-resolution patch extraction operators respectively. Their sizes are linked through the value of the scaling parameter \( s \): whereas \( R_{i,j}^L \) extracts a patch of size \( q \times q \) pixels (arbitrary), \( R_{k,l}^H \) extracts a patch of size \( p \times p \) pixels, where \( p = s(q - 1) + 1 \). The relation between these sizes is further explained in Figure 3.4. This transforms the penalty term in Equation (3.19) into

\[
\eta_{SR}(X) = \sum_{(k,l)\in\Omega} \sum_{t\in[1..T]} \sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \| D_p R_{i,j}^H HX - R_{i,j}^L y_t \|_2^2 + \lambda \cdot TV(X) \tag{3.20}
\]

Figure 3.5 presents a block diagram to explain this penalty. As can be seen, different size patches are extracted from \( x \) and \( y_t \), and brought to the same grounds by a decimation operation.

**Note:** The term \((i,j) \in N(k,l)\) is no longer applicable as before, since now \((i,j)\) and \((k,l)\) are pixels on different resolution grids. Therefore, we introduce a new notation \( N^L \) referring to the equivalent low-resolution neighborhood of \((k,l)\), i.e \((i,j) \in N^L(k,l)\) is short for the full notation \((i,j)\) s.t. \((si,sj) \in N(k,l)\).

The penalty term we have defined above is the one we shall focus on hereafter. Next we show how to use this penalty term as the basis for a practical and simple super resolution
Figure 3.4: The relation between the patch sizes \( q \) and \( p = (q - 1)s + 1 \) demonstrated on specific examples: \( [q, s] = [3, 1], [3, 2] \), and \( [3, 3] \).

Figure 3.5: A description of the expression \( D_p R_{i,j}^H H X - R_{i,j}^L y_t \) in Equation (3.20).
algorithm. The first step we take is a separation of the deblurring from the fusion of the images. Using the substitution \( Z = HX \), the first problem we have is the estimation of \( Z \) from the measurements \( \{y_t\}_{t=1}^T \). This is done by minimizing the energy function

\[
\eta_{SR}^A(Z) = \sum_{(k,l) \in \Omega} \sum_{t \in [1..T]} \sum_{(i,j) \in N^c(k,l)} w[k, l, i, j, t] \| D_p R_{k,l}^H Z - R_{i,j}^L y_t \|_2^2.
\]  

(3.21)

Once found, we use the found \( \hat{Z} \) to estimate \( X \) by minimizing

\[
\eta_{SR}^B(X) = \| \hat{Z} - HX \|_2^2 + \lambda TV(X).
\]  

(3.22)

The second part is a classic simple deblurring problem. This is of course an under-determined problem (since \( H \) is usually singular), and needs regularization. We will not discuss how to solve it here, see [11, 55, 105, 75, 90, 12] for a background view of this field, along with several state-of-the-art deblurring methods.

The idea of breaking the super-resolution task into two parts – fusing the inputs and then deblurring – has been suggested previously in [47, 52, 53, 54]. In the case of pure translational motion, and when both problems are treated as maximum-likelihood (which is not done here), such a separation is equivalent to the joint solution. It is important to note that in the derivation proposed above, the separation is not optimal, and leads to inferior results. However, the separation allows for a much simpler algorithm (both conceptually and implementation-wise), so the sub-optimality is the price to be paid for this simplicity.

The fusion stage in Equation (3.21) seems to lack regularization, and as such, one may question its stability. Stability is gained through the weights – those should be computed in such a way that ensures that every pixel has at least a few “quality” (assigned a meaningful weight) destinations. Furthermore, by also assigning a non-zero weight to the original value of the pixel, further stability is obtained, guaranteeing that in case no appropriate patches are found, the result will default to the initial estimate.

We shall leave the deblurring aside, and focus hereafter on the fusion of the measurements. This will allow us to simplify the description of the algorithm. The next step is deriving the penalty function in Equation (3.20) with respect to \( Z \), and finding the zero intersection:

\[
0 = \frac{d \eta_{SR}^A(Z)}{dZ} = 2 \sum_{(k,l) \in \Omega} \sum_{t \in [1..T]} \sum_{(i,j) \in N^c(k,l)} w[k, l, i, j, t] (D_p R_{k,l}^H)^T (D_p R_{k,l}^H Z - R_{i,j}^L y_t).
\]  

(3.23)
This leads to the solution
\[
\hat{Z} = \left[ \sum_{(k,l)\in \Omega} \sum_{t\in [1..T]} \sum_{(i,j)\in N^T(k,l)} w[k,l,i,j,t] \left( D_p R^H_{k,l} \right)^T \left( D_p R^H_{k,l} \right) \right]^{-1} \cdot \left[ \sum_{(k,l)\in \Omega} \sum_{t\in [1..T]} \sum_{(i,j)\in N^T(k,l)} w[k,l,i,j,t] \left( D_p R^H_{k,l} \right)^T R^L_{i,j} y_t \right],
\]
which can be further simplified, by defining
\[
\bar{w}[k,l] = \sum_{t\in [1..T]} \sum_{(i,j)\in N^T(k,l)} w[k,l,i,j,t],
\]
leading to
\[
\hat{Z} = \left[ \sum_{(k,l)\in \Omega} \bar{w}[k,l] \left( D_p R^H_{k,l} \right)^T \left( D_p R^H_{k,l} \right) \right]^{-1} \cdot \left[ \sum_{(k,l)\in \Omega} \left( D_p R^H_{k,l} \right)^T \left( \sum_{t\in [1..T]} \sum_{(i,j)\in N^T(k,l)} w[k,l,i,j,t] R^L_{i,j} y_t \right) \right].
\]
The right term that involves the measurements \( \{ y_t \}_{t=1}^T \) performs a series of simple operations that include: (i) Extraction of patches from the measurements; (ii) Zero-padding interpolation of these patches (done by \( D_p^T \)); (iii) An accumulation of the resulting patch with a proper weight at the destination location.

Following the same rationale as the one practiced for Equation (3.9), the matrix to be inverted in the above expression is a diagonal one and positive semi-definite, normalizing the accumulation in each pixel. The term \( \left( D_p R^H_{k,l} \right)^T \left( D_p R^H_{k,l} \right) \) extracts a patch from location \((k,l)\), scales it down and then up again by zero padding, and finally puts it back into the same original location. Thus, this is a diagonal matrix with 1-es for the pixels surviving this path, and zeros elsewhere. The matrix to be inverted is a weighted sum of such diagonal matrices, and thus it is diagonal as well. If every pixel \((k,l)\) gets some accumulation, this matrix is strictly positive definite, and its inversion is permitted.

### 3.4.2 A Simplified Numerical Algorithm

We again follow the path of Section 3.3, and propose a special and simplified case where the patch extraction operator \( R^L_{i,j} \) extracts only one pixel. This means that \( R^H_{k,l} \) extracts a patch of size \( s \times s \) pixels, to become one pixel after the decimation operation \( D_p^4 \). This

\[\text{Note that this change of patch-sizes applies only to the penalty term itself, and does not effect the patch size for the weights computation.}\]
simplifies the penalty function in Equation (3.21), since $D_p R_{k,l} Z = Z[k,l]$ and $R_{i,j} y_t = y_t[i,j]$, leading to

$$\eta_{SR}^A(Z) = \sum_{(k,l) \in \Omega} \sum_{t \in [1..T]} \sum_{(i,j) \in N^R_{k,l}} w[k,l,i,j,t] (Z[k,l] - y_t[i,j])^2.$$  \hspace{1cm} (3.27)

This functional is separable, handling every pixel in the target image $Z$ separately. This implies that an independent penalty can be written for every destination pixel,

$$\eta_{SR}^A(Z[k,l]) = \sum_{t \in [1..T]} \sum_{(i,j) \in N^R_{k,l}} w[k,l,i,j,t] (Z[k,l] - y_t[i,j])^2.$$ \hspace{1cm} (3.28)

leading to a closed-form solution

$$\hat{Z}[k,l] = \frac{\sum_{t \in [1..T]} \sum_{(i,j) \in N^R_{k,l}} w[k,l,i,j,t] y_t[i,j]}{\sum_{t \in [1..T]} \sum_{(i,j) \in N^R_{k,l}} w[k,l,i,j,t]}.$$ \hspace{1cm} (3.29)

In order for this solution to exist, it is enough that for each pixel $(k,l)$ there exists at least one none-zero weight $w[k,l,i,j,t]$.

Notice that the above formula looks exactly the same as the one used for video denoising, posed in Equation (3.18). Are the two equivalent? The answer is negative, due to two important reasons. First, there is a gap between the definitions of the neighborhoods $N(k,l)$ and $N^R_{k,l}$. Whereas for the video denoising this neighborhood is defined simply as the set of all neighbors for the central location $(k,l)$, the neighborhood referred to in Equation (3.29) considers locations $(i,j)$ that after scaling up by $s$ are neighbors of $(k,l)$.

The second difference between (3.18) and (3.29) refers to the weights to be used. As described earlier, we want the weight $w[k,l,i,j,t]$ to reflect the probability that the pixel $Z[k,l]$ and the pixel $y_t[i,j]$ originated from the same place. This computation will be based on the similarity of the area around both pixels, in the same manner the weights for the NLM are computed (since they serve a similar purpose), as described in Equation (3.3). The function $f$ that takes into account the geometric distance between the patches is set to be constant, thereby giving no preference to nearby patches over distant ones. This allows more robustness to various motion patterns, including patterns that contain large motions. However, in some cases, it may be beneficial to assign higher weights to patches closer temporally or geometrically.

However, for matching these two areas, we need to address the fact that the neighborhoods around the pixels in $Z$ and in $y_t$ are not of the same scale. We can address this in one of two ways: (i) Down-sample the neighborhood in the high-resolution image and...
bring it down to the low resolution grid; or (ii) Up-sample the low-resolution neighborhood to match the high-resolution one. In our tests we have found that these two options perform similarly well, with a small advantage to the second (and more complex) one.

As in the denoising case, the computation of the weights requires the use of the unknown $\mathbf{Z}$. Instead, the weights are computed by using an estimated version of $\mathbf{Z}$ being a scaled-up version of the reference frame we aim to super-resolve. This scale-up is done using a conventional image interpolation algorithm such as bilinear, bicubic or the Lanczos method [133, 124].

The interpolated images are only crude estimates of the desired outcome, and therefore the weights computed by relying on these estimates are also somewhat crude. Since after one iteration of the algorithm we obtain a better estimate, it is possible to use these new estimates for re-computing the weights, and computing a better still estimate of the desired outcome. This process may be iterated several times, although in our simulations we use only two such iterations (i.e. one iteration that relies on the interpolated frames for computing the weights, and a second iteration that relies on the results of the first iteration). This means that all frames are first processed using the interpolated frames for the weights computation, and only then is the second iteration applied. If throughput constraints do not allow this, it is possible to compute the weights in the second iteration while still using the interpolated versions of all frames, and a new estimate is only used for the currently processed frame. This requires some minor modifications to the weights computation.

3.4.3 The Proposed Algorithms – A Summary

We now summarize the two algorithms previously described. First, we describe in Tables 3.1 and 3.2 the general super-resolution algorithms that were developed in this section. The simpler version that uses low-resolution patches of one pixel is obtained by using the proper assignments for $\mathbf{R}_k^H$ and $\mathbf{R}_i^L$ (while not changing $\mathbf{R}$).

The complexity of these two algorithms is essentially the same as that of their NLM counterparts, with the addition of a deblurring process, which can be considered negligible. The core of the algorithm, which also requires most of the computations, is computing the weights. Considering a nominal case with a search area of $31 \times 31$ low-resolution pixels in the spatial domain, and 15 images in each in the temporal axis, we have $\approx 14,000$ pixels in this spatio-temporal window. For each pixel in the search area, the block difference is
Objective: Estimate $\hat{X}_{t_0}$ – a super-resolved image at time $t_0$.

Inputs: 
- $y_t$ - input set of low resolution and noisy images.
- $s$ - the desired scaling factor (any integer).
- $q$ - the size of the low resolution patch ($R^L$).
- $p$ - size of the high resolution patch ($R^H$): $p = s(q - 1) + 1$.
- $\{Y_t\}_{t=1}^T$ - An initial estimate of the super-resolved sequence.

Initialization:
- Set $Z_{t_0} = Y_{t_0}$.
- Set $V$ and $W$ to be zero images of the same size as $Z_{t_0}$.

Fusion:
For each $(k, l) \in \Omega$ and each $(i, j, t)$ such that $(si, sj, t) \in N(k, l, t_0)$
1. Compute Weights: $w[k, l, i, j, t] = \exp\{-\|\hat{R}_{k,l}Z_{t_0} - \hat{R}_{si,sj}Y_t\|^2_2 / 2\sigma^2\}$.
2. Accumulate Inputs:
   - Extract the low-resolution patch $R^L_{i,j}y_t$,
   - upscale it by zero-filling,
   - accumulate it in its proper location
     $$ V = V + w[k, l, i, j, t] (R^H_{k,l})^T D^T_p R^L_{i,j}y_t. $$
3. Accumulate Weights: For each patch accumulated above, apply the following update of the weight image
   $$ W^+ = w[k, l, i, j, t] (R^H_{k,l})^T D^T_p R^L_{i,j}1. $$

Normalization: Set $Z_{t_0}[k, l] = V[k, l] / W[k, l]$.

Deblurring: Minimize
$$ \eta_{SR}^R (X) = \|Z_{t_0} - HX\|^2_2 + \lambda TV (X) \text{ w.r.t. } X, $$
and set the result to be $\hat{X}_{t_0}$.

Result: The output is $\hat{X}_{t_0}$.

Table 3.1: Summary of the core of the super resolution algorithm - processing one image using a given initial estimate for the super-resolved sequence.
Objective: Estimate $\left\{ \hat{X}_t \right\}_1^T$ - the super-resolved sequence.

Inputs:  
- $\{y_t\}_1^T$ - input set of low resolution and noisy images;  
- $s$ - the desired scaling factor (any integer).

Preprocessing: Compute $\{Y_t\}_1^T$ - a Lanczos interpolation of the input sequence.

Iterating: Perform the following steps per each iteration

1. Super-resolve each image using the algorithm outlined in Table 3.1, obtaining a new estimate for the super-resolved sequence $\left\{ \hat{X}_t \right\}_1^T$.

2. Update current estimation $\{Y_t\}_1^T = \left\{ \hat{X}_t \right\}_1^T$, and use as initial estimate for the following iteration.

Result: The output is $\left\{ \hat{X}_t \right\}_1^T$.

Table 3.2: Summary of the entire super resolution algorithm - used for resolving all the frames in the sequence.

computed, with a block size of $13 \times 13$ (high-res.) pixels. Thus, there is a total of almost 2,400,000 operations per pixel. This amount is of course very large, and must be reduced in order to make the algorithm practical. We will now describe a few speedup methods for the proposed algorithm. Several of the methods to speedup the NLM algorithm were suggested originally in [82], and were adopted in our simulations:

1. Computing the weights can be done using block differences in the low-resolution images, instead of on the interpolated images. This saves a factor of $\approx s^2$.

2. Computing fast estimations for the similarity between blocks, such as the difference between the average grey level or the average direction of the gradient, can eliminate many non-probable destinations from further processing. Such an approach was suggested in [82], and was found to be very effective for the original NLM algorithm.

3. If the patch used to compute the weights is rectangular with equal weights to all pixels, the computation of the weight can be sped up dramatically. Using the Integral Image ($II(x,y) = \sum_{i=1}^{x} \sum_{j=1}^{y} I(i,j)$), the block difference can be computed using a small constant number of calculations per pixel, regardless of block size. Using such a patch structure has only a slight effect on the quality of the outputs [126].
4. A coarse-to-fine approach reduces the effective search area for each pixel, thus reducing the number of required calculations.

5. The search area can be adapted to the temporal distance, making it small for nearby images and growing for images further away, also reducing the total effective search area.

6. Since weights are symmetrical, half the calculations can be saved, by applying computed weights to both participating patches.

7. Since most of the algorithm is local in nature, it lends itself easily to parallelization. As 4 and 8 processor configurations are currently widely available, this can be used for speeding up the algorithm by about one order of magnitude.

These suggested speedup methods can reduce the complexity by at least $3 - 4$ orders of magnitude without a noticeable drop in the quality of the outputs. This makes the proposed algorithm practical. As for the memory requirement, the proposed algorithm uses approximately as much memory as required to hold the entire processed sequence in the high resolution scale. This is usually not a limitation.

### 3.5 Experimental Results

#### 3.5.1 Experiments

In this section we validate the potential of the proposed algorithm (we use the simpler version discussed in Section 3.4.2) by presenting the obtained results of processing several image sequences. We start with one synthetic (text) sequence with global motion that comes to demonstrate the conceptual super-resolution capabilities of the proposed algorithms. Then we turn to three real sequences with a general motion pattern. As there are no published methods that perform super resolution on such general sequences, the comparison we provide is to a single image up-sampling using the Lanczos algorithm [133, 124], that effectively approximates the Sinc interpolation.

The Lanczos interpolation also serves as a benchmark for the synthetic case, even though super resolution algorithms that rely on global motion estimation can process this specific sequence quite successfully. This is because the benchmark for the proposed algorithm should also be able to handle all sequences, and not limited to the global motion case.
All the tests in this section are prepared in the following manner: An input sequence is blurred using a $3 \times 3$ uniform mask, decimated by a factor of $1 : 3$ (in each axis), and then contaminated by an additive noise with $std = 2$. All images are in the input range $[0, 255]$.

The first test is a very simple synthetic test, that motion-estimated-based super-resolution algorithms are expected to resolve well, intended to show that the proposed algorithm indeed achieves super resolution. A text image is used to generate a 9-image input sequence, by applying integer displacements prior to blurring, decimation and the addition of noise. The displacements are chosen so that the entire decimation space is covered (i.e. $dx = \{0, 1, 2\}$ and $dy = \{0, 1, 2\}$). The result for this test is shown in Figure 3.6, including a comparison to the results obtained by the regularized shift-and-add algorithm [52, 53], which is a conventional motion-estimation-based super-algorithm resolution. The block size used for computing the weights ($\hat{R}$) was set to $31 \times 31$, since the motion in the sequence is limited to displacements, and a larger block allows capturing the true displacement better (for true sequences, this size will be greatly reduces, as explained later). The value of $\sigma$ that moderates the weights was set to 7.5 (due to the large differences between white and black values in the scene). Two iterations were ran on the entire sequence, the first iteration used for computing the weights for the second iteration.

We also ran a similar test, displaying the behavior of the proposed algorithm and shift-and-add approach when one of the images from the set is omitted. The results for this test are displayed in Figure (3.7).

Even though the proposed algorithm does not exploit the fact the motion in the sequence is only global translation, it still achieves good results. The text is almost completely readable in the result of the proposed algorithm. This shows that sub-pixel details (relative to the low resolution grid) are indeed recovered by the proposed algorithm. In terms of Peak-Signal to Noise Ratio (PSNR)$^5$, a $3 : 1$ pixel-replication in each axis leads to $13.47$dB, the Lanczos interpolation gives $13.84$dB, a deblurred Lanczos interpolation gives $13.9$dB, the regularized shift-and-add gives $18.4$dB, and the proposed algorithm gives $18.48$dB, slightly out-performing the classic approach. For the test with one image omitted, the shift-and-add gives $18.16$dB, while the proposed algorithm slightly under-performs with a result of $17.7$dB. The similarity in performance between the shift-and-add approach and the proposed approach attests to the power of the proposed method, as even though

$^5$Defined as $20 \log_{10} \left( \frac{256}{\sqrt{MSE}} \right)$, where MSE is the mean-squared-error obtained per pixel.
Figure 3.6: Results for the synthetic text sequence. (a) Original (ground-truth) image. (b) Pixel replicated image, 13.47dB. (c) Lanczos interpolation, 13.84dB. (d) Deblurred Lanczos interpolation, 13.9dB. (e) Result of shift-and-add algorithm [52, 53], 18.4dB. (f) Result of proposed algorithm, 18.48dB.
The state of the art movie restoration methods like AWA, LMMSE either estimate motion and filter out the trajectories, or compensate the motion by an optical flow estimate and then filter out the compensated motion. Now, the motion estimation problem is fundamentally ill-posed. This fact is known as the aperture problem: trajectories are ambiguous since they could coincide with any premade in the space-time Sophie surface. In this paper, we try to show that, for denoising, the aperture problem can be taken advantage of. Indeed, by the aperture problem, many pixels in the neighboring frames are similar to the current pixel one wishes to denoise. Thus, denoising by an averaging process can use many more pixels just than the ones on a single trajectory. This observation leads to use for Denis a recently introduced denoising method, the NL-means algorithm. This static 3D algorithm outperforms motion compensated algorithms, as it does not lose motion details. It involves the whole movie sequence, including the current frame, and not just a trajectory. Experimental evidence will be given that it also improves the ‘dirt and scratches’ detection algorithm.

The state of the art movie restoration methods like AWA, LMMSE either estimate motion and filter out the trajectories, or compensate the motion by an optical flow estimate and then filter out the compensated motion. Now, the motion estimation problem is fundamentally ill-posed. This fact is known as the aperture problem: trajectories are ambiguous since they could coincide with any premade in the space-time Sophie surface. In this paper, we try to show that, for denoising, the aperture problem can be taken advantage of. Indeed, by the aperture problem, many pixels in the neighboring frames are similar to the current pixel one wishes to denoise. Thus, denoising by an averaging process can use many more pixels just than the ones on a single trajectory. This observation leads to use for Denis a recently introduced denoising method, the NL-means algorithm. This static 3D algorithm outperforms motion compensated algorithms, as it does not lose motion details. It involves the whole movie sequence, including the current frame, and not just a trajectory. Experimental evidence will be given that it also improves the ‘dirt and scratches’ detection algorithm.

Figure 3.7: Results for the synthetic text sequence, with one image missing. Left: Result of shift-and-add algorithm [52, 53], 18.16dB. Right: Result of proposed algorithm, 17.7dB.

It does not rely directly on the global motion assumption, it achieves similar results to a successful method that does.

It is obvious that while the PSNR measures of the shift-and-add approach and the proposed algorithm are similar, the results of the shift-and-add are visually more pleasing. This is to be expected, since this approach is able to fully benefit from the global motion assumption, while the proposed algorithm does not. This is due to the proposed algorithm trading off quality for increased robustness to general motion patterns (which the shift-and-add approach can’t cope with).

When observing the resulting images of the proposed algorithm, they seem somewhat over-sharpened. In fact, applying less deblurring iterations results in a more visually pleasing result, however, the objective PSNR measure yields a lower value for these images. It seems like a stronger deblurring mechanism, applied to the results of the fusion stage, can generate results that are both visually pleasing and relatively accurate reconstructions. Since the novelty of this chapter lies in the fusion stage, we have not experimented with other such mechanisms.

We now turn to test the proposed algorithm on real sequences, containing general motion patterns, which represent the actual problem the proposed algorithm is designed to tackle. The 3 sequences we test on are usually used for testing compression algorithms, referred to as ”Miss America”, ”Foreman”, and ”Suzie”. The super-resolution results for these sequences (scaling up by a factor of 3 to 1 in each axis) are shown in Figures 3.8, 3.9, and 3.10 respectively. All of these sequences (original, low-quality, Lanczos interpolation and processed sequences) appear in the first author’s website, at http:\www.cs.technion.ac.il\~matanpr\NLM-SR.
Each result shows several (3rd, 8th, 13th, 18th, 23th, and the 28th) frames from each sequence (each row represents one frame). For each frame, the input low resolution image, the ground truth image, the result of the Lanczos interpolation, and the result of the proposed algorithm are presented from left to right. It is very evident that the results of the proposed algorithm are dramatically better when compared to the Lanczos interpolation - the output is sharper, contains more details, and is more visually pleasing. It is important to note, that we display the results of the Lanczos interpolation without any post-deblurring. Applying deblurring to the Lanczos interpolation results in a slightly sharper image, but one that also contains unwanted artifacts (as can be seen in Figure 3.6 (c) and (d)) and the PSNR measures are also equivalent.

In processing all of these sequences, all 30 frames participated in the reconstruction of each image. The similarity block size used for computing the weights ($\hat{R}$) was $13 \times 13$, and did not vary between the different tests. The search area (i.e. the size of the neighborhood $N$) was determined manually, in order to reduce computation time\textsuperscript{6}: a search area of $13 \times 13$ pixels in each image for the “Miss America” sequence, $21 \times 21$ for the “Foreman” sequence, and $31 \times 31$ for the “Suzie” sequence. The parameter $\sigma$ was set to 2.2 for all sequences. Two iterations were again used, where the first provides the updated weights for the second iteration. Just to put things into perspective, we add that the entire simulation is done on Matlab, running on a regular Pentium 3GHz (2GB RAM) machine, requiring approximately 20 seconds per frame for the most demanding case – the “Suzie” sequence with high-resolution frame size of $210 \times 250$ pixels.

Table 3.3 presents the mean PSNR for each of the three test sequences, evaluating the quality of the pixel-replicated scaled-up sequence, the Lanczos interpolation, and the results we obtain after the first and the second iterations of the proposed algorithm. While the super-resolution results show higher PSNR, we see that (i) the PSNR gain is mild, and does not reflect the visual quality of the sequences obtained; and (ii) Even though the first iteration result is typically inferior in visual quality, its PSNR assessment tends to be higher.

Beyond the usual and common complaint on the inability of PSNR to grasp image quality in general, it seems that our algorithm in particular is very sensitive to this measure. One possible reason for this could be the fact that in avoiding explicit motion estimation, the result we obtain is slightly shifted (or warped) with respect to the reference it aims

\textsuperscript{6}Applying a bigger search area results in an only slightly different super-resolution outcome.
Figure 3.8: Results for the 3rd, 8th, 13th, 18th, 23th, and the 28th frames from the "Miss America" sequence. From left to right: low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.
Figure 3.9: Results for the 3rd, 8th, 13th, 18th, 23th, and the 28th frames from the "Foreman" sequence. From left to right: low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.
Figure 3.10: Results for the 3rd, 8th, 13th, 18th, 23th, and the 28th frames from the "Suzie" sequence. From left to right: low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.
to recover. This, of-course, leads to disastrous drop in PSNR. Another reason that may account for such behavior is the over-sharpening that the deblurring stage introduces. We intend to further explore this matter in our future work, as we hope to provide visually pleasing results that are also backed up by good PSNR behavior.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Pixel Replication</th>
<th>Lanczos Interpolation</th>
<th>Deblurred Lanczos Interpolation</th>
<th>Our Result (1st Iteration)</th>
<th>Our Result (2nd iteration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miss-America</td>
<td>31.43</td>
<td>34.08</td>
<td>34.26</td>
<td>35.91</td>
<td>34.97</td>
</tr>
<tr>
<td>Foreman</td>
<td>28.99</td>
<td>31.25</td>
<td>31.13</td>
<td>34.27</td>
<td>33.13</td>
</tr>
<tr>
<td>Suzie</td>
<td>30.02</td>
<td>31.4</td>
<td>31.59</td>
<td>33.74</td>
<td>33.32</td>
</tr>
</tbody>
</table>

Table 3.3: Mean-PSNR results for the three test sequences. This tables gives the PSNR for the pixel-replicated sequences, the Lanczos interpolation, the Lanczos interpolation with consequent deblurring, and the results of the proposed algorithm (first and second iteration).

3.5.2 Discussion

The results obtained are very encouraging, being artifact-free and of high quality. Still, we believe that these results could be further improved and extended in various ways. The manual selection of the various parameters could be replaced by an adaptive setting of their values. One specific parameter of importance is the filtering parameter $\sigma$, effecting the weights computation. Selecting this parameter in a locally adaptive way seems like a natural step to take, allowing the algorithm to adapt better to both small and large details.

Similarly, an adaptive selection of the window size (both for the weights computation and in the algorithm itself), as done in [13, 29], can improve the algorithm’s performance. A large block can help estimate the motion more accurately. A small block can better adapt to general and varying motion. Adaptively selecting the window size, as in [13], is expected to deliver much better results than just settling for a global tradeoff.

A similar, yet somewhat different concept, is a better control over the search area. While the estimated motion pattern is fuzzy, reducing the number of candidate locations can help reduce complexity and improve the results. Relying on some coarse optical flow computation as a preceding step for the fuzzy motion estimation can bring these improvements. An alternative is computing a coarse fuzzy motion pattern in a coarser
scale, and proceeding to search in the actual scale only around the likely coarse destination.

A somewhat different direction for further research is adopting the fuzzy motion concept in other approaches to the super resolution problem. In the video denoising field, the very successful NLM was followed by even more successful contributions using fuzzy motion, such as the SW3D [29] and the method relying on sparse and redundant representations [100]. We believe that these approaches, and others, can also serve as the basis for more successful super resolution algorithms that do not rely on explicit motion estimation.

### 3.6 Summary

This chapter introduced a novel and successful super resolution algorithm, which does not rely on explicit motion estimation. Instead, a local and patch-based approach is combined with fuzzy motion estimation. This allows the algorithm to handle diverse motion fields, instead of the common global motion limitation that characterizes traditional super-resolution methods.

The algorithm developed here is an extension of the very successful video denoising Non-Local-Means algorithm, reported in [15]. Its ability to denoise image sequences without explicit motion estimation, which was previously considered necessary in this field, led us to start our search for an explicit-motion-estimation-free super resolution algorithm there.

By analyzing the forces existing in the NLM algorithm, we were able to write an energy function, whose minimization leads to a powerful denoising algorithm, of which the NLM is a special case. Then, we extended this energy function to the super resolution problem, by making the necessary changes. Starting from this energy minimization, we developed a simple yet very effective super resolution algorithm.

Finally, we have processed some real sequences that contain complex motion patterns with the proposed algorithm. The obtained results are artifact free and of high quality, thus proving the ability of the proposed method to handle general sequences. Lastly, we have made several suggestions as to directions for further research.
3.A Proofs

Theorem 1 Assuming that
1. \((i, j) \in N(k, l)\) implies \((k, l) \in N(i, j)\), and
2. \(w[k, l, i, j] = w[i, j, k, l]\),
the following two equalities hold true:

\[
\sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^{T} R_{k,l} x = \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{i,j}^{T} R_{i,j} x \quad (3.A.1)
\]

and

\[
\sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{i,j}^{T} R_{k,l} x = \sum_{(k,l) \in \Omega} \sum_{(i,j) \in N(k,l)} w[k, l, i, j] R_{k,l}^{T} R_{i,j} x \quad (3.A.2)
\]

Proof: We start with Equation (3.A.1). Both sides of this equation describe processes where image patches are piled with proper weights. Consider an arbitrary location \((m, n)\) in the resulting image, and let’s see which patches are accumulated at this point as a center one. Recall that the operator \(R_{k,l}^{T}\) assigns a patch to location \((k, l)\).

Looking at the Left-Hand-Side (LHS), the only patch to be positioned in location \((m, n)\) (i.e., to be multiplied by \(R_{m,n}^{T}\)) is \(R_{m,n} x\), but this is done several times, accumulating a total weight of \(\sum_{(i,j) \in N(m,n)} w[m, n, i, j]\).

The Right-Hand-Side (RHS) performs a slightly more involved accumulation. Among all \((k, l)\) pixels in the image, only those that are neighbors to \((m, n)\) are relevant, and each of those contributes one patch of the form \(R_{m,n} x\) with a weight \(w[k, l, m, n]\). Thus, again we obtain that only one patch is used, but accumulated several times. The total weight of this accumulation is given by \(\sum_{(k,l) \in N(m,n)} w[k, l, m, n]\). Due to the symmetry of the weights, we have that

\[
\sum_{(k,l) \in N(m,n)} w[k, l, m, n] = \sum_{(k,l) \in N(m,n)} w[m, n, k, l], \quad (3.A.3)
\]

which is the same as the LHS accumulated weight.

Turning to the equality in Equation (3.A.2), we adopt a similar analysis. Starting this time with the RHS, the overall patches to be accumulated in location \((m, n)\) (i.e., those multiplied by \(R_{m,n}^{T}\)) are given by

\[
\sum_{(i,j) \in N(m,n)} w[m, n, i, j] R_{i,j} x.
\]
In the LHS, only pixels \((k,l)\) that have \((m,n)\) in their neighborhood are relevant in the outer summation. For those, only one neighbor, the \((m,n)\) one, is relevant, as this is the only one multiplied by \(R_{m,n}^T\), and thus positioned in the location we consider. Thus, the accumulation in this case becomes

\[
\sum_{(k,l) \in N(m,n)} w[k,l,m,n] R_{k,l} x,
\]

and due to the symmetry of the weights, this is the same as the RHS term.

\[\text{Theorem 2} \quad \text{The matrix } A = \lambda I + \sum_{kl} \bar{w}[k,l] R_{k,l}^T R_{k,l} \text{ is diagonal and positive definite.}\]

\textbf{Proof:} Consider first the term \(R_{k,l}^T R_{k,l}\) for an arbitrary \((k,l)\) position. This matrix is positive semi-definite by definition, due to the multiplication of a matrix by its adjoint. Furthermore, as an operator that manipulates an image, the multiplication by \(R_{k,l}\) extracts a patch from location \((k,l)\). The later multiplication by \(R_{k,l}^T\) creates a zero-filled image, with the same patch returned to the \((k,l)\) location. Thus, this matrix is a diagonal matrix, with ones on the main diagonal for pixels belonging to the patch, and zeros elsewhere.

We assume that the weights \(w[k,l,i,j]\), and thus also \(\bar{w}[k,l]\), are non-negative by their definition, and thus the matrix \(\sum_{kl} \bar{w}[k,l] R_{k,l}^T R_{k,l}\) is a non-negative weighted average of positive semi-definite and diagonal matrix. Thus, the result is also diagonal and semi-definite. The addition of \(\lambda \cdot I\) preserves the diagonal structure, and lifts the smallest eigenvalue of the overall matrix to \(\lambda > 0\), thus leading to a strictly positive-definite matrix, as claimed.
Chapter 4

Super Resolution With
Probabilistic Motion Estimation

The contents of this chapter had been published in the paper M. Protter and M. Elad “Super-resolution with probabilistic motion-estimation”, IEEE Transactions on Image Processing, Vol. 18, No. 8, Pages 1899-1904, August 2009 [101].

Super-resolution reconstruction (SRR) has long been relying on very accurate motion estimation between the frames for a successful process. However, recent work proposes SRR that bypasses the need for an explicit motion estimation [102, 120] (see also previous chapter). In this chapter we present a new framework that ultimately leads to the same algorithm as in our prior work [102]. The contribution of this chapter is two-fold. First, the suggested approach is much simpler and more intuitive, relying on the classic SRR formulation, and using a probabilistic and crude motion estimation. Second, the new approach offers various extensions not covered in our previous work, such as more general re-sampling tasks (e.g., de-interlacing).

4.1 Introduction

Super-Resolution Reconstruction (SRR) proposes a fusion of several low quality images \( \{y_t\}_{t=1}^T \) into one higher quality result \( x \) with better optical resolution. A wide variety of SRR algorithms have been developed in the past two decades – see [102] for a list of representatives of this vast literature. A popular model used for relating the measurements to the super-resolved image, assumes that \( \{y_t\}_{t=1}^T \) are generated from \( x \) through a sequence of operations that includes (i) geometrical warps \( F_t \), (ii) a linear space-invariant blur \( H \),
(iii) a decimation step represented by $D$, and finally (iv) an additive zero-mean white and Gaussian noise $n_t$ that represents both measurements noise and model mismatch\footnote{In [53], the model mismatches are modeled as Laplacian, with $L_1$ penalization as to obtain robustness to outliers. In our work, we choose a Gaussian model, which simplifies the algorithmic development. Nevertheless, a robustness to outliers is obtained by the probabilistic approach, as will be discussed later, in Section 4.3.} [53]. These are all linear operators, represented by a matrix multiplying the image they operate on. We assume hereafter that $H$ and $D$ are identical for all images in the sequence. This model leads to the following set of equations:

$$y_t = DHF_t x + n_t \quad \text{for} \quad t = 1, 2, \ldots, T. \quad (4.1)$$

The recovery of $x$ from $\{y_t\}_{t=1}^T$ is thus an inverse problem, combining denoising, deblurring, scaling-up operation, and fusion of the different images, all merged to one. By setting $F_1 = I$, we refer to $y_1$ as the reference image, and aim to construct $x$ as its super-resolved version.

SRR relies on the assumption that $D$, $H$, and $F_t$ are known, or can be reliably estimated from the given data. In particular, such reconstruction relies on the ability to estimate the motion in the scene with a sub-pixel accuracy, so as to enable the merger of the different image sampling grids properly. Many SRR algorithms start with such an estimating of the motion in the sequence (e.g., [71, 111, 7, 53, 54]), or couple it with the recovery process, as a joint-estimation task [69, 127, 114].

Highly accurate general motion estimation, known as optical flow, is a severely under-determined problem. When inaccurately estimated motion is used within one of the existing SRR algorithms, it often leads to disturbing artifacts that cause the output to be inferior even when compared to the given measurements. For this reason, some simplifying assumptions as to the structure of the motion are made, such as global warps or rigid bodies. Only under these assumptions is the motion estimation in currently available SRR algorithms accurate enough to lead to a successful reconstruction of a super-resolved image. This had led to the commonly agreed and unavoidable conclusion that general content movies are not likely to be handled well by classical SRR techniques.

Recently, several papers have tried to circumvent this problem by avoiding explicit motion estimation altogether [102, 120]. The method in [120] relies on extending the steerable kernel method to multi-frame super-resolution. The method in [102] generalizes the very successful Non-Local-Means (NLM) [17] denoising method to performing super-
resolution. The derivation of the SRR algorithm in [102] is done by defining an energy functional that explains the NLM, and then modifying it to serve the SRR task. Both methods do not explicitly estimate the motion, and both are shown to be able to handle general content video sequences quite successfully.

In this chapter, we approach the explicit-motion-estimation-free SRR from a different perspective. Our starting point is the classic SRR, as in [53], and the bijective motion between pixels in each pair of images is replaced with a probabilistic motion field. This simple and alternative derivation is shown to lead to the same line of algorithms that are proposed in [102]. Furthermore, the framework proposed here allows different extensions, such as a treatment of spatio-temporal re-sampling problems. We show this adaptation in general, and demonstrate its applicability on the de-interlacing problem.

The structure of this chapter is as follows. Section 4.2 describes a classic SRR formulation, as used in [71, 111, 7, 53, 54], on which we build our eventual algorithm. Section 4.3 presents the use of probabilistic motion with the classic SRR, and develops the proposed algorithm. The adaptation to other re-sampling tasks is also described. Section 4.4 provides results for SRR and de-interlacing, demonstrating the abilities of the proposed method. We conclude in Section 4.5, outlining the key contribution of this work.

4.2 Classic Super-Resolution: Background

Using the model in Equation (4.1), the Maximum-Likelihood (ML) estimate of \( \mathbf{x} \) is obtained by minimizing the penalty function

\[
\epsilon_{ML}^2 (\mathbf{x}) = \frac{1}{2} \sum_{t=1}^{T} \| \mathbf{DH} \mathbf{F} t \mathbf{x} - \mathbf{y}_t \|^2
\]  

(4.2)

with respect to \( \mathbf{x} \). This expression suggests that we should seek an image \( \mathbf{x} \) that explains best the set of measurements given. Minimization of (4.2) leads to

\[
\frac{\partial \epsilon_{ML}^2 (\mathbf{x})}{\partial \mathbf{x}} = \sum_{t=1}^{T} \mathbf{F}_t^T \mathbf{H}^T \mathbf{D}^T (\mathbf{DH} \mathbf{F}_t \mathbf{x} - \mathbf{y}_t) = 0.
\]  

(4.3)

Denoting \( \mathbf{A} = \sum_{t=1}^{T} \mathbf{F}_t^T \mathbf{H}^T \mathbf{D}^T \mathbf{DH} \mathbf{F}_t \) and \( \mathbf{b} = \sum_{t=1}^{T} \mathbf{F}_t^T \mathbf{H}^T \mathbf{D}^T \mathbf{y}_t \), we face a linear system of equations \( \mathbf{A} \hat{\mathbf{x}}_{ML} = \mathbf{b} \).

In many cases the measurements are not sufficient for recovering \( \mathbf{x} \). This is manifested in a singular or possibly ill-conditioned matrix \( \mathbf{A} \). In such cases a regularization is required.
The Maximum A-posteriori Probability (MAP) estimation proposes a penalty of the form

$$\epsilon_{MAP}^2(x) = \epsilon_{ML}^2(x) + \lambda \cdot R(x),$$

(4.4)

where the functional $R$ is a regularization term that adds an algebraic stability to the inversion of $A$. Beyond the gained stability, $R$ also introduces the means to incorporate prior knowledge about the sought $x$, such as spatial smoothness, sparsity of its wavelet representation, minimum entropy, etc. In this work we shall use the total variation choice $R(x) = TV(x)$ that accumulates the gradients norms with $\ell^1$, forcing (piece-wise) smoothness [108]. Thus, the MAP estimate in our case becomes the minimizer of

$$\epsilon_{MAP}^2(x) = \frac{1}{2} \sum_{t=1}^{T} \|DHF_t x - y_t\|_2^2 + \lambda \cdot TV(x),$$

(4.5)

which is typically obtained by an iterative algorithm [71, 111, 69, 7, 53, 54, 127, 114]. This is the core technique we build upon.

In all of the above discussion we assume that the operators $D$, $H$, and $F_t$ are known. The decimation $D$ is dependent on the resolution scale-factor we aim to achieve, and as such, it is easily fixed. In this work we shall assume that this resolution factor is an integer $s \geq 1$ on both axes. In most cases the blur $H$ refers to the camera PSF, and therefore it is also accessible. Even if this is not the case, the blur is typically dependent on few parameters, and those, in the worst case, can be manually set.

As opposed to these operators, the matrices $F_t$ are harder to obtain. They depend on the scene and require highly accurate motion estimation for their construction. As such accuracy is hard to obtain in general, classical SRR algorithms often assume a simple motion pattern, such as pure translation or global affine warp. Attempts to embed the motion estimation within the SRR process have been made, with little success [69, 127, 114]. As already mentioned, inaccurately estimated motion within SRR often leads to disturbing artifacts that cause the output to be inferior even when compared to a simple interpolated version of $y_1$. This fact motivated a quest for bypassing explicit motion estimation, as indeed practiced in [102, 120].

4.3 The Proposed Algorithm

4.3.1 The New Formulation

We aim to introduce the notion of probabilistic motion estimation to the above classic SRR formulation. Note that the warp operator $F_t$ considers a bijective (one-to-one) cor-
response between pixels in the reference and the \( t \)-th image, and as such, it introduces sensitivity to errors. We replace this motion field with a probabilistic one that assigns each pixel in the reference image with many possible correspondences in all the images in the sequence (including itself), each with an assigned probability of being correct.

How could this become useful for super-resolution for handling general motion patterns? Here we offer one possible way that illustrates that such idea could fit in SRR. The operator \( F_t \) represents the motion field between the first image and image \( t \), by indicating for each pixel in the first image its destination in image \( t \). This is equivalent to independently listing a single 2D translation vector for each pixel (where each pixel is assigned a translation, independent of other pixels). Therefore, the entire motion field is represented as a collection of various displacement vectors.

If the size of the maximal translation is at most \( D \) pixels, then a set of \( M = (2D + 1)^2 \) displacements covers all the possible ones to be encountered. By defining \( \{ F_m \}_{m=1}^M \) to be this set of global translations\(^2\), we can write the following equation

\[
F_t x = \sum_{m=1}^{M} Q_{m,t} F_m x,
\]

which describes the action of warping the image \( x \) based on the operator \( F_t \). The matrices \( \{ Q_{m,t} \}_{1}^{M} \) are diagonal weighting ones, containing 1-es along the main diagonal for pixels whose motion is the displacement \( F_m \), namely \([dx(m), dy(m)]\), and zeros otherwise. In such a way, it is possible to represent the most complex of motion fields by a linear combination of global translations.

In this formulation, we have replaced the single warping operator with a linear combination of global translation representing the same general motion field. Still, this notation implies a one-to-one relationship between pixels in both images. The next natural step for introducing a probabilistic motion field is to relax the definition of \( Q_{m,t} \), enabling continuous values to reflect varying confidences per pixel and per motion trajectory. This leads to a newly defined super-resolution penalty that replaces the use of \( F_t \) by their decompositions as in (4.6).

While this seems like a worthy path to consider, in this work we slightly divert from this approach, in a quest for a yet simpler algorithm. We modify the ML formulation

\(^2\)For simplicity, we shall use a set of integer displacements only.
posed in Equation (4.2) by proposing the following probabilistic ML (PML) penalty

\[ \epsilon_{PML}^2(x) = \frac{1}{2} \sum_{m=1}^{M} \sum_{t=1}^{T} \| \text{DHF}_m x - y_t \|^2_{W_{m,t}}. \] (4.7)

We rely on the same intuition as described above, but in a slightly different way. Rather than accumulate the various global translations to form the effect of \( F_t \) as in Equation (4.6), we accumulate the least-squares errors that result from such global displacements\(^4\), and assign a weight matrix \( W_{m,t} \) to each. Notice that the weights used in Equation (4.7) are different from those introduced in (4.6). Whereas \( Q_{m,t} \) are defined for each pixel in the high resolution image, \( W_{m,t} \) are also diagonal matrices, but defined over the low-resolution grid. We shall proceed with the assumption that \( W_{m,t} \) are known, and revisit their computation in Section 4.3.5.

It is important to note that although this formulation contains only global translations, it is still able to process any complex motion field, using the same rational that has led to Equation (4.6). If the motion field is known, it can be re-created by properly assigning the values of \( W_{m,t} \) to be 1-es for those pixels whose motion is \( F_m \) and zeros for all others.

One could interpret the above expression as a marginalization of the least-squared error term with respect to the motion probability density function, in a way that resembles the concept proposed in [96]. The authors of [96] perform such a marginalization in order to avoid inaccuracies in the motion estimation, but their integration is performed over the parameters of a global motion model. In our case, very similar to the video denoising scenario, we handle local motion, and the probabilistic view-point contributes both to a better handling of the estimated motion inaccuracies and also to the noise reduction.

As a final point in this section, we return to the matter of robustness. The usage of the above PML has another distinct advantage of robustifying the algorithm to outliers. One such example could be a scenario in which one of the images in the set is an outlier. In such a case, the weights assigned to the pixels in this image will be zeros, since the images do not match. Therefore, those pixels will effectively not be considered in the minimization, or in other words, treated as outliers, as required.

### 4.3.2 Separating the Blur Treatment

Our task is the minimization of a functional that has two terms in it: \( \epsilon_{PML}^2(x) \) and a regularization (e.g., TV). Rather than handling this problem directly, we decompose

\[^3\text{We use the notation } \| a \|_W^2 = a^T W a.\]
\[^4\text{It is possible to use other sets of warps, such as ones that allow rotations as well.}\]
it, following the methods developed in [47, 53, 54]. Since both \( H \) and \( F_m \) are space-invariant operators, they can be assumed to have a block-circulant structure (assuming a cyclic boundary treatment), and as such, they commute. Thus, defining \( z = Hx \), we concentrate first on estimating the “blurry” high resolution image \( z \) by minimizing

\[
\epsilon_{PML}^2(z) = \frac{1}{2} \sum_{m=1}^{M} \sum_{t=1}^{T} \|D_{\text{W}}F_m z - y_t\|_W^2,
\]

which will be the fusion step. Then we apply a conventional deblurring step, by minimizing

\[
\epsilon_{DB}^2(x) = \|Hx - z\|_2^2 + \lambda \cdot TV(x).
\]

This two step process is sub-optimal to the joint treatment, but nevertheless leads to a simplified algorithm. As the second step is conventional and well-known, we focus hereafter on the fusion step. Note that the deblurring mechanism chosen here is relatively simple and could be replaced by more advanced techniques, thereby leading to better results.

### 4.3.3 The Algorithm: A Matrix-Vector Version

We focus now on the minimization of Equation (4.8). The derivative of this functional is given by

\[
\frac{\partial \epsilon_{PML}^2(z)}{\partial z} = \sum_{m=1}^{M} \sum_{t=1}^{T} F_m^T D^T W_{m,t}(DF_m z - y_t),
\]

which leads to a linear system of equations.

We introduce the following new notations, in order to simplify the obtained expressions

\[
\tilde{W}_m = \sum_{t=1}^{T} W_{m,t} \quad \text{and} \quad \tilde{y}_m = \sum_{t=1}^{T} W_{m,t} y_t.
\]

The matrix \( \tilde{W}_m \) is a diagonal matrix, as it is the sum of diagonal matrices. We obtain

\[
\sum_{m=1}^{M} F_m^T D^T \tilde{W}_m DF_m z = \sum_{m=1}^{M} F_m^T D^T \tilde{y}_m.
\]

This linear system of equations seems complicated. As we show next, it can be rewritten for each pixel in \( z \) in a closed form, revealing a simple structure that leads to a stable solution.

### 4.3.4 The Algorithm: A Pixel-wise Version

The Right-Hand-Side (RHS) in Equation (4.12) is an image of the same size as \( z \). Furthermore, as we are about to show, the matrix multiplying \( z \) on the Left-Hand-Side (LHS) is a
diagonal positive definite matrix. Thus, we can turn the above vector-matrix formulation into a pixel-wise one. Let us consider a specific pixel at location \([i, j]\) in \(z\), and see its construction. As this pixel is dependent only on the \([i, j]\)-th pixel in the RHS image (up to a scalar being the diagonal element in the matrix on the LHS), we start by constructing this element.

For a specific \(F_m\) that shifts by \([dx(m), dy(m)]\), the term \(F_m^T v\) positions the \([i + dx(m), j + dy(m)]\)-th element from the image \(v\) in the destination \([i, j]\) (the transpose has the effect of an inverse displacement). The image \(u = D^T \tilde{y}_m\) is a scale-up version of the low-resolution image \(\tilde{y}_m\) by zero-filling. This implies that if the location \([i + dx(m), j + dy(m)]\) is not an integer multiple of \(s\) (the resolution ratio), this location has a zero entry. Otherwise, the entry is simply \(\tilde{y}_m[k, l]\), where \([k, l] = \frac{[i + dx(m), j + dy(m)]}{s}\). Thus, at location \([i, j]\) we get

\[
RHS[i, j] = \sum_{[k, l] \in N(i, j)} \tilde{y}_m[k, l],
\]

where we have defined the neighborhood set

\[
N(i, j) = \{[k, l] \mid \forall m \in [1, M], s \cdot k = i + dx(m), s \cdot l = j + dy(m)\}
\]

Plugging the definition of \(\tilde{y}_m\) from Equation (4.11) yields

\[
RHS[i, j] = \sum_{[k, l] \in N(i, j)} \sum_{t=1}^{T} W_{m,t}[k, l] y_t[k, l].
\]

In this expression, \(W_{m,t}[k, l]\) refers to the entry on the main diagonal in \(W_{m,t}\) that multiplies the \([k, l]\) entry in \(y_t\).

We now discuss the Left-Hand-Side (LHS) in (4.12). The operator \(D^T \tilde{W}_m D\) within this expression is a diagonal matrix that decimates an image by a factor \(s\) in each axis, weights each pixel by the diagonal weight matrix \(\tilde{W}_m\), and then up-scales back the image using the same factor by zero-filling. This means that when operating on an image \(v\), a pixel in location \([i, j]\) is nulled if \([i, j] / s\) is a non-integer, and otherwise it is simply weighted, i.e. it becomes \(\tilde{W}_m[i, j] \cdot v[i, j]\).

When the operator \(F_m^T D^T \tilde{W}_m D F_m\) is applied to the \([i, j]\)-th pixel in \(z\), it shifts it to the \([i + dx(m), j + dy(m)]\)-th location, nulls it or weights it, based on whether \([i + dx(m), j + dy(m)] / s\) is an integer, and finally shifts the outcome back by \([-dx(m), -dy(m)]\) to its original place, \([i, j]\). The fact that the operator \(F_m^T D^T \tilde{W}_m D F_m\) returns every pixel to its original location means that this matrix is diagonal, as every output pixel depends only
on the value of the input pixel in the same location. Thus, the scalar that multiplies the 

\[
\text{LHS}[i,j] = \sum_{[k,l] \in N(i,j)} \tilde{W}_m[k,l]z[i,j] = \sum_{[k,l] \in N(i,j)} \sum_{t=1}^T W_{m,t}[k,l]z[i,j],
\]

(4.16)

where we have made use of the definition of \( \tilde{W}_m \) in Equation (4.11). This expression sums all the weights in (4.15), serving as a normalization term. Assuming that this sum is positive (i.e. at least one weight is non-zero), combining Equations (4.15) and (4.16) leads to a closed form expression for the \([i,j]\)-th pixel in the estimated \( \hat{z} \),

\[
\hat{z}[i,j] = \frac{\sum_{[k,l] \in N(i,j)} \sum_{t=1}^T W_{m,t}[k,l]y_t[k,l]}{\sum_{[k,l] \in N(i,j)} \sum_{t=1}^T W_{m,t}[k,l]}.
\]

(4.17)

and the resemblance to the fusion algorithm in our prior work is evident (see Equation (30) in [102]). Just as explained there, the similarity of the final algorithm to the NLM stands out, but there is a subtle difference between the two, related to the domain of the averaging. The proposed algorithm differs considerably from an interpolation followed by application of NLM – we show a visual comparison between the two in the following section.

### 4.3.5 Computing the Weights

In order to complete the description of the algorithm we must explain how \( W_{m,t}[i,j] \) are computed. Based on Equation (4.8), these weights are supposed to encompass the fit, per pixel, of the desired high resolution image \( z \) after being transformed by \( F_m \) and decimated by \( D \), with the input image \( y_t \). Thus, the weights could be related to the error \( D F_m z - y_t \).

In order to better estimate the fit, we propose to use some spatial support for each pixel instead of computing the plain difference. Defining \( R_{i,j} \) as an operator that extracts a patch of a fixed and pre-determined size (say \( q \times q \) pixels) from an image, the weights are computed by

\[
W_{m,t}[i,j] = \exp \left\{ -\frac{\| R_{i,j}(DF_mz - y_t) \|_2^2}{2\sigma^2} \right\} \cdot f \left( \sqrt{(dx(m))^2 + (dy(m))^2 + (t - 1)^2} \right).
\]

(4.18)

The first part in the above formula gives a value that is inversely proportional to the Euclidean distance between the transformed image \( DF_mz \) and the input image \( y_t \), computed over some support around each pixel. The second part adds a decaying weight as a
function of the displacement and time shift magnitudes versus the reference frame. The function \( f \) can be chosen as any monotonically non-increasing function (e.g., box function or Gaussian bell).

The computation of the weights requires the use of the unknown \( z \). Instead, the weights are computed at the beginning by using an estimated version of \( z \), being a scaled-up version of the reference frame \( y_1 \). This scale-up is done using a conventional image interpolation algorithm such as bilinear, bicubic, or the Lanczos method. As this is a crude version of the desired outcome, the process can be iterated, using the newly estimated image \( \hat{z} \). In our tests we employ two such iterations only.

The method in which the weights are computed is reminiscent of classic block-matching based SR algorithms (e.g., [19]). However, there is a key difference between these algorithms and the one proposed here. In the classic block-matching based SR, block-matching is used to determine a single trajectory for each pixel in the current image (that is being processed) to the others, and as such, estimate the motion. In contrast, in the proposed algorithm, a method similar to block-matching is used to estimate the probability of each trajectory. Once computed, all these trajectories are considered together, according to their probabilities, as opposed to selecting only the single most likely one. This difference is what enables the proposed algorithm to handle complex scenarios where highly accurate motion estimation is not currently possible.

### 4.3.6 Other Resampling Tasks

We wish to adapt the proposed framework to other re-sampling tasks, such as de-interlacing, inpainting and more. We start by explaining this extension intuitively. The re-sampling task can be considered as computing pixel values for only some of the pixels in each image ("missing pixels"). For example, the de-interlacing task may be viewed as providing pixel values only for the even rows for the odd numbered fields, as well as for the odd rows in the even numbered fields. Formulating this idea, given each input image (or field) \( y_t \), it can be linked to the original (unknown) image \( Y_t \) using a masking operator \( M_t : y_t = M_t Y_t \).

Simply put, \( M_t \) discards all un-sampled pixels. It is a binary matrix, with as many rows as the number of pixels in \( y_t \) and as many columns as pixels in \( Y_t \), with entries of ones indicating which pixels are to be kept. Note that \( y_t \) contains only sampled pixels. In the in-painting case, it contains only the un-masked pixels.

In line with the idea of the probabilistic motion estimation, \( Y_t \) can be constructed as
a (pixel-wise) weighted average of different transformations of the target image $\mathbf{x}$. The image $\mathbf{x}$ that we seek should be as similar as possible to each $\mathbf{y}_t$, after undergoing each of the transformations and the relevant masking. This required similarity is weighted on a pixel-wise basis, according to the (local) probability of the specific transformation having taken place. Put into the maximum likelihood formulation, a penalty function very similar to Equation (4.7) arises, where the decimation operator is replaced by $\mathbf{M}_t$,

$$
\epsilon^2_{PML}(\mathbf{x}) = \frac{1}{2} \sum_{m=1}^{M} \sum_{t=1}^{T} \| M_t \mathbf{H} \mathbf{F}_m \mathbf{x} - \mathbf{y}_t \|^2_{\mathbf{W}_{m,t}} .
$$

(4.19)

Minimizing this functional proceeds very similarly to the steps described before. The treatment of the blur is separated, and a pixel-wise formula for the values of $\mathbf{z}$ is given by Equation (4.17). The difference is in the order of summation, as the neighborhood $\mathcal{N}(i,j)$ of a pixel is now time (and spatial) dependent. This is because the masking may be different for every image in the sequence.

The weights for this formula are computed very similarly to the SRR case, described in Equation (4.18). However, these tasks can benefit from computing the weights in high resolution scale. Thus, if we consider that $\mathbf{W}_{m,t}$ is for the coarse scale, we denote $\tilde{\mathbf{W}}_{m,t} = \mathbf{M}_t \mathbf{W}_{m,t}$, with $\mathbf{W}_{m,t}$ being the same size as $\mathbf{Y}_t$. The formula for each entry of $\tilde{\mathbf{W}}_{m,t}$ (when arranged as an image) is therefore the same as in Equation (4.18), but with $\mathbf{F}_m \mathbf{z} - \mathbf{Y}_t$ replacing $\mathbf{D} \mathbf{F}_m \mathbf{z} - \mathbf{y}_t$. In these weights, $\mathbf{Y}_t$ is an interpolated version of $\mathbf{y}_t$ (with the interpolation method depending on the specific task). Of course, these weights should be computed only for pixels that are kept after the masking $\mathbf{W}_{m,t} = \mathbf{M}_t \tilde{\mathbf{W}}_{m,t}$.

### 4.4 Experimental Results

We now turn to demonstrate the potential of the proposed SRR algorithm by presenting the results for image sequences with a general motion pattern. Since the algorithm tested here is the very same one as in [102], we concentrate on one such example. The original sequence “Suzie” has been blurred using a $3 \times 3$ uniform mask, decimated by a factor of 1 : 3 (in each axis), and then contaminated by additive white zero-mean Gaussian noise with $\text{STD} = 2$. The degraded sequence was the input to the proposed SRR algorithm, and Figure 4.1 presents the obtained results for the 3rd, 8th, 13th, 18th, 23th, and the 28th frames$^5$.

$^5$The sequences appearing in this section (input and output) and others from [102], with the various parameters used to generate them, can be found at http://www.cs.technion.ac.il/~matanpr/NLM-SR.
We also compare the results using the average PSNR, an objective quality measure
\[
PSNR = 10 \log_{10} \left( \frac{255^2}{\| \hat{X} - X \|_2^2} \right) \text{ [dB]},
\]
where \( \hat{X} \) and \( X \) are the original and reconstructed images respectively. For the above sequence, the PSNR for the pixel replicated low-quality sequence, the Lanczos results and the proposed algorithm are 30dB, 31.4dB, and 33.74dB respectively.

The above presented sequence, and the others in [102], are all synthetic, in the sense that the blur kernel and decimation are known. Note, however, that the motion in the sequences is real, and not synthetically generated. In order to demonstrate the proposed algorithm on a directly captured sequence, we provide a second experiment “Trevor”, whose results are displayed in Figure 4.2. In this case, there is no ground-truth image available to compare to. Therefore, to demonstrate that a super-resolution effect is achieved, a comparison is made to an interpolated sequence. This interpolation is obtained by a Lanczos interpolation, followed by NLM filtering for denoising, and then deblurring. This comparison serves two goals: (1) It indeed verifies that the proposed algorithm obtains SR effect; and it demonstrates the difference between simply running NLM and deblurring after up-scaling, compared to running the proposed algorithm. This comparison is important, as the two schemes are confusingly similar. Clearly, a far better image is obtained.
with the proposed algorithm.

We have also tested the proposed generalized algorithm on an interlaced sequence. We used the Foreman sequence and composed each interlaced frame by taking the odd numbered rows from one frame, and the even numbered rows from the next, resulting in a sequence with half as many frames. This sequence was also contaminated by additive white zero-mean Gaussian noise with $STD = 2$. This generated sequence can be considered a true interlaced sequence, as no manipulation (e.g. simulated blurring) of the pixels has been made other than half the pixels being discarded.

This sequence has been processed by the framework suggested in Section 4.3.6, with the result appearing in Figure 4.3. The initial interlaced sequence was split into fields, and each field was expanded by a factor of two in the vertical axis only. The missing rows were interpolated by averaging the rows immediately above and below each missing row. The masks $M_t$ were designed to discard the even rows in the odd numbered images, and the odd rows in the even numbered images. 5 interlaced frames (10 fields) were used for processing, and the search area consisted of 10 pixels in every direction. We display the results for two iterations (where the first is used for computing the weights for the second), although the differences are much less dramatic than in the SRR case. As done above, we
also show the results of directly filtering the re-scaled sequence with the NLM filter, to highlight the difference of the proposed approach. Note how the staircase effect (on the wall) is much decayed by the proposed algorithm. It should be noted that the purpose of this test is only to demonstrate the applicability of the proposed framework to other re-sampling tasks, without claiming that it out-performs other de-interlacing methods. Further work is required to compare the proposed technique to existing de-interlacing algorithms.

![De-Interlacing Results](image)

Figure 4.3: De-Interlacing Results. (a) Original (ground-truth) image. (b) Interlaced image. (c) Row Averaging, 29.87dB. (d) Row Averaging followed by NLM processing, 29.93dB. (e) Proposed algorithm - first iteration, 30.69dB. (f) Proposed algorithm - second iteration, 30.71dB.

Before concluding the results section, we address the computational complexity of the algorithm presented here. As already explained in [102], the overall algorithm is very heavy – the weights’ computation stage is the most demanding. For a nominal case, in which the search area is $31 \times 31$ pixels in the low-resolution, 15 images in the sequence, and a patch size of $13 \times 13$ pixels for computing the weights, there are about 2,400,000 operations per pixel. While this value may seem prohibitive, there are various methods in which this computational burden can be substantially reduced. We refer the reader to [102] (Section 4.3) for an elaborate discussion.
4.5 Summary

In our earlier work we developed an explicit-motion-estimation-free SRR algorithm by extending the NLM [102]. In this chapter we approach the same task from a different perspective, basing it on a probabilistic and crude motion estimation. Interestingly, this approach (under some assumptions) leads to the same algorithm described in [102]. However, the formulation described here is more intuitive, as it relies on the classic super-resolution framework and on the imaging model. Furthermore, this formulation allows for different extensions than those proposed in [102]. We also show how the framework can in fact be adapted to any re-sampling task. An example of de-interlacing is given, to show the validity of this adaptation. This example shows that even sequences with large, highly non-rigid motion patterns can be successfully de-interlaced by the proposed framework.
Chapter 5

Closed-Form MMSE Estimation for Signal Denoising Under Sparse Representation Modeling Over a Unitary Dictionary

The contents of this chapter had been published in the paper M. Protter, I. Yavneh and M. Elad, ”Closed-Form MMSE Estimation for Signal Denoising Under Sparse Representation Modeling Over a Unitary Dictionary”, IEEE Transactions on Signal Processing, Vol. 58, No. 7, Pages 3471-3484, July 2010 [104].

This chapter deals with the Bayesian signal denoising problem, assuming a prior based on a sparse representation modeling over a unitary dictionary. It is well known that the Maximum A-Posteriori Probability (MAP) estimator in such a case has a closed-form solution based on a simple shrinkage. The focus in this chapter is on the better performing and less familiar Minimum-Mean-Squared-Error (MMSE) estimator. We show that this estimator also leads to a simple formula, in the form of a plain recursive expression for evaluating the contribution of every atom in the solution. An extension of the model to real-world signals is also offered, considering heteroscedastic non-zero entries in the representation, and allowing varying probabilities for the chosen atoms and the overall cardinality of the sparse representation. The MAP and MMSE estimators are re-developed for this extended model, again resulting in closed-form simple algorithms. Finally, the superiority of the MMSE estimator is demonstrated both on synthetically generated signals
and on real-world signals (image patches).

5.1 Introduction

One of the most fundamental and extensively studied problems in signal processing is the removal of additive noise, known as denoising. In this task, it is assumed that the measured signal \( y \in \mathbb{R}^n \) is the result of a clean signal \( x \in \mathbb{R}^n \) being contaminated by noise, \( y = x + e \). As is often done, we limit the discussion to zero-mean i.i.d. Gaussian noise.

In order to be able to distinguish the signal from the noise, it is important to characterize the signal family as well. One very successful model, that has attracted attention in recent years, leans on the signal’s sparsity with respect to some transform. In such a model, the signal is assumed to be representable as a linear combination of a few basic signal building blocks known as atoms. Formally put, \( x \) can be represented as \( x = D\alpha \), where \( D \in \mathbb{R}^{n \times m} \) is a known dictionary (set of atoms \( \{d_j\}_{j=1}^n \)) and \( \alpha \) is a sparse vector of coefficients. “Sparse” here means that \( \alpha \) contains a small number (compared to \( n \)) of non-zero coefficients. In general, the dictionary may be redundant, containing more atoms than the dimension of the signal (\( m \geq n \)).

How can this model be used for recovering \( x \) from the measurement \( y \)? A commonly used method (see [14] and references therein) is to seek a signal \( \hat{x} \) that is both sparse with respect to \( D \) (i.e., has a sparse representation) and close enough to the measured signal. This task can be written as seeking the representation \( \hat{\alpha} \) defined by

\[
\hat{\alpha} = \arg\min_\alpha \|\alpha\|_0 + \lambda \|y - D\alpha\|_2^2 ,
\]

where \( \|\alpha\|_0 \) counts the number of non-zeros in \( \alpha \) and \( \lambda \) is a positive parameter. This energy functional contains two terms, the first promoting sparsity of the signal and the second promoting proximity to the measurement. This minimization task can be shown to be related to the Maximum A-Posteriori (MAP) probability estimator [14].

Solving the minimization task is in general NP-hard [86], and therefore approximate solvers are required. One approach can be to replace the \( \ell_0 \)-norm with \( \ell_1 \), leading to a family of algorithms known as Basis Pursuit [22]. Another commonly used approach is a greedy algorithm, such as the Orthogonal Matching Pursuit (OMP) [84, 21, 92]. In this

\footnote{While the discussion in this chapter is over the reals, all the derivations here apply over the complex field just as well.}
algorithm, one atom is selected at each step, such that the norm of the residual (that portion of the signal not yet represented) is best decreased.

While MAP estimation, as manifested above, promotes seeking the single sparsest representation, recent work shows that a better result (in the $L_2$ sense) is possible using the Minimum-Mean-Squared-Error (MMSE) estimator [78, 109, 49, 103]. The MMSE estimator requires a weighted average of all the possible sparse representations that may explain the signal, with weights related to their probability. Just like the MAP in the general setting, this estimation is infeasible to compute, and thus an approximation is proposed. For example, the work reported in [78, 109] offers approximations based on a tree search for candidate solutions with pruning of ones less likely to explain well the signal. Similarly, the work reported in [49] suggests a random version of the OMP for getting several representations, followed by plain averaging.

More broadly, in the realm of sparse representations, mixing several estimators to get a better estimate has been studied in various directions in the past decade. One such direction considers fusion of estimators that use different dictionaries [51, 117]. The machine-learning and the statistics literature offers several recent contributions (see [32, 33, 74] for representative work), where a group of competing estimators are combined (aggregated) using exponential weights, leading to an estimate that goes beyond the best of the group. Clearly, there is a growing interest in Bayesian estimators that go beyond the MAP, and in non-Bayesian techniques that provide an alternative motivation for aggregation of sparse estimators.

In this chapter we focus on the special case where the dictionary $D$ is square ($n = m$) and unitary $D^T D = I$. In such a case, the problem formed in Equation (5.1) need not be approximated, as there is a closed-form non-iterative solution in the form of shrinkage over simple inner products [37, 38, 6, 42]. Furthermore, the OMP becomes exact in such a case. Naturally, these facts make the MAP estimator a very appealing approach for the unitary case.

The question we address in this chapter is the following: Does the MMSE estimator also enjoy a simple closed-form solution for the unitary case? We show that this is indeed the case, and develop a recursive formula that leads to the exact MMSE estimation. We start our treatment with a simple model that assumes that all the non-zero entries in the representation are drawn from the same distribution (i.i.d), and with a fixed and known cardinality. We then present a more general signal model based on a sparse representa-
tion, considering heteroscedastic non-zero entries in the representation, allowing varying probabilities for the chosen atoms, and imposing a probability rule on the cardinality of sparse representation. We extend both the MAP and MMSE estimators to this more complex model, and derive simple and exact algorithms for obtaining these estimators. We test these estimators on both synthetic and real-world signals (image patches) and demonstrate the superior performance of the proposed MMSE estimator in these tests.

We note that a preliminary version of this chapter has appeared in [103], showing the core recursive formula for the MMSE computation for a simple sparse representation modeling, and demonstrating it on elementary synthetic experiments. Here we present a much-extended version of this work that includes (i) a richer model that better fits general content signals; (ii) full development of the MAP and MMSE closed-form solutions for this extended model, and with more details; (iii) a numerical stability analysis of the recursive formula; and (iv) a wider experimental part with new tests on real-world signals, where the parameters of the model are estimated as part of the denoising process.

The structure of the chapter is as follows: In the next section we formulate the denoising problem and review the prior work on the MAP and the MMSE estimators. Section 5.3 derives the closed-form recursive formula for the MMSE estimator for a simple signal model and analyzes its numerical behavior. In section 5.4 we propose an extended generative signal model and develop the MAP and MMSE estimators for it, resulting in simple and exact algorithms for their recovery. We also discuss the need and means to estimate the many free parameters of this model. Section 5.5 presents an empirical study on both synthetic and real-world signals, demonstrating the various algorithms developed, and Section 5.6 presents conclusions.

5.2 Prior Work

In order to deploy the MAP and MMSE estimators for the denoising task, we need to start by defining the signal creation process. The literature on sparse representation modeling, and orthogonal wavelet coefficients in particular, is rich with ideas on how to model signals. A hierarchical Bernoulli-Gaussian mixture is commonly used to model such coefficients, in order to derive the shrinkage to be applied on them [25, 26, 27, 1, 5]. Alternatively, Generalized Gaussians have also been used to model these coefficients [85, 34]. Such models assume independence between these coefficients, which makes the consequent estimation
task easier. In this work we take a different path, and follow closely the source model considered in [78, 109, 49], where a non-uniform prior on the selection of the non-zero coefficients is considered, with a subsequent coupling between the different coefficients.

We assume that \( x = D\alpha \) is generated by first choosing the support of \( \alpha \) (locations of non-zero coefficients), denoted by \( S \), using the probability function \( P(S) \). Following [49] and [103] we shall restrict our treatment for now to the case where all supports with \( |S| = k \) are equally probable, and all the others have zero probability. In Section 5.4 we remove this limiting assumption and extend the analysis to the more general case. We denote this set of permissible supports by \( \Omega_k \). Once \( S \) is chosen, the representation’s non-zeros are formed as a set of \( k \) random i.i.d. entries drawn from the Normal distribution \( N(0, \sigma_x^2) \). As explained above, the signal \( x = D\alpha \) is then contaminated by a random i.i.d. Gaussian noise vector \( e \), resulting in the measured noisy vector \( y = x + e \).

We define the \( |S| \times m \) matrix \( P_S \) that extracts the \( |S| \) non-zero entries from a sparse vector \( \alpha \in \mathbb{R}^m \), i.e., \( P_S\alpha = \alpha_S \in \mathbb{R}^{|S|} \). We further denote by \( D_S = DP_S^T \) the sub-matrix of \( D \) that contains only the columns corresponding to the support. We introduce the following two additional notations for simplicity of later expressions and analysis:

\[
G_S = \frac{1}{\sigma_x^2}D_S^T D_S + \frac{1}{\sigma_e^2}I \quad \text{and} \quad v_S = \frac{1}{\sigma_x^2}D_S^T y.
\]

For the signal model described herein, if the support \( S \) is known, the MMSE estimator for \( x \) (termed the oracle) is obtained by minimizing

\[
J(S) = \frac{1}{\sigma^2_x} \| D_S\alpha_S - y \|^2 + \frac{1}{\sigma_x^2} \| \alpha_S \|^2,
\]

and is given by

\[
\hat{\alpha}_{\text{oracle}} = \left( \frac{1}{\sigma_x^2}D_S^T D_S + \frac{1}{\sigma_e^2}I \right)^{-1} \frac{1}{\sigma_x^2}D_S^T y = G_S^{-1}v_S \quad \text{(5.2)}
\]

This result can easily be obtained by observing that \( P(\alpha_S|y) \) is proportional to \( P(y|\alpha_S)P(\alpha_S) \) (using Bayes’s rule). Due to the Gaussian noise, we have \( P(y|\alpha_S) \propto \exp\{-\|D_S\alpha_S - y\|^2/2\sigma^2\} \). Similarly, the Gaussian distribution of the non-zero entries in \( \alpha_S \) implies \( P(\alpha_S) \propto \exp\{-\|\alpha_S\|^2/2\sigma_x^2\} \). Thus, \( P(\alpha_S|y) \) is a Gaussian distribution, and its mean (or maximum, as the two align) yields the oracle estimation of \( \alpha_S \). A multiplication by \( D_S \) leads to the oracle estimation of the corresponding signal \( x \), as in Equation (5.2).

---

\( ^2 \)We depart from the Zellner g-prior as used in [49]. This prior assumes orthogonalization of the columns of the support as part of the signal generation. See [109, 49] for more details.
As the support in the actual problem is random and unknown, the MMSE estimation becomes an expectation over all possible supports. This is a weighted average of many such “oracles”, $x_S$, as given in Equation (5.2), each considering one possible support. Those are to be weighted by their probability to explain $y$, which leads to

$$\hat{x}_{MMSE} = \sum_{S \in \Omega_k} P(S|y) x_S = \sum_{S \in \Omega_k} P(S|y) D_S G_S^{-1} v_S.$$ (5.3)

It can be shown [49] that, up to a normalization factor, $P(S|y)$ is given by

$$P(S|y) \propto \exp \left\{ \frac{1}{2} v_S^T G_S^{-1} v_S + \frac{1}{2} \log \left( \det (G_S^{-1}) \right) \right\}.$$ (5.4)

Roughly speaking, if we assume that $G_S^{-1}$ is approximately proportional to $I$, this expression suggests that highly probable supports are those with high energy remaining in the projection of $y$ onto $D_S$. For a more elaborate derivation of these terms we refer the interested reader to Equations (7) and (8) in [49].

The MAP estimator is obtained by choosing the support $S$ that maximizes the above probability, $P(S|y)$, and computing the oracle estimation for this support. Both this estimation and the MMSE one require in general a sweep through all $\binom{m}{k}$ supports in $\Omega_k$, which is an infeasible task in general, due to the exponentially growing size of this set. Thus, OMP is used to approximate the MAP by solving an exact MAP estimator for $k = 1$ (one atom), peeling the portion of the signal found, and repeating the process [84].

Similarly, the MMSE needs to be approximated, and several methods have been proposed for this task in recent years. The work in [78, 109] proposes a deterministic process of selecting a small group of well-chosen supports over which to average. Those are found in a greedy fashion, by forming a tree search and pruning less-likely solutions. The Random-OMP algorithm [49] repeats the OMP several times, with a random choice of the next atom, based on $P(S|y)$ for $k = 1$. This yields an approximate Gibbs sampler for this distribution, and thus plain averaging of the representations found leads to a good approximation of the MMSE estimation. It is important to note in this context that the MMSE estimator and the Random-OMP that approximates it, generally do not result in a sparse representation, but they are still better than the MAP (as shown in [49]), even though the original signal is in fact sparse. This property of the estimators results from the aggregation of many (or in fact all) different supports, leading to an equivalent support which is not sparse. For a more detailed discussion of the phenomenon, see [49].

In the unitary case, any subset of columns from $D$ is orthogonal (i.e., $D_S^T D_S = I$), and thus the above expressions can be further simplified. Starting with the matrix $G_S$, it
becomes
\[ G_S = \frac{1}{\sigma^2} D_S^T D_S + \frac{1}{\sigma_S^2} I = \left( \frac{1}{\sigma^2} + \frac{1}{\sigma_S^2} \right) I = \frac{\sigma^2 + \sigma_S^2}{\sigma^2 \sigma_S^2} I. \] (5.5)

Similarly, the weights \( P(S|y) \) become
\[
P(S|y) \propto \exp \left\{ \frac{1}{2} v_S^T G_S^{-1} v_S + \frac{1}{2} \log \left( \det \left( G_S^{-1} \right) \right) \right\}
\]
\[
\propto \exp \left\{ \frac{1}{2 \sigma^2} \cdot \frac{\sigma_S^2}{\sigma^2 + \sigma_S^2} ||D_S^T y||^2_2 \right\}.
\] (5.6)

Note that the log-factor has been removed as it is equal for all the supports in \( \Omega_k \). Furthermore, this probability is computed only up to a factor which equals \( 1/P(y) \). Instead of computing it directly, we use the fact that the sum of probabilities must equal to 1 in order to normalize the probabilities correctly.

Equation 5.6 clarifies that the MAP support is the one that maximizes \( ||D_S^T y||^2 \), and is easily found by computing \( D_S^T y \), sorting the resulting vector by (absolute) size, and choosing the first \( k \) elements. Thus, MAP for this case can be computed exactly with a simple algorithm. Furthermore, OMP in such a case is also exact, as the sequential detection of the largest inner product leads to the same outcome.

Naturally, we should wonder whether the unitary case offers such a simple and closed-form solution for the MMSE, which bypasses the need for the above described approximations (e.g., the Random-OMP). This is the topic of the next two sections.

5.3 The Case of a Unitary Dictionary

5.3.1 MMSE Over a Unitary Dictionary - Fundamentals

The development in this section follows the one in [103] with important modifications to make the derivation clearer, more precise, and more general. Our goal is to show that for a unitary dictionary \( D \), the MMSE estimation can be computed exactly (up to rounding errors) while avoiding combinatorial computations. Recall that for a unitary matrix \( D \), we have
\[ ||D\alpha - y||^2_2 = ||\alpha - D^T y||^2_2 = \sum_{i=1}^{n} (\alpha_i - \beta_i)^2, \]
where \( \beta = D^T y \) and \( \beta_i \) the \( i^{th} \) entry of \( \beta \). This will be helpful in later derivations.

The MMSE estimation in Equation (5.3) can be read differently. Every possible support in the summation provides a candidate representation vector \( \alpha_S = G_S^{-1} v_S \in \mathbb{R}^k \).
Multiplication of the form $P^T_S \alpha_S$ provides a sparse vector of length $m$ that contains the entries of $\alpha_S$ as its non-zeros. Thus, the MMSE estimator is given by

$$\hat{x}_{\text{MMSE}} = \sum_{S \in \Omega_k} P(S|y)D_S \alpha_S = D \cdot \sum_{S \in \Omega_k} P(S|y)P^T_S G^{-1}_S v_S.$$  \hfill (5.7)

Here we have used the relation $D_S = DP^T_S$, and thus the multiplication by $D$ is performed outside the summation. This expression suggests that there is one effective representation that governs the estimated outcome, given (removing the multiplication by $D$) by

$$\hat{\alpha}_{\text{MMSE}} = \sum_{S \in \Omega_k} P(S|y)P^T_S \alpha_S = \sum_{S \in \Omega_k} P(S|y)P^T_S G^{-1}_S v_S.$$  

This implies that every one of the $n$ (recall that $m = n$) atoms contributes a pre-specified portion to the overall MMSE estimation. We shall exploit the fact that the matrix $D$ is unitary, and construct a closed-form formula for these $n$ contributions, thus turning this estimator into a practical algorithm.

Denote $c^2 = \sigma^2_x / (\sigma^2 + \sigma^2_x)$. Returning to Equation (5.6), we observe that in the unitary case,

$$P(S|y) \propto \exp \left\{ \frac{c^2 \| D^T_S y \|^2}{2\sigma^2} \right\} = \prod_{i \in S} \exp \left\{ \frac{c^2 \beta_i^2}{2\sigma^2} \right\} \propto \prod_{i \in S} q_i,$$

where we have denoted

$$q_i = \frac{\exp(c^2 \beta_i^2 / 2\sigma^2)}{\sum_{j=1}^n \exp(c^2 \beta_j^2 / 2\sigma^2)}.$$  \hfill (5.8)

Thus we have

$$P(S|y) = A_k \prod_{j \in S} q_j,$$

where $A_k$ is a normalizing constant yielding $\sum_{S \in \Omega_k} P(S|y) = 1$. Note that for $k = 1$, the probability of the support being the $j^{th}$ atom is simply $P(S = \{j\}|y) = q_j$, hence $A_1 = 1$, since the $q_j$’s are properly normalized. Now we can obtain a simpler formulation for the MMSE estimator. Using the notations of $c^2$ and $\beta$, we can write $G^{-1}_S = \sigma^2 c^2 I$ (from Equation (5.5)) and $v_S = P_S \beta / \sigma^2$. Assigning these and the formula for $P(S|y)$ into Equation (5.7), we get that

$$\hat{x}_{\text{MMSE}} = \sum_{S \in \Omega_k} P(S|y) \cdot D_S G^{-1}_S v_S$$

$$= \sum_{S \in \Omega_k} \left[ A_k \left( \prod_{i \in S} q_i \right) D_S \left( \sigma^2 c^2 I \right) \left( \frac{1}{\sigma^2} P_S \beta \right) \right]$$

\hfill (5.9)
\[ = c^2 A_k \sum_{S \in \Omega_k} \left[ \left( \prod_{i \in S} q_i \right) \cdot (D_S P_S \beta) \right] \]
\[ = c^2 A_k \sum_{S \in \Omega_k} \left[ \left( \prod_{i \in S} q_i \right) \cdot \left( \sum_{i \in S} \beta_i d_i \right) \right] . \]

Computing this formula in a straightforward manner requires a prohibitive \( O(n^k) \) operations, as every group of \( k = |S| \) atoms has to be considered and summed. In order to simplify this expression, we introduce the indicator function
\[
I_S(i) = \begin{cases} 
1 & i \in S \\
0 & i \notin S 
\end{cases},
\]
and rewrite Equation (5.9) as
\[
\hat{x}_{MMSE} = c^2 A_k \sum_{S \in \Omega_k} \left[ \left( \prod_{j \in S} q_j \right) \left( \sum_{i=1}^{n} \beta_i d_i I_S(i) \right) \right].
\]
Rearranging the order of summations and multiplications in this equation yields the equivalent expression,
\[
\hat{x}_{MMSE} = c^2 A_k \sum_{i=1}^{n} \left[ \left( \sum_{S \in \Omega_k} I_S(i) \left( \prod_{j \in S} q_j \right) \right) \beta_i d_i \right] = c^2 \sum_{i=1}^{n} q_k^i \beta_i d_i, \tag{5.10}
\]
where we have introduced the notation
\[
q_k^i = A_k \sum_{S \in \Omega_k} I_S(i) \left( \prod_{j \in S} q_j \right).
\tag{5.11}
\]
The straightforward way to compute this scalar value would be by sweeping through all supports \( S \) in \( \Omega_k \) that contain the \( i \)th atom (there are \( \binom{n-1}{k-1} \) of those), computing for each of them \( A_k \prod_{j \in S} q_j \), (i.e., \( P(S|y) \)), and summing these up. Thus, \( q_k^i \) is nothing but the probability that atom \( i \) will be included in the support. This computation is still exponential and thus prohibitive, but, as we show next, an efficient recursive formula for these values is within reach. Note that, using this notation, the MMSE estimator can be written as
\[
\hat{x}_{MMSE} = c^2 D \operatorname{diag}(q^k) \beta, \text{ where } q^k \text{ is a vector of length } n \text{ comprised of the probabilities } \{q_k^i\}_{i=1}^{n}, \text{ and } \operatorname{diag}(v) \text{ is a diagonal matrix containing the values of } v \text{ along its main diagonal.}
\]

### 5.3.2 Obtaining a Closed-Form MMSE Formula

We proceed towards our goal of a closed-form formula by considering by way of analogy the following game. Suppose that \( k \) balls are tossed independently at a group of \( n \) buckets
of various sizes. Suppose that, if we were to toss a single ball, the probability that it would land in bucket \( i \) would be \( q_i \) (with \( 0 < q_i < 1 \) for all \( i \), and \( \sum_i q_i = 1 \), that is, the ball always lands in some bucket.) This “round” of \( k \) tosses is repeated over and over again. If the \( k \) balls fall into \( k \) different buckets in a given round, this round is declared valid and this \( k \)-tuple of buckets is tallied. However, if two or more balls fall into any single bucket in a given round of \( k \) tosses, the round is void and nothing is tallied. The task is to calculate the \( q^k_i \)—the probability that some ball will fall into bucket \( i \) in a valid round of \( k \) tosses—for \( i = 1, \ldots, n \).

Why is this game relevant? A valid round consists of \( k \) independent tosses landing in \( k \) different buckets, and therefore the probability of any particular \( k \)-tuple of buckets is clearly proportional to the product of its \( q_i \)'s. The probability of each bucket participating in the \( k \)-tuple is therefore the sum of probabilities of the \( k \)-tuples \((S)\) that contain it, analogously to (5.11). Based on this analogy, we make the following observations, which will be useful later:

- **Base:** For \( k = 1 \) (only a single toss) we get \( q^1 = q \), the vector whose elements are \( q_i \) (the individual probabilities of each bucket), as defined in Equation (5.8).

- **Bounds on \( q^k \):** Since every bucket has a nonzero probability, and at most participates in all tuples, we have \( 0 < q^k \leq 1 \) element-wise for \( k = 1, \ldots, n - 1 \), and \( q^n = 1 \), where \( 0 \) and \( 1 \) are the \( n \)-vectors of all zeros and all ones, respectively.

- **Preservation of order:** If \( q_j \geq q_i \) then \( q^k_j \geq q^k_i \) for \( k = 1, \ldots, n \), with equality occurring if and only if \( q_j = q_i \) or \( k = n \). That is, a more likely bucket (with greater probability of being hit in a single toss) remains more likely as we increase the number of balls per round.

- **Monotonicity in \( k \):** For \( k = 2, \ldots, n \), \( q^k > q^{k-1} \) element-wise, because increasing the number of balls increases the probability of every one of the buckets.

- **Monotonicity of ratios in \( k \):** If \( q^k_j > q^k_i \) then \( q^{k-1}_j/q^{k-1}_i < q^{k-1}_j/q^{k-1}_i \), for \( k = 2, \ldots, n \).

This claim is non-trivial, and its proof is given in Appendix 5.A.\(^3\)

\(^3\)A particular implication of this property is that it shows that the Random-OMP algorithm [49] remains inexact, even if given an infinite number of iterations to run. This is because the Random-OMP selects the atoms with probabilities according to the initial ratios \( q^1_j/q^1_i \), while those ratios should decrease as \( k \) increases. This also hints that the inexactness of the Random-OMP increases with \( k \).
• **Symmetry:** Assume henceforth that one of the balls (only) is colored red. Since the color has no effect of any significance, the probability that the red ball will fall into bucket \( i \) in a given round is clearly equal to the probability that any one of the other \( k - 1 \) balls will fall into this bucket.

• **Normalization:** The vectors \( q^k \) satisfy the normalization condition \( \sum_{i=1}^{n} q_i^k = k \), that is, the sum of probabilities of all buckets is equal to the number of balls per round. This allows us to determine the \( A_k \)'s. This property is implied by the Symmetry property, by which the probability that the red ball will fall into bucket \( i \) in a valid round is \( q_i^k / k \). Since the overall probability that the red ball will fall into some bucket in a valid round is 1, we have \( \sum_{i=1}^{n} q_i^k / k = 1 \), from which the Normalization property follows.

We next derive the recursive formula for computing \( q^k \). For \( k = 1 \) we have \( q^1 = q \) by the Base property. For \( j = 2, \ldots, k \), we have that \( q_j^i \) is proportional to the probability that the red ball will fall into bucket \( i \) (which is \( q_i \)) times the probability that the remaining balls will comprise a valid round of \( j - 1 \) balls that does not include bucket \( i \) (which is \( 1 - q_{i^{j-1}} \)). This product needs to be normalized so as to satisfy the Normalization property, yielding

\[
q_j^i = j \frac{q_i (1 - q_{i^{j-1}})}{1 - \sum_{\ell=1}^{n} q_{\ell} q_{\ell^{j-1}}}. \tag{5.12}
\]

The full vector of probabilities is thus given by

\[
q^j = F^{j} (q^{j-1}) \equiv j \frac{\text{diag}(q)(1 - q^{j-1})}{1 - q^{j} q^{j-1}}. \tag{5.13}
\]

with \( q \) given in Equation (5.8):

\[
q_i = \exp\left(\frac{c^2 \beta_i^2}{2\sigma^2}\right) \sum_{j=1}^{n} \exp\left(\frac{c^2 \beta_j^2}{2\sigma^2}\right).
\]

### 5.3.3 Numerical Instability

Unfortunately, the recursive formula (5.13) tends to suffer from instability, manifest in a fast growth of numerical errors during the iterations when \( k \) is not small. To study this effect, we perform a linear stability analysis. Suppose that \( q^{j-1} \) contains an error (vector), \( \delta^{j-1} \). Then, ignoring the (typically machine-accuracy, hence negligible) numerical errors in \( q \) and in the arithmetic operations of (5.13), we obtain by taking the first term of the Taylor series of \( F^j \), given by

\[
\delta^j \approx C^j \delta^{j-1}, \tag{5.14}
\]

59
where $\mathbf{C}^j = \partial F^j (\mathbf{q}^{j-1})$ is the gradient matrix of $F^j$, which can be computed easily from (5.12). After rearrangement, the elements of $\mathbf{C}^j$ can be written as

$$C_{\ell,m}^j = \begin{cases} -\frac{(j - q_1^j)}{j \left(1 - q_1^{j-1}\right)} q_1^j & \text{if } \ell = m, \\ \frac{\left(q_1^j\right)^2 q_m}{j \left(1 - q_1^{j-1}\right) q_1^j} & \text{otherwise}. \end{cases}$$  \hspace{1cm} (5.15)$$

The error propagation per iteration is determined by the spectral properties of $\mathbf{C}^j$. These are hard to compute in general, but we can clearly see the source of the numerical trouble by considering a special case where two elements of $\mathbf{q}$ happen to be exactly the same. Without loss of generality, assume that these are the first two elements, i.e., $q_1 = q_2$, and therefore, by the Preservation of order property, $q_1^j = q_2^j$ for all $j$. For all $j$ we then have by (5.15):

$$C_{1,1}^j = C_{2,2}^j = -\frac{(j - q_1^j)}{j \left(1 - q_1^{j-1}\right)} q_1^j \quad \text{and} \quad C_{2,1}^j = C_{1,2}^j = \frac{\left(q_1^j\right)^2}{j \left(1 - q_1^{j-1}\right)},$$

$$C_{\ell,1}^j = C_{\ell,2}^j \quad \text{for all } \ell > 2 \quad \text{and} \quad C_{1,m}^j = C_{2,m}^j \quad \text{for all } m > 2.$$  

It is now immediate to verify that the vector of size $n$ given by $\mathbf{v} = (1, -1, 0, 0, \ldots, 0)^T$ is an eigenvector of $\mathbf{C}^j$ for all $j$, with eigenvalue given by

$$\lambda_j = C_{1,1}^j - C_{1,2}^j = -\frac{q_1^j}{1 - q_1^{j-1}}.$$  

For $q_1^j < 0.5$, we get $|\lambda_j| < 1$, and the iteration is stable with respect to errors of the form $\mathbf{v}$. However, by the Monotonicity property, $q_1^j$ grows with $j$, eventually reaching 1 at $j = n$. Once $q_1^{j-1}$ crosses 0.5, an oscillating (since $\lambda_j < -1$) pair-wise antisymmetric divergence of the error kicks in, with the divergence rate growing with each iteration, because $|\lambda_j|$ grows with $j$. A key feature here is that the eigenvector $\mathbf{v}$ is shared by all the $\mathbf{C}^j$’s, so it grows in absolute value at each iteration (once $\lambda_j < -1$).

Although this analysis assumes a pair of equal elements, the unstable behavior it implies is quite general. Nevertheless, the instability can largely be kept at bay by enforcing the known constraints implied by the properties above on solutions obtained from the recursive formula. Imposing these constraints at each iteration of the recursive formula is a relatively cheap method of keeping the numerical errors under control. Furthermore,
if at stage $j$ during the calculation of the formula it is determined that one (or more) probability $q_j^k$ attains a value sufficiently close to 1 (which also means that it is a source of numerical instability, cf., discussion above), we can set this value to 1 at all subsequent iterations due to the Monotonicity property. To improve the numerical accuracy for the rest of the entries, we may eliminate this element of $q$ and then recalculate $q^{k-1}$ for the remaining entries.

5.4 Extending the Model to Real-World Signals

The model we have relied on so far has simplified the analysis and the derivation of the MAP and MMSE estimators. However, this model is far too limited for handling real-world signals. More specifically, we have relied on three assumptions that we cannot generally make:

- All coefficients in the support are assumed to be drawn according to the same normal distribution with the same variance $\sigma_x^2$.
- The size of the support $|S|$ is fixed and known.
- Given that $|S|$ is known, $P(S)$ is equal for all supports of this size, and hence all atoms are (a priori) equally likely to be selected.

Unfortunately, these assumptions are too simplistic for faithfully describing real-world signals (such as image patches), and thus cannot function as a good prior signal model for denoising. In order to construct a model fitting real-world signals, these assumptions must be relaxed and generalized, and the formulas for the MAP and MMSE estimators must be adapted accordingly. The assumption regarding the equal distribution of the coefficients is the first we choose to tackle. We relax the remaining two assumptions together by proposing a general signal generation model. The resulting model is general enough to describe a wide range of signals, and can be successfully harnessed for image denoising, as will be shown in Section 5.5. We now describe in detail the required extensions and adaptations, which are then summarized in Table 5.1 at the end of this section.

5.4.1 Treating a Heteroscedastic Coefficient Set

Previously it was assumed that all coefficients share the same prior variance $\sigma_x^2$. Assuming that all coefficients behave identically is unrealistic, so we now allow the variance to be
atom-dependent and denote it by $\sigma_i^2$. Accordingly, we define $c_i^2 = \sigma_i^2/(\sigma_i^2 + \sigma^2)$, which also becomes atom dependent. The oracle in the unitary case becomes

$$\hat{x}_{oracle} = \sum_{i \in S} c_i^2 \beta_i d_i.$$  \hfill (5.16)

This is easily verified following the explanation given in Section 5.2 for the derivation of the oracle formula in the general case. Using the fact that for the non-zero portion of $\alpha$ we now have $P(\alpha_S) \propto \exp\{-\sum_{i=1}^n \alpha_i^2/2\sigma_i^2\}$, we observe that the posterior probability $P(\alpha_S|y)$ is Gaussian, and the expression given in (5.16) is its mean.

A second effect of the different variances per atom appears in the posterior probability $P(S|y)$. Using Equations (5.4) and (5.6) one notices that the log-factor cannot be discarded, and this expression becomes

$$P(S|y) \propto \prod_{i \in S} \exp\left\{\frac{c_i^2 \beta_i^2}{2\sigma^2} + \frac{\log(c_i^2)}{2}\right\} \propto \prod_{i \in S} q_i,$$  \hfill (5.17)

which implies a somewhat modified definition for $q_i$.

The MAP estimator selects the $k$ atoms with the largest $q_i$ values and projects onto them using the oracle formula in Equation (5.16). The MMSE estimator uses a formula very similar to the one introduced before in Equation (5.10),

$$\hat{x}_{MMSE} = A_k \sum_{i=1}^n \left[\left(\sum_{S \in \Omega_k} I_S(i) \left(\prod_{j \in S} q_j\right)\right) c_i^2 \beta_i d_i\right] = \sum_{i=1}^n q_i^k c_i^2 \beta_i d_i,$$  \hfill (5.18)

with the two changes being the redefinition of $q_i$ and the atom-dependent value $c_i$ replacing the constant $c$. Interestingly, the recursive formula for the update of $q_i^k$ remains the same as in Equation (5.12), as do the constraints that are employed in stabilizing its numerical evaluation.

### 5.4.2 Extending the Signal Generation Model

The assumption that only a specific cardinality exists, and moreover, that all supports of this cardinality are equally likely, is unrealistic. For example, smooth and slowly varying signals may have a very sparse representation, while highly textured signals may require many more atoms for an adequate representation. Furthermore, some atoms are expected to appear more frequently than others, increasing the probability of some supports and reducing the probability of other supports. These observations lead to the generative signal model we now consider.
Assume that the size of the support is chosen randomly according to a known probability $P_C(k) = P(|S| = k)$, thus relaxing the fixed support size constraint introduced in section 5.2. Then, $k$ atoms are chosen sequentially, where atom $i$ has a probability $P_a^i$ of being selected (normalized such that $\sum_i P_a^i = 1$). If the resulting group consists of $k$ distinct atoms, this support draw is considered valid, otherwise (that is, in the event of at least one repetition) it is discarded and the random atom selection process is restarted. Lastly, the active coefficients for the selected support are drawn at random from the distributions $\mathcal{N}(0, \sigma_i^2)$, as before.

In order to adapt the estimators to this more general model, we should update the definition of $P(S)$ to reflect the new signal model. The probability of a specific support to be chosen is proportional to the probability of the size of the support multiplied by the individual probabilities of the atoms to be chosen $P(S) \propto P_C(|S|) \cdot \prod_{i \in S} P_a^i$, with a normalization such that for every $k$, $\sum_{S \in \Omega_k} P(S) = P_C(k)$. This implies that the choice of atoms is independent of the choice of support size. Denoting $T_k = \sum_{S \in \Omega_k} \prod_{i \in S} P_a^i$, the probability of a specific support $S$ is given by

$$P(S) = \frac{P_C(|S|)}{T_{|S|}} \cdot \prod_{i \in S} P_a^i. \quad (5.19)$$

The formula for the normalization factor $T_k$ is reminiscent of the formula for $q_k^i$ given in equation (5.11). Indeed, in order to compute $T_k$ we need to apply the recursive formula on the values $\{P_a^i\}_{i=1}^n$, and for each $k$, sum the resulting values (after undoing the numerically stabilizing normalization), and divide by $k$ (as each possible support contributes to $k$ entries). Note that in the general case, in which the signal model is to be applied to a large set of signals, this procedure is needed only once, as it is a property of the model and does not depend on the specific signal.

Using the a priori probability of each support, the overall posterior probability of each support becomes

$$P(S|y) \propto P(y|S) \cdot P(S) \propto \frac{P_C(|S|)}{T_{|S|}} \cdot \prod_{i \in S} (q_i \cdot P_a^i) = \frac{P_C(|S|)}{T_{|S|}} \cdot \prod_{i \in S} \tilde{q}_i, \quad (5.20)$$

with $\tilde{q}_i = q_i \cdot P_a^i$, and $q_i$ taken from Equation (5.17).

The MAP estimator for this more general model is simply the one that maximizes the probability given in Equation (5.20). Recovering it starts by computing $\tilde{q}_i$ for each atom. Then, at each step, one atom is added to the current representation, in descending order of magnitude of $\tilde{q}_i$, and the relative posterior probability of this support is computed
according to Equation (5.20). Of the \( n \) supports generated in this procedure, the likeliest one (which is also the likeliest over all supports) is selected, and by computing the oracle for this support, the MAP estimator emerges. Note that the value computed by Equation (5.20) is not normalized, and therefore it does not represent a true probability. This has no effect on the MAP estimator, however, as we seek the support with the largest probability, and the order is not changed by the lack of normalization.

For the MMSE estimator, all cardinalities with their appropriate probabilities must be considered. Going back to Equation (5.9), this translates into

\[
\hat{x}_{MMSE} = \sum_S P(S|y) \sum_{i \in S} c_i^2 \beta_i d_i = \sum_k \sum_{S \in \Omega_k} P(S|y) \cdot \sum_{i \in S} c_i^2 \beta_i d_i
\]

\[
= \sum_k \frac{P_C(k)}{T_k} \sum_{S \in \Omega_k} \prod_{i \in S} \tilde{q}_i \cdot \sum_{i \in S} c_i^2 \beta_i d_i.
\]

The summation over \( S \in \Omega_k \) is exactly the same as was developed for the single cardinality case in Equation (5.18), with the slight re-definition of \( \tilde{q}_i \) instead of \( q_i \). Therefore, the recursive formula developed in the previous section can be used to obtain the MMSE estimator, by obtaining the MMSE estimate for each support size, and merging them with appropriate weights.

Some care is required, however, in the application of the recursive formula. In the single cardinality case, the various stabilizing normalizations could be ignored, as a normalization by the sum of weights was to be applied in the end. When applying the formula to this more general model, this normalization must be tracked and then undone, in order to properly reflect the relative weights of the different cardinalities.

### 5.4.3 Model Summary and Parameter Estimation

The generative model and its estimators are summarized in table 5.1. There are several parameters that govern the behavior of the model, and those are assumed to be known for the estimation task to complete. The model requires explicit and a priori knowledge of the variances \( \{\sigma_i\}_{i=1}^n \) per atom, the prior probability of each support size \( \{P_C(k)\}_{k=1}^n \) and the prior probability of each atom to be chosen \( \{P_a\}_{i=1}^n \).

This model has a large number of parameters, and when applying the MAP and MMSE estimators to real-world signals, these are unknown to begin with. Therefore, some method of estimating these parameters must be used, or else, the estimators are rendered useless\(^4\).

\(^4\)Note that \( \sigma \), which characterizes the noise, is not part of these parameters, and in this work it is assumed as known.
One approach can be to use a set of high-quality (i.e., almost noise-free) signals in order to learn the parameters, and then apply the estimators based on these parameters. This approach, however, assumes that different images share the same parameter set.

An alternative method is to estimate these parameters from the noisy data directly. We assume that when facing a denoising task, many noisy signal instances are to be denoised together. For example, for image denoising, as will be the case in the experimental section, each $8 \times 8$ patch extracted from the image is considered as one noisy signal. Taking all these signals together, we may ask what would be the best set of parameters that describe these signals (taking into account that they are also noisy).

From the noisy signals, a direct Maximum Likelihood (ML) approach can be undertaken to find the most likely set of parameters to have generated the noisy signals. Unfortunately, the maximization task obtained is quite complex. Instead, we adopt a block-coordinate-descent-like approach, where the signals are first denoised by a parameter-less method (hard-thresholding, which is equivalent to a specific set of parameters that includes equal $P_a$ and $\sigma_i$ for all atoms), and from the cleaner signals we estimate the parameters using an ML formulation, which is built on clean-data. This approach of pre-denoising has been suggested elsewhere, such as in [29]. In principle, this method should be iterated, updating the parameters after the denoising. However, we found that one such iteration is sufficient to get a reliable set of estimates for the parameters, and this is indeed the way we operate in subsequent experiments. Therefore, the only manually set parameter is the parameter that controls the initial denoising, the threshold under which coefficients are considered to be zeros. More details on the parameter estimation process are given in Appendix 5.B.

5.5 Experimental Results

We now proceed to demonstrate the proposed exact MMSE estimator and its superiority over the MAP. We also present one possible approximation of the MMSE, the Random-OMP algorithm [49], to illustrate the gain achieved by using the closed-form solution proposed. Our tests are performed first on synthetic signals, where the model parameters are known and are used by the estimators. We also introduce tests on real-world signals (image patches), where the parameters are unknown and therefore the estimation of the model parameters is required for the estimators as part of the overall treatment.
General Signal Generation Model

Parameters of the Model

1. $P_C(k)$: the probability of each support size $P_C(k) = P(|S| = k)$.
2. $\{P^n_i\}_{i=1}^n$: the probability of each atom to be chosen at each step.
3. $\sigma^2_i$: the variance of the coefficients for each atom.

Signal Generation Model

1. The support size $k = |S|$ is chosen according to $P_C(k)$.
2. Repeat until $k$ unique atoms are chosen: Select $k$ atoms sequentially, each time selecting one according to $\{P^n_i\}_{i=1}^n$.
3. For the atoms selected $i \in S$, draw coefficients $\alpha_i \sim N(0, \sigma^2_i)$.
4. Construct the signal $x = \sum_{i \in S} \alpha_i d_i$.

MAP Estimator

$$q_i = \frac{\exp(c^2 \beta_i^2 / 2\sigma^2)}{\sum_{j=1}^n \exp(c^2 \beta_j^2 / 2\sigma^2)} \text{ for } i = 1, 2, \ldots, n$$
$$\tilde{q}_i = q_i \cdot P^a_i \text{ for } i = 1, 2, \ldots, n$$
$$T_k = \sum_{S \in \Omega_k} \prod_{i \in S} P^n_i \text{ for } k = 1, 2, \ldots, n$$
$$\hat{S}_{MAP} = \arg \max_S \frac{P_C(|S|)}{T_1} \prod_{i \in S} \tilde{q}_i$$
$$\hat{x}_{MAP} = \sum_{i \in \hat{S}_{MAP}} c^2_i \beta_i d_i$$

MMSE Estimator

$$\hat{x}_{MMSE} = \sum_{k=1}^n \frac{P_C(|S|) \sum_{i=1}^n q^k_i c^2_i \beta_i d_i}{T_k}$$
$$q^k_i = k \frac{\tilde{q}_i (1 - q^{k-1}_i)}{1 - \sum_{\ell=1}^n \tilde{q}_\ell q^{k-1}_\ell}$$
$$q^1_i = q_i$$

Table 5.1: The general signal generation model, and the implied MAP and MMSE estimators
5.5.1 Synthetic Experiments

When performing synthetic tests, we have complete control over the signal generation process and its parameters. Since the parameters are known, as well as the standard deviation of the noise, their exact values are given to the estimators in order to check the estimators’ performance in “optimal” settings. The dictionary used in all of the synthetic tests is generated randomly and then orthogonalized, to create a random unitary dictionary. We start by focusing on the simplest model (introduced in Section 5.3), in which the support size \( k \) is known and fixed, and all supports are equally likely. Generating a signal according to this model is done by randomly choosing a set of \( k \) unique atoms, using a uniform probability over all \( \binom{n}{k} \) possibilities. For the selected atoms, coefficients \( \alpha_i \) are drawn independently from a Normal distribution \( \mathcal{N}(0, \sigma_x^2) \). The resulting sparse vector of coefficients is multiplied by the dictionary to obtain the ground-truth signal.

Each entry is independently contaminated by white Gaussian noise \( \mathcal{N}(0, \sigma_x^2) \) to create the input signal (note that due to \( \mathbf{D} \) being unitary, this is equivalent to contaminating the coefficients themselves with additive white Gaussian noise with the same parameters).

For all tests, the dimension of the signals is \( n = 64 \).

The noisy signal is denoised by several methods: (i) MAP estimator, (ii) Random-OMP that approximates the MMSE [49] (averaging 20 representations) ; (iii) An exact and exhaustive MMSE using Equation (5.3) (the complexity of this estimator is exponential in \( k \)); (iv) The recursive MMSE formula; and (v) An oracle that knows the exact support. This process is repeated for 1000 signals, and the mean \( L_2 \) error is averaged over all signals to obtain an estimate of the expected quality of each estimator. The denoising effect is quantified by the Relative-Mean-Squared-Error (RMSE), which is obtained by dividing the MSE of each sample by the standard deviation of the noise, averaged over all signals. The RMSE reflects exactly the ratio between the noise energies in the reconstructed image and the initial one (e.g., an RMSE of 0.1 implies that the noise has been attenuated by a factor of 10).

In order to test the performance of these estimators under different noise conditions, several such tests are run, with \( \sigma_x = 1 \) kept constant in all tests, and the noise level \( \sigma \) varying in the range \( 0.1 – 2 \). This is sufficient, since the important parameter is the ratio \( \sigma_x/\sigma \), and not their individual absolute values. Figure 5.1 shows the denoising effect achieved by each method, when \( |S| = 5 \).

Next, we slightly generalize the generation model, by keeping the support size fixed
\(|S| = 5\) as before, but using a heteroscedastic coefficient set (where \(\sigma_i\) are linearly spaced in the range \(0.5 - 2\)), and allowing each atom to have a different probability to appear \((P_{i}^a = 0.5^i\), normalized to 1 and randomly assigned to the atoms\). The result of such a test appears in Figure 5.2. It is apparent that there is quite a big gap in performance between the MMSE estimator and its approximation via the Random-OMP, demonstrating the importance of the closed-form formula presented here.

The same test, but when the signals are not very sparse \((|S| = 20)\) is displayed in Figure 5.3, showing similar behavior. This test does not feature the exhaustive MMSE, due to its exponential complexity. In the last synthetic test, we apply the most general model, where the probability of each cardinality is given by \(P_C(|S|) = 0.8^{|S|}, |S| = 1,\ldots, 5\) (normalized to sum to 1), with \(\sigma_i\) and \(P_{i}^a\) as in the previous test. The results for this test appears in Figure 5.4, and are an average over 5 different random assignments (which all yield similar results). This test does not include the Random-OMP estimator, which was originally developed only for the fixed support size scenario. As the focus of the chapter is the exact estimator, we chose to avoid extending the approximate (and inferior) Random-OMP to the most general signal model.

To better understand the differences between the estimators, we show in Figure 5.5 the effective representation achieved by each method (for \(|S| = 3\)), for one example signal. The MAP estimator selects the wrong atoms, due to the relatively strong noise \((\sigma = 0.6)\).

![Figure 5.1: Relative denoising achieved (compared to the noisy signal), averaged over a 1000 signals, by several methods, for different noise amplitudes and \(|S| = 5\).](image-url)
Figure 5.2: Relative denoising achieved (compared to the noisy signal), averaged over a 1000 signals, by several methods, for different noise amplitudes, for $|S| = 5$, a heteroscedastic coefficient set and different probabilities for each atom.

Figure 5.3: Relative denoising achieved (compared to the noisy signal), averaged over a 1000 signals, by several methods, for different noise amplitudes, for $|S| = 20$, a heteroscedastic coefficient set and different probabilities for each atom.
Figure 5.4: Relative denoising achieved (compared to the noisy signal), averaged over a 1000 signals, by several methods, for the most general signal model (averaged over 5 different random assignments of atom probabilities).

Figure 5.5: The effective representation achieved by different methods for one example signal, with noise standard deviation $\sigma = 0.6$. 
5.5.2 Real-World Signals

In order to present experiments on real-world signals, we use $8 \times 8$ image-patches drawn without overlap from an image, to which white Gaussian noise has been added. These are selected to compose the real-world data-set for our experiments. The unitary dictionary for these experiments is the Discrete Cosine Transform (DCT) dictionary, which is known to serve natural image content adequately (i.e., sparsely).

It is important to note that there is no attempt to compare the estimators to the state-of-the-art in image denoising. This is because our building blocks—such as a non-adaptive and unitary dictionary—are too limited for this comparison to be fair. Our goal is to demonstrate the superior performance of the MMSE estimator, and to offer the possibility that incorporating it into more complex denoising mechanisms may indeed improve denoising results.

Unlike the synthetic experiments detailed in the previous section, when working on real-world images the various parameters of the model are unknown, and must be estimated from the data. We note that we assume the noise variance $\sigma^2$ is known (or estimated using other methods), and the values of the parameters of the signal generation model are estimated from the noisy data, as detailed in Section 5.4 and in Appendix 5.B. Only one parameter is to be set by the user, and that is the parameter controlling the hard-thresholding in the initial denoising that is used to estimate the parameters.

The test set for the experiments includes 7 different images ($15^{th}$ frame from “garden”, “tennis” and “mobile” sequences, and the images “Barbara”, “boat”, “fingerprint” and “peppers”), and various noise-levels: $\sigma = 10, 15, 20, 25, 30, 40, 50, 75$, (which are equivalent to $\text{PSNR}^5$ of $28.12\text{dB}$, $24.64\text{dB}$, $22.10\text{dB}$, $20.16\text{dB}$, $18.60\text{dB}$, $16.11\text{dB}$, $14.14\text{dB}$ and $10.61\text{dB}$, respectively), with pixel values in the range $[0, 255]$. The average (over all images) improvement in $\text{PSNR}$ of the cleaned image compared to the noisy image appears in Figure 5.6. The MMSE estimator outperforms the MAP estimator by about $0.5\text{dB}$ on average, with this gap being fairly consistent over the different images. In order to highlight the gap between the MMSE and the MAP, Figure 5.7 displays the advantage in $\text{PSNR}$ of the MMSE estimator over the MAP estimator. The error bars in this figure indicate one standard deviation of this gap. Comparisons using the Structural Similarity Index (SSIM) [129] were also carried out, displaying similar behaviour - a slight advantage

\[\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\sum_{n=1}^{p}(\hat{X}(n) - X(n))^2}\right)\text{[dB]},\]

where $\hat{X}$ and $X$ are the clean and the reconstructed images, respectively, and $p$ is the number of pixels in the image.
for the MMSE estimator.

As discussed above, the parameter estimation relies on a setting of a single parameter - the amount of energy to remove from the signals in the crude preliminary denoising stage - and the performance of the estimators relies on the quality of the parameter estimation. In order to run a fair comparison, we varied the value of this parameter in order to optimize the average performance (over all the images in the set) of each estimator individually (i.e., one optimization for the MAP estimator, and another for the MMSE estimator).

One conclusion from these experiments is that for weak noise levels ($\sigma \leq 20$), it is beneficial to remove relatively little energy in the crude denoising stage, e.g., $T = 0.1 \cdot \sigma$, for both estimators. When working on moderate and strong noise, the best choice is $T = 1.05 \cdot \sigma$. This phenomena can be explained mostly by model mismatch, as the model we force on the signals (sparse representation over a unitary dictionary) in itself inserts some “noise” into the estimation process, and therefore the denoising performance when the noise is weak is limited.

A further analysis of the sensitivity to the setting of this parameter has been carried out. When deviating from the optimal choice, even considerably, the MMSE estimator loses at most $0.1 dB$ on average PSNR performance, while the MAP estimator displays a more considerable drop in performance, up to $0.5 dB$. This hints that the MMSE may be more robust to errors in the parameter estimation stage, i.e., it is more robust to model mis-matches.

A visual comparison of the results of the different estimators is presented in Figure 5.8, for “Boat” image to which white Gaussian noise with $\sigma = 30$ has been added. The images are constructed by returning the processed patches to their original location (again, with no overlap). It is well known that increasing the overlap between the patches improves results [2, 3]. We choose to refrain from doing this, as a large overlap between patches introduces an MMSE flavor, regardless of the estimator itself, and it thus partly obscures the differences between the estimators. In order to complete the picture, the parameters estimated for this image appear in Figure 5.9.

5.6 Summary

In this work we discuss the problem of denoising a signal known to have a sparse representation, studying the MAP and the MMSE estimators. We focus on unitary dictionaries,
Figure 5.6: Relative denoising achieved (compared to the noisy signal), averaged over all blocks in 7 images, by the MAP and MMSE estimators.

Figure 5.7: PSNR gap between MMSE and MAP (with positive values indicating the MMSE is better performing), averaged over all blocks in 7 images. The error bars indicate one standard deviation of the gap.
Figure 5.8: A visual comparison of the reconstructed image by the MAP and MMSE estimators, for the different training options, on the center portion of the “boat” image with noise level $\sigma = 30$. 

Ground truth image

Noisy image $PSNR = 18.76dB$

MAP, $PSNR = 25.88dB$

MMSE, $PSNR = 26.30dB$
Standard deviation of coefficients of each atom $\sigma_i$  Individual probabilities of each atom $P_{i}^{a}$

Log-Probability as a function of cardinality ($|S|$)

Figure 5.9: The values of the estimated model parameters, using the block-coordinate-descent method described in Appendix 5.B, for the “Boat” image with noise $\sigma = 30$. The values for $\sigma_i$ and $P_{i}^{a}$ are arranged as $8 \times 8$ arrays, corresponding to the increasing vertical and horizontal frequencies that construct the DCT dictionary. Note that the value for the top-left atom (the DC atom) is much larger than the rest, for both $\sigma_i$ and $P_{i}^{a}$, and its value is “saturated” in both figures.
for which we show that a closed-form, exact, and simple recursive formula exists for the MMSE estimator. This replaces the need for an approximation, such as the Random-OMP algorithm. We show experimentally that this exact MMSE formula outperforms the Random-OMP and the OMP (which is the exact MAP). We also discuss several numerical issues which arise when this formula is implemented in practice.

This work then extends the somewhat limited signal generation model to accommodate real-world signals. We describe how the parameters of this model are estimated, and present experiments in which the parameters are estimated from the noisy signals themselves. We show the clear advantage of the MMSE estimator over the MAP estimator in these tests, both objectively and visually.

The main drawbacks of the work presented here is the complexity of the signal generation model. This complexity leads to two difficulties. The first is that the recursive formula, while relatively efficient, still requires a large number of computations. This also limits the dimensions of the signals to work on, inducing us to work on image patches instead of a full-scale image. The second problem that arises is that the parameter estimation process, while mathematically justifiable, is still relatively weak.

We believe that future work should address different sparse signal generation models, and by doing so, find an even more efficient way to compute the MMSE, and perhaps gain a more stable estimation of the parameters involved. Another possible future direction is obtaining efficient optimal estimators for different types of risks, such as the mean over absolute errors.

5.A Recursive Formula - Monotonicity of Ratios Proof

In this appendix we aim to prove the monotonicity property as described in Section 5.3: Given that $q_j > q_i$, then

$$\frac{q_j^k}{q_i^k} < \frac{q_j^{k-1}}{q_i^{k-1}} \text{ for all } k.$$  

Proof: Let us start by writing out $q_j^k$ as in Equation (5.11). The tuples the $j^{th}$ element participates in are divided into two groups, based on whether the $i^{th}$ element also participates in the tuple or not:

$$q_j^k = \sum_{\{S \in \Omega_i \mid j \in S\}} \prod_{i \in S} q_i$$  

(5.A.1)
where we have denoted:

\[ A^k = \sum_{\{S \in \Omega_k | i, j \in S\}} \prod_{l \in S} q_l, \quad B^k = \sum_{\{S \in \Omega_{k-1} | i, j \notin S\}} \prod_{l \in S} q_l, \]

and note that the two terms are related through: \( A^k = q_i \cdot q_j \cdot B^{k-2} \).

Let us analyze \( B^k \) more closely. This is in fact the sum of products over all tuples of size \( k \) from the elements of \( q \), excluding \( q_i \) and \( q_j \). The sum of these elements, due to the normalization \( \sum_{l=1}^n q_l = 1 \), is \( t = 1 - q_i - q_j \). Let us now denote by \( \{r_m\}_{m=1}^{n-2} \) all the elements of \( q \), excluding \( q_i \) and \( q_j \), and after being multiplied by \( \frac{1}{t} \). The multiplication leads to \( \sum_{m=1}^{n-2} r_m = 1 \). Substituting into \( B^k \) we obtain that:

\[ B^k = t^k \cdot \sum_{S \in \Omega_k} \prod_{l \in S} r_l. \]

Now, we observe that the second term is the sum of products over all \( k - \)tuples of elements from \( r \). Since \( \sum_{m=1}^{n-2} r_m = 1 \), this is exactly the game described in Section 5.3.2. Therefore, we can use the normalization property to claim that \( \sum_{S \in \Omega_k} \prod_{l \in S} r_l = k \), and arrive at:

\[ B^k = t^k \cdot k \quad A^k = q_i \cdot q_j \cdot t^{k-2} \cdot (k - 2) \quad (5.A.2) \]

Going back to what we set to prove, and substitute into Equation (5.A.1), we now need to prove that

\[ \frac{A^k + q_i \cdot B^{k-1}}{A^k + q_i \cdot B^{k-1}} < \frac{A^{k-1} + q_j \cdot B^{k-2}}{A^{k-1} + q_i \cdot B^{k-2}}. \]

Multiplying along the diagonals, removing common terms and rearranging, our new goal to prove is:

\[ (q_j - q_i) \cdot A^{k-1} \cdot B^{k-1} < (q_j - q_i) \cdot A^k \cdot B^{k-2} \]
\[ A^{k-1} \cdot B^{k-1} < A^k \cdot B^{k-2}; \]

where the last step is valid since we know \( q_j > q_i \).

Now we substitute the formulas for \( A^k \) and \( B^k \) given in (5.A.2), changing what we need to prove to:

\[ q_i \cdot q_j \cdot t^{k-3} \cdot (k - 3) \cdot t^{k-1} \cdot (k - 1) < q_i \cdot q_j \cdot t^{k-2} \cdot (k - 2) \cdot t^{k-2} \cdot (k - 2) \]
\[ q_i \cdot q_j \cdot t^{2k-4} \cdot (k - 3) \cdot (k - 1) < q_i \cdot q_j \cdot t^{2k-4} \cdot (k - 2) \cdot (k - 2) \]
\[ k^2 - 4k + 3 < k^2 - 4k + 4, \]

113
which is a true statement, and therefore all the inequalities are true, and we have proven what we set out to prove.

## 5.B Parameter Estimation

### 5.B.1 A Direct Maximum Likelihood Approach

In this appendix, we discuss more elaborately how the parameter estimation process described in Section 5.4.3 is developed. The goal of this stage is to estimate the values of the different parameters of the signal generation model, given a set of noisy signals \( \{y^m\}_{m=1}^M \), or equivalently (due to the unitary dictionary) a set of noisy coefficients \( \{\beta^m\}_{m=1}^M \) (where \( \beta^m = D^T y^m \)). We note again that we assume the variance \( \sigma^2 \) of the noise is known or was estimated by other means.

A possible approach to this problem is the Maximum Likelihood (ML) approach. Our goal is to find a set of parameters \( \Theta = \{\{\sigma_i\}_{i=1}^n, \{P_C(k)\}_{k=1}^n, \{P_a\}_{i=1}^n\} \) which is the most likely to have produced this set of noisy signals. The parameters are then found by solving the following maximization problem:

\[
\hat{\Theta} = \arg \max_{\Theta} \prod_{m=1}^{M} P(\beta^m | \Theta) = \arg \max_{\Theta} \sum_{m=1}^{M} \log (P(\beta^m | \Theta)) ,
\]

where the second step is the maximization of the log-likelihood. The probability of a specific signal to be generated given this set of parameters is

\[
P(\beta^m | \Theta) = \sum_{S \in \Omega} P(\beta^m | S, \Theta) \cdot P(S | \Theta).
\]

This probability is obtained by considering each possible support, and computing the probability that this support generated the signal, multiplied by the probability of the support to have been chosen. The sum over all possible supports is the actual probability of the signal to have been generated from the set of parameters \( \Theta \).

The probability of a signal to be generated, given a known support \( S \), a set of parameters \( \Theta \) and with the Gaussian noise assumption is

\[
P(\beta^m | S, \Theta) = \prod_{j \in S^m} \frac{1}{\sqrt{2\pi(\sigma^2 + \sigma_j^2)}} \cdot e^{-\frac{(a_j^m)^2}{2(\sigma^2 + \sigma_j^2)}},
\]

and the probability of a support to be generated given the set of parameters \( \Theta \) is given in Equation (5.19):

\[
P(S^m | \Theta) = \frac{P_C(|S^m|)}{T_{|S^m|}} \cdot \prod_{i \in S^m} P_a^i ,
\]
with \( T_k = \sum_{S \in \Omega_k} \prod_{i \in S} P_i \) a normalization factor. Note that the probability in Equation (5.B.2) is computed only over the coefficients in the support. Since the support is assumed to be known, we focus only on the coefficients inside the support, and the probability of them being generated, while ignoring the coefficients outside the support (which are known to be 0).

Unfortunately, assigning those into the full ML expression in Equation (5.B.1) yields a highly complex argument, the maximization of which is very complicated (due to both the summation over all supports and the normalization factors \( T_k \)). Therefore, we change our course slightly and turn to a block-coordinate-descent approach, which may assume that the data it operates on is clean.

### 5.B.2 Block Coordinate Descent approach

In the block-coordinate-descent approach, the denoising stage and the parameter estimation stage are carried out alternatingly, where one is estimated while the other is considered known, and vice-versa. In practice, the first stage is a simple denoising mechanism, such as hard-thresholding. An initial crude denoising stage prior to a more complex denoising mechanism is common; see for example [29].

The set of denoised signals - or in fact, the supports found - can then be used to estimate the parameters, again in an ML formulation. Then, a denoising stage can be again carried out, using the explicit signal generation model, obtaining a better result. The new denoised set can then be used to better estimate the parameters, and so on. In the experiments described above, the parameters were estimated only using the initial crude denoising.

Given the denoised signals, how can we estimate the parameters? We assume that instead of the denoised signals, we obtain the hypothesized support for each signal \( S^m \). Inserting the known support for each signal into Equation (5.B.1), we no longer need to sum over all supports and we can use Equation (5.B.2), arriving at the following maximization problem:

\[
\hat{\Theta} = \arg \max_{\Theta} \sum_{m=1}^{M} \log \left( P(\beta^m|\Theta) \right)
\]

\[
= \arg \max_{\Theta} \sum_{m=1}^{M} \log \left( \prod_{j \in S^m} \frac{1}{\sqrt{2\pi(\sigma^2 + \sigma_j^2)}} \cdot e^{-\frac{(\beta_j^m)^2}{2(\sigma^2 + \sigma_j^2)}} \cdot \frac{P_C(|S^m|)}{T_{|S^m|}} \cdot \prod_{i \in S^m} P_i \right)
\]
\[
\arg\max_\Theta \sum_{m=1}^M \left[ \sum_{j \in S^m} \log \left( \frac{1}{\sqrt{2\pi(\sigma_j^2 + \sigma_m^2)}} \cdot e^{-\frac{(x_m^m)^2}{2(\sigma_j^2 + \sigma_m^2)}} \right) 
+ \log(P_C(|S^m|)) - \log(T_{|S^m|}) + \sum_{i \in S^m} \log(P_i^a) \right].
\]

This argument can be divided into four separate sums: the first depending only on \(\{\sigma_i\}_{i=1}^n\), the second depending only on \(\{P_C(k)\}_{k=1}^n\) and the third and fourth depending only on \(\{P_i^a\}_{i=1}^n\). Therefore, we can separate the maximization problem into three parts, and recover each set of parameters.

The values of \(\{\sigma_i\}_{i=1}^n\) are recovered by taking a derivative of the first term, and finding the zero crossing. This gives rise to \(n\) independent maximization problems. We omit this straightforward (but tedious) procedure, which eventually leads to:

\[
\hat{\sigma}_k = \sqrt{\frac{\sum_{m: k \in S^m} \left( \beta_k^m \right)^2}{|\{m | k \in S^m\}|} - \sigma^2}.
\]

Maximizing over \(\{P_C(k)\}_{k=0}^n\), we must remember the constraint that \(\sum_{k=1}^n P_C(k) = 1\).

For this constrained maximization, we use lagrange multipliers. Again, we jump straight to the result, being

\[
\hat{P}_C(k) = \frac{|\{m | |S^m| = k\}|}{M}.
\]

The last part of finding \(\{P_i^a\}_{i=1}^n\) is more challenging compared to the first two, and a closed-form solution is not available, because of the existence of \(T_{|S^m|}\) inside the formula, which requires the application of the recursive formula at each evaluation of the function. Instead, we shall try to maximize the function value using gradient ascent. In order to simplify the notation, we now denote \(z_i \equiv P_i^a\), and the function to maximize is

\[
F = \sum_{m=1}^M \left[ -\log(T_{|S^m|}) + \sum_{i \in S^m} \log(z_i) \right].
\]

We denote by \(M_k^a\) the number of supports containing the \(k^{th}\) atom, and by \(M_k^C\) the number of supports of size \(k\). Now we can rewrite this function as

\[
F = -\sum_{k=1}^n M_k^C \cdot \log(T_k) + \sum_{k=1}^n M_k^a \cdot \log(z_k).
\]

We rewrite \(T_k\) as a function of \(z_i\)

\[
T_k = \sum_{S \in \Omega_k} \prod_{l \in S} z_l = z_i \cdot \sum_{\{S \in \Omega_{k-1} | l \notin S\}} \prod_{l \in S} z_l.
\]
From this last step it can be seen that only the first term depends on $z_i$. Taking a derivative of $T_k$ with respect to $z_i$ leads therefore to

$$\frac{\partial T_k}{\partial z_i} = \sum_{\{S \in \Omega_{k-1} | i \notin S\}} \prod_{l \in S} z_l - \sum_{\{S \in \Omega_{k-1} | i \in S\}} \prod_{l \in S} z_l = T_{k-1} - \sum_{\{S \in \Omega_{k-1} | i \in S\}} \prod_{l \in S} z_l$$

Now, we remind ourselves of the recursive formula introduced in Section 5.3.1. This formula allows us to efficiently compute $Z_i^k = \sum_{S \in \Omega_k} \prod_{l \in S} z_l$. Observing that $T_k = \frac{1}{k} \sum_{l=1}^{N} Z_i^k$, we get a simple formula for computing the function value in 5.B.3, as well as a simple way to compute the derivative:

$$\frac{\partial F}{\partial z_i} = -\sum_{k=1}^{n} M_k^C \cdot \frac{T_{k-1} - Z_i^{k-1}}{T_k} + \sum_{k=1}^{n} \frac{M_k^a}{z_i}$$

An efficient initialization for this maximization problem is $z_i = M_i^a / M$, which is the relative number of supports the $i^{th}$ atom appears in divided by the total number of supports. While this is not the optimal choice, it is quite near, and in both synthetic experiments (done to validate the parameter estimation process) and real-world experiments, the change of the values in the optimization problems was very mild. Since the initial denoising is inaccurate, it makes sense not to try to obtain extreme accuracy for $\{P_i^a\}_{i=1}^{n}$, and instead remain with the initial estimate suggested here. The synthetic and real-world experiments demonstrate that indeed, the results obtained by the two options are extremely close.
Chapter 6

Discussion

6.1 Motion-Estimation-Free Denoising of Image Sequences

In their paper describing the application of the Non Local Means (NLM) algorithm to denoising of image sequences [17], Buades, Coll and Morel claim "Denoising of image sequences does not require motion estimation". As an extension of the bilateral filter [41, 122], the NLM was shown to perform denoising of image sequences quite well.

In our paper [100], given in Chapter 2, we suggest an image sequence denoising algorithm, which extends the sparse and redundant representation-based image denoising algorithm [2, 3] to image sequences. The resulting algorithm, like the NLM, performed denoising of image sequences without any motion estimation. Objective tests, comparing to various methods considered to be state of the art, had been carried out, and are detailed in Tables 2.1, 2.2 and 2.4. These tests demonstrate that the proposed method out-performs any motion-estimation based denoising method and motions-estimation-free algorithms such as the NLM method. Only the BM3D [29], another motion-estimation-free denoising algorithm, displayed comparable results.

It is most interesting to observe Figure 2.2 displaying several atoms from the dictionary. It is clear that the dictionary "learns" motion patterns appearing in the sequence. The atoms shown display several speeds of motion towards the left. The sequence also displays such motions, as the overall motion is to the left, with varying speeds stemming from the different distances from the camera.

Therefore, the denoising is obtained by learning motion patterns existing in the sequence in the training stage. For the actual denoising, each space-time patch is assigned a combination of just a few motion patterns. There is no straightforward way to infer the
actual motion vectors from this mechanism, making this algorithm completely motion-estimation free.

The question this research has left us with is whether the ability to process image sequences without motion estimation can also be applied to more complex tasks, where the estimation of motion is even more complex and therefore even more limiting. We turned to one of the most complicated tasks - Super-Resolution - where motion estimation was deemed necessary for the 20 years this field had existed.

6.2 Motion-Estimation-Free Super-Resolution

Our journey started with the NLM filter. When applied to image sequence denoising, it may be thought of as using probabilistic motion estimation. For each pixel to be denoised, each of the pixels in its space-time neighbourhood is assigned a weight, used to compute a weighted average of all the neighbouring pixels. The resulting value represents the denoised value of the pixel.

The weights (should) reflect the crude probability that each of the neighbouring pixels has the same original value as the pixel to be denoised. Pixels with a large probability should be assigned a large weight, as we want them to have a large effect on the output. Pixels with low likelihood should be assigned a low weight, as to have little effect on the overall result. Thus, when looking at the entire weights map computed for a specific pixel, locations with large weights represent locations which are likely destinations for the pixel. This weights map (when properly normalized) can therefore be viewed as probabilistic motion estimation for the pixel. Instead of finding just one location in every image, many locations are allowed, with varying probabilities.

The field of super-resolution, since its introduction over 20 years ago, has assumed that motion estimation is a key first step for improving the resolution of image sequences. If the accuracy of the motion estimation is not within fractions of a pixel, the reconstructed result is likely to look much worse even when compared to the degraded input.

We have found that the notion of probabilistic motion estimation - through the NLM - can be incorporated into super-resolution. By first designing a mathematical formulation for the NLM (writing it as a prior), and carefully extending it to perform SR, we arrived at a simple formula (3.29). Each pixel of the super-resolved output is computed as a weighted average of pixels from the low-resolution images, in a space-time neighbourhood
of its low-resolution location. This formulations seems very similar to that of the original NLM, but with the different scales taking non-trivial roles.

It turned out that avoiding explicit (and error-prone) motion estimation by using NLM-like probabilistic motion estimation can process sequences where the motion is complex. Such sequences were traditionally avoided by the motion-estimation-based algorithms. Several results can be seen in Figures 3.6, 3.8, 3.9, and 3.10. The reliance on aggregating many motion trajectories allows more robustness to errors, as even if a few false trajectories are assigned large weights, they are offset by many correct trajectories inserted into the weighted average. Furthermore, using many trajectories allows taking advantage not only of the temporal redundancy but also of the spatial redundancy, thus performing super-resolution even in static areas (which cannot be done in classic super-resolution). This method was the first super-resolution method able to perform SR on sequences with general motion patterns. Following it, several more SR methods avoiding motion-estimation have also been proposed, relying on BM3D [30], on kernel regression [120] and more.

While the extension of the NLM to SR brought very successful results, we tried to find a more intuitive path to incorporating probabilistic motion into super-resolution. In Chapter 4, we have shown that it is possible to alter the classical SR framework, and replace the reliance on explicit motion estimation with a weighted aggregation of all possible trajectories. Under some simplifying (and not at all necessary) assumptions, this approach has led to the same simple, local weighted average, as the approach based on the NLM. Equations (3.29) and (4.17) are in fact the same. However, the two methods are by no means identical, as had we made different assumptions, the resulting algorithms would have been different.

The added benefit of introducing probabilistic motion estimation into classic SR framework is the ease in which it is adapted to perform other re-sampling tasks. We have shown in Figure 4.3 that with only slight modifications, it is possible to solve the de-interlacing problem with a high degree of success.

The introduction of probabilistic motion into SR has raised an interesting perspective. While it is known that there exists a true motion field between the images, it is better (in this case) to avoid attempting an estimation of the true motion, relying instead on probabilistic estimation. By doing this, we replaced an explicit model with an implicit one. This intuition has led us to work on the MMSE estimator.
6.3 MAP and MMSE

We returned to the denoising task, as a simple test-bed for checking our model. We also employed the sparse and redundant representations modelling for the signals we work on. Unlike the work on image sequence denoising, where the model - the dictionary - was also learned, the question we sought to answer in this research was: given the noisy signal, and the dictionary it was created from, what is the best way to denoise the signal?

In denoising images [2, 3] or image sequences [100], when denoising a signal, we attempted to find the sparsest signal (over the dictionary) that is close enough to the noisy signal for the difference to be explained by the noise. Since in the general case, this problem is NP-hard, it is not possible to actually find the sparsest representation, and instead, an approximation was used - the OMP algorithm [21, 92].

In an interesting experiment by Yavneh and Elad, instead of the deterministic OMP algorithm, many sparse representations that are able to explain the signal were generated using a randomized version of the OMP algorithm, termed Rand-OMP [49]. It turned out that averaging these many representations led to much better denoising results compared to using only the sparsest representation obtained by the OMP. One possible cause of this difference is that the OMP is actually likely to choose the wrong atoms, even if the final outcome is similar to the original, high quality representation.

An analysis of this phenomena in [49] has led to the understanding that the OMP actually approximates the MAP estimator, in effect looking for the representation most likely to explain the noisy signal. As explained, due to the NP-hardness of the problem, it is not possible to recover the true MAP estimator, and an approximation is used.

The averaging of the many representations approximates the MMSE estimator, which seeks to minimize the expected error of the result [49]. A straightforward (yet technically impossible for any reasonably sized problem) way to compute the exact MMSE is to aggregate over all possible representations, with weights reflecting the probability of each sparse representation to explain the noisy signal. The randomization in the OMP generates representations with a probability that is related to their probability to explain the noisy signal. Therefore, a plain average is all that is needed to approximate the MMSE.

This description bears resemblance to the intuition gained in the SR case - while an explicit model exists, its estimation is too hard and error prone. Therefore, it is better to use an implicit model. The implicit model takes the form of a weighted average of
"explanations", where each is assigned a weight reflecting its probability of being the true "explanation". In the denoising case, the same intuition applies - instead of looking for the one correct representation, many likely ones are used.

As the need to approximate the MAP and MMSE bothered us, we have tried to seek cases in which recovering their exact values are possible. We discovered that when the dictionary is unitary, it is possible to come up with a closed-form formula (5.13) that computes the MMSE exactly. The MAP also enjoys a simple formula, and it too no longer requires approximation. Not surprisingly, as in the general case, the MMSE estimator considerably out-performs the MAP estimator, as can be seen in Figures 5.1, 5.2, 5.3, 5.4, 5.6 and 5.7.

6.4 Summary and Future Directions

The main theme of this research has been to show that restoration of image sequences is possible while avoiding the need for explicit, error-prone, hard to compute motion estimation. It has been shown in both simple (denoising) and complex (super-resolution) scenarios that avoiding motion estimation is a viable path and is usually more successful than motion-estimation-based techniques.

Continuing along the lines laid out in the thesis, there are several possible directions to extend these ideas to new tasks and improved performance:

• Image Sequence Deblurring: While deblurring seems to be a special case of super-resolution (without decimation), the handling of the deblurring part in the proposed algorithms in Chapters 3 and 4 is actually (somewhat intentionally) poor, as it is done on each image individually. It could be very beneficial to design an image sequence deblurring algorithm, relying on the same principles of avoiding motion estimation.

• Sparse and Redundant Representations for Super-Resolution: For image sequence denoising, the model of sparse and redundant representations offered state-of-the-art results, and specifically better than the NLM algorithm. This leads us to believe that an SR algorithm relying on this model will also improve upon the results presented here. While sparse and redundant representations had been harnessed to single-image SR (a fundamentally different task) [134, 135], it has not yet been applied to multi-frame SR.
• MMSE and NLM: The MMSE estimator bares similarity to the NLM, with averaging many signals with weights common to both. A further understanding of the links between the two might improve the results achieved by both approaches.

• Harnessing MMSE to image denoising: The potential of denoising signals using the MMSE estimator had been shown in [49] and in Chapter 5. However, initial attempts to apply the MMSE for denoising of images show only a very mild benefit. We believe this direction should be further studied, as it may contain untapped potential.

• Complexity: One major drawback shared by the various algorithms presented in this thesis is that they are all very computationally demanding. Improving the efficiency of the proposed algorithms would greatly improve their applicability. Some of the algorithms lend themselves naturally to parallel implementations, possibly offering much reduced run-times.

• 3D Reconstruction: Just as super-resolution had relied on motion estimation, 3D reconstruction also relies on estimating motion between two or more frames, in order to compute the 3D location of viewed objects. As in super-resolution, the need for motion estimation limits the accuracy that can be achieved by the reconstruction. Since in 3D reconstruction, as in SR, motion estimation is not the end-product but only the means, we wonder if it can also be avoided, while allowing improved results.

In a wider perspective, in all the research done here, implicit models replaced the usage of explicit ones. In image sequences, even though the true motion field indeed exists, improved results are obtained by using implicit models to describe it, either using sparse and redundant representations or the concept of probabilistic motion estimation. Even in the denoising of sparse signals, when it is known that the signal indeed has a sparse representation, it is better to denoise using a multitude of sparse representations than settling for the sparsest one alone.

The approach of replacing explicit models with implicit ones seems to us as an interesting course to consider when tackling various problems. It may be helpful in overcoming barriers traditionally thought too hard to overcome, as was done in super-resolution.
Bibliography


137


ל تعالى ספורט-רולציצי, חתוש ניקי האותות מרחה הנפשות ייחודי, כש כל היחסות לדבוק
במיסוגים של ייצוג נונחי שונת. על, 커, וחסונת הדוכ הזדמאתם באזירת המזרחיים לע פיתולים של
ייצוגים נומרי דקלים שלא קדים. מכשא חירי עד עותק רעה אך עד מדו, זה, השיט פורח
לקות מرياضة היה הקתשות שהוים 고וד דקלים בחוא פאפרה על ספלי המל ReadOnly, זה שיתפיש
בן舨זון והשוח 않ב ס癖 על ידי בזמת השג שית ושמתש מהפרת את הדרכ הבנה השואת המקריר, למסי המפרש את השואת של השבב חור התזות
הרועש, ישנה המณะ שערוע. על,וא, חישון היא הבשר ייחודי, היה אל אפשרי המבחנה
היישובית, על כל שיש זורק בקורוב.

במאז הודות לכל ישנה שייחות ייחודי, הב מיצוגים באנפ אקריא-למחזית ממסר ייצוגי דקלים
שכלם מס eiusים את ההוא רעשה. מישעי על ייצוגים אלה יוני תיב (מסatorialית) מחיריםית שוחר
MMSE קודה מביג דקלים ביית. הדול עי נמצ אפיות הייצוג מקראב את משער וש
שמטרתי❌¼ עותק את השואת החפירה (על כsmouth התצאותו ניקו סובע עובח עותי מתחופ לשאר את
הספרס ביוית). במ דא חור כויש שוט מועך הז מייש כייש מבחנה השיתוב.

בנобще הדונות המופתואים בביית הוא זע ממסתיבות על ממסים. וא, פאני סעייר ממקן המנני
 isError הוא מזון ממיד - ייטארו - כי דא צורב בקורוב על הששוית, כיום שאו צה
לгонי להטיך ס㌔ה והשעור ייסית ליחובב המודיק. כלו כוא ז הללו על סמר ניסיון
MMSE הספרד על הינדוס ואתיתות שעינה במקנה הז משער הז שער חלי שער
מיי מוסי, כיי ישון ביצועים מושרים.
ניתן מתוך אחת למשימות נוסף בקריאת תמונות. כך, ניתן אף לתמונות נוספים בגרישה גם סמכית. זו התמצאה בłąךโรค תלה תמונות (הנוסף) מכיוון של ציון, והעדרים, במקביל פ$ar שאותו בتأمين (ולאחר). ב Lansing, וב፳ נתח הרOfYear (למשל). נורא ופרסים ראש لهذا מקום לארוג בכר המחודש. במשנה, תקסט עם קריא בגרisure הלקט.

הוןakah לארוג בכר המשנה.

תקציר

_teleography
technology

This work develops an innovative method for handling telegraphs, using advanced techniques in computer science. The method involves the analysis of large sets of telegraph data, employing sophisticated algorithms for pattern recognition and noise reduction. The approach is particularly effective in dealing with the challenges posed by historical telegraph data, which often contain errors and inconsistencies. The work has been conducted at the Technion's Computer Science Department, as part of a Ph.D. thesis submitted in 2011.
המחק עשה בהנחיית פרוף מיכאל אלעל
בפקולטה למדעי המחשב

בראש בראשון, אני מבקש להודות להנחתה של פרוף מיכאל אלעל. היא ידעה מה旎יلاحוק אחרי קדימה ונחמתلاحוק חזה עד יוחר. היא כל מה ישולחת בברך במנחה.

אני מבקש להודות всем חברים הפקולטה שיצרו עבורי אתים ואלמוד הבקורסים שלתם. למדתי הרבה מכל אחד מהם.

אני רוצה להודות всем חברים, חנה ומשה, על שלימודי אחריות עבורי קושי, על שעדדו "{אודות-Benzing}" לשאוף להיזון בכול מה שאיני יכול להזינו, על נג諾ות המתחדויותطرفית. לעבורי, אני מודה להזינו לחיים אורתז, על כל.

שחייתו למית לכל doch תומכה מובנה.

אני מודה לך קרן ייקובס-קואלוקס
על התרומת הכספים הנדרשים בהשתלמותי.
עיבוד רצפיים של תמונות
للאתרים/Shurot Suchot

تحرير על מחקר

לשם مليובי החלק של הדרישה לתכנית הלימודים
ויקטור פילוסופיה

מתן פרוטור

הנהלת לטכנולוגיה - מכון טכנולוגי לישראל
שбот תשע"יא חינוך ונאות 2011
עיבוד רפפים של תמונות
ללא שעורים ותנודות

מוח פורטר